

**IMPLEMENTASI ALGORITMA GENETIKA DALAM
APLIKASI TIMETABLING JADWAL MENGAJAR ASISTEN
LABORATORIUM KOMPUTASI DAN SISTEM CERDAS
TEKNIK INFORMATIKA UNIVERSITAS ISLAM INDONESIA**

LAPORAN TUGAS AHIR

*Diajukan Sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana Teknik Informatika*



Disusun Oleh:

Nama : Ivan Adhi Prasetya

NIM : 10523006

**JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA**

LEMBAR PENGESAHAN PEMBIMBING

**IMPLEMENTASI ALGORITMA GENETIKA DALAM
APLIKASI TIMETABLING JADWAL MENGAJAR ASISTEN
LABORATORIUM KOMPUTASI DAN SISTEM CERDAS
TEKNIK INFORMATIKA UNIVERSITAS ISLAM INDONESIA**

LAPORAN TUGAS AKHIR

Disusun oleh:

Nama: Ivan Adhi Prasetya

NIM: 10523006

Yogyakarta, 7 Mei 2014

Telah Diterima dan Disetujui dengan Baik Oleh:

Dosen Pembimbing

Zainudin Zukhri, S.T, M.IT

LEMBAR PRNGESAHAN PENGUJI

**IMPLEMENTASI ALGORITMA GENETIKA DALAM
APLIKASI TIMETABLING JADWAL MENGAJAR ASISTEN
LABORATORIUM KOMPUTASI DAN SISTEM CERDAS
TEKNIK INFORMATIKA UNIVERSITAS ISLAM INDONESIA**

TUGAS AKHIR

OLEH:

Nama: Ivan Adhi Prasetya

NIM: 10 523 006

Telah Dipertahankan di Depan Sidang Penguji Sebagai Salah satu Syarat Untuk
Memperoleh Gelar Sarjana Jurusan Teknik Informatika.

Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 7 Mei 2014

Tim Penguji,

Zainudin Zukhri, S.T., M.IT

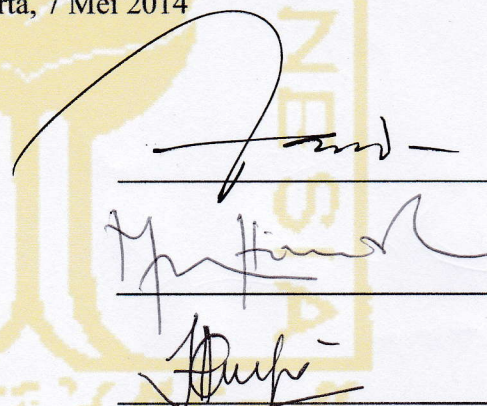
Ketua

Izzati Muhimmah, S.T., M.Sc., PhD.

Anggota I

Taufiq Hidayat, S.T., M.Sc.

Anggota II



Mengetahui,

Ketua Jurusan Teknik Informatika



Yudi Prayudi, S.Si., M. Kom.

LEMBAR PERNYATAAN KEASLIAN
HASIL TUGAS AKHIR

Saya yang bertanda tangan di bawah ini,

Nama : Ivan Adhi Prasetya

NIM : 10523006

Menyatakan bahwa seluruh komponen dan isi dalam laporan Tugas Akhir ini adalah hasil karya sendiri. Apabila di kemudian hari terbukti bahwa ada beberapa bagian dari karya ini adalah bukan hasil karya saya sendiri, maka saya siap menanggung resiko dan konsekuensi apapun.

Demikian pernyataan ini saya buat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 7 Mei 2014

Ivan Adhi Prasetya

HALAMAN PERSEMBAHAN

Karya sederhana ini Saya persembahkan untuk keluargaku tercinta,

Ibu Saya Niniq Cahyani Suminarti Handayani

Ayah Saya Haryo Manon Sejati

Kakak Saya Erfika Gea Calista Farhani

Adik Saya Andre Surya Pramudya dan Syifa Natania Farhani

Kakak Ipar Saya Guntur Putra

Keponakan Saya Raffa Arganta Putra

HALAMAN MOTTO

*Jika Anda telah memiliki impian,
berilah tanggal bagi pencapaiannya,
dan impian itu akan menjadi cita-cita.*

Ingatlah, bahwa

CITA-CITA ADALAH IMPIAN YANG BERTANGGAL.

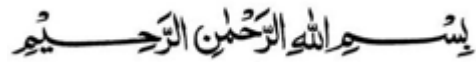
*Kemudian,
Anda tinggal menyusun urutan tindakan
untuk mencapainya pada tanggal itu.*

*Sehingga,
semakin Anda menunda,
semakin tanggal itu terdorong menjauh.*

*Dan semakin Anda malas,
semakin cita-cita itu menjadi tidak berarti.*

(Mario Teguh)

KATA PENGANTAR



Assalamu'alaikum Wr. Wb

Alhamdulillahirobbil'alamiin, segala puji dan syukur kepada Allah SWT, atas segala rahmat, hidayah dan karunia-Nya, sehingga penulisan laporan Tugas Akhir yang berjudul Implementasi Algoritma Genetika Dalam Aplikasi Timetabling Jadwal Mengajar Asisten Laboratorium Komputasi Dan Sistem Cerdas Teknik Informatika Universitas Islam Indonesia ini dapat diselesaikan dengan baik. Shalawat dan salam selalu tercurah kepada Nabi Muhammad SAW.

Laporan Tugas Akhir ini dibuat sebagai salah satu syarat yang harus dipenuhi untuk memperoleh gelar sarjana di Jurusan Teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia. Laporan Tugas Akhir juga sebagai sarana untuk mempraktekkan secara langsung ilmu dan teori yang telah didapatkan selama menjalani perkuliahan di Jurusan Teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia.

Melalui Tugas Akhir ini penulis dapat mempelajari pembuatan implementasi teori Algoritma Genetika dengan baik yang disertai dengan dokumentasi, sehingga penelitian tugas akhir ini dapat dikembangkan lebih lanjut oleh peneliti yang lain. Selama melakukan penelitian tentu penulis mengalami beberapa kesulitan, namun kesulitan tersebut dapat teratasi berkat dukungan dari beberapa pihak. Ucapan terimakasih Penulis diucapkan kepada:

1. Allah SWT, atas segala rahmat dan karunia-Nya, sehingga penulis diberikan kesehatan dalam menyelesaikan Tugas Akhir ini, dan dipertemukan dengan orang-orang yang membantu penulis baik secara langsung maupun secara moral.
2. Orang tua penulis, Ibu Niniq Cahyani Suminarti Handayani dan Bapak Haryo Manon Sejati, atas semua do'a, nasehat, kasih sayang, dukungan, dan segala yang diberikan.

3. Bapak Ir. Gumbolo Hadi Susanto, M.Sc., selaku Dekan Fakultas Teknologi Industri Universitas Islam Indonesia dan seluruh jajaran Dekanat Fakultas Teknologi Industri.
4. Bapak Yudi Prayudi, S.Si., M.Kom., selaku Ketua Jurusan Teknik Informatika.
5. Bapak Zainudin Zukhri, S.T., M.IT., selaku pembimbing yang telah memberikan bimbingan, pengarahan, saran, dan semangat selama pengerjaan Tugas Akhir dan laporan ini.
6. Kepala Laboratorium Komputasi dan Sistem Cerdas Ibu Izzati Muhimmah. S.T., M.Sc., Ph.D. yang selalu mengingatkan tentang Tugas Akhir kepada Saya.
7. Dosen-dosen Jurusan Teknik Informatika Universitas Islam Indonesia dan seluruh jajaran staf jurusan atas semua ilmu pengetahuan, saran, motivasi dan bantuannya.
8. Devika Zanzabilla, terima kasih atas do'a, dukungan, semangat dan bantuannya.
9. Teman-teman 1 kelas, teman-teman 1 angkatan, teman-teman SSJ48, teman-teman KOSMIK, dan teman-teman Tim Promo FTI UII yang menghibur dan mengisi waktu jenuh selama pengerjaan Tugas Akhir ini.
10. Teman dekat Saya Makhfuzi Fahmi yang telah membimbing saya belajar dan memberikan banyak ilmu serta membagi pengalamannya kepada Saya dari Semester 1 sampai pengerjaan Tugas Akhir ini selesai.
11. Teman-teman Laboratorium Komputasi dan Sistem Cerdas (KSC). Mas Imam, Mas Alvin, Mas Fadil, Mbak Wulan, Mbak Novi, Uzi, Yopi, Deni, Dadang, Lina, Izum, Aul, Ito, Aab, Deka, Kiki, dan Peye, terima kasih atas kerja keras, pelajaran dan kekeluargaan teman-teman sekalian.
12. Seluruh teman-teman yang mengerjakan Tugas Akhir di Tahun Ajaran ini yang telah memberikan informasi dan motivasinya kepada Saya.
13. Semua pihak yang tidak dapat disebutkan satu persatu yang telah membantu sehingga laporan tugas akhir terselesaikan.

Terima kasih kepada semua pihak yang telah membantu terselesaikannya penulisan laporan Tugas Akhir ini. Semoga Allah SWT melimpahkan rahmat dan hidayahNya serta membalas semua kebaikan yang telah diberikan. Penulis menyadari bahwa dalam penyusunan laporan Tugas Akhir ini masih banyak terdapat kekeliruan dan kekurangan. Untuk itu penulis menyampaikan permohonan maaf dan diharapkan kritik dan saran yang membangun untuk penyempurnaan di kesempatan berikutnya.

Akhir kata semoga laporan ini dapat bermanfaat bagi kita semua. Amin.
Wassalamu'alaikum Wr. Wb

Yogyakarta, 7 Mei 2014

Ivan Adhi Prasetya

SARI

Timetabling sering kali menjadi permasalahan di beberapa institusi. Penyusunan *timetable* dapat menjadi masalah besar jika data *timetable* yang harus disusun memiliki banyak aturan atau memiliki data yang besar. Masalah tersebut dapat menjadi sebab penyusunan *timetable* membutuhkan waktu yang lama. Permasalahan *timetable* pun sudah dapat dibantu dengan beberapa perangkat lunak. Perangkat lunak tersebut dibuat untuk permasalahan *timetable* yang spesifik pada *timetable* tertentu. Tugas Akhir ini merupakan implementasi Algoritma Genetika pada perangkat lunak *timetabling* mengajar Asisten di Laboratorium Komputasi dan Sistem Cerdas. Perangkat lunak yang akan dibuat membutuhkan analisis kebutuhan perangkat lunak dan perancangan perangkat lunak. Analisis kebutuhan dilakukan menggunakan metode terstruktur dan perancangan dilakukan menggunakan *flowchart*. Setelah implementasi selesai kemudian dilakukan analisis kinerja perangkat lunak dengan pengujian normal dan tidak normal serta analisis waktu kompilasi perangkat lunak. Hasil penelitian Tugas Akhir ini adalah Algoritma Genetika mampu diterapkan dalam pembuatan *timetable* mengajar Asisten laboratorium Komputasi dan Sistem Cerdas. Analisis representasi kromosom yang tepat juga akan dilakukan agar implementasi dapat lebih optimum.

Kata Kunci:

Algoritma Genetika, Optimasi, *Timetabling*.

TAKARIR

<i>advanced</i>	tingkat lanjut
<i>algorithm</i>	algoritma/ kumpulan perintah untuk menyelesaikan suatu masalah
<i>alleles</i>	nilai suatu gen.
<i>bug</i>	kesalahan program
<i>chromosome</i>	kromosom
<i>crossover</i>	penyilangan kromosom
<i>database</i>	kumpulan data yang disimpan secara sistematis di dalam komputer dan dapat diolah atau dimanipulasi menggunakan perangkat lunak (program aplikasi) untuk menghasilkan informasi
<i>decoding</i>	menerjemahkan sandi
<i>delete</i>	hapus
<i>encoding</i>	penyandian/ pengkodean
<i>file</i>	arsip
<i>fitness</i>	nilai yang diberikan untuk solusi yang terbuat
<i>flowchart</i>	diagram alir
<i>genes</i>	(gen) bagian dari kromosom yang mewakili individu
<i>genetic</i>	sesuatu yang berhubungan dengan keturunan
<i>genotype</i>	keadaan genetik dari suatu individu atau sekumpulan individu
<i>hard-constraint</i>	aturan yang wajib ditaati
<i>heap</i>	memori virtual pada bahasa pemrograman java
<i>import</i>	suatu kegiatan pemindahan barang dari tempat lain ke tempat kita
<i>input</i>	masukan

<i>interview</i>	metode pengumpulan data dengan jalan tanya jawab lisan secara sepihak yang dikerjakan secara sistematis dan berlandaskan pada tujuan penelitian
<i>locus</i>	posisi suatu gen.
<i>memory</i>	tempat penyimpanan data
<i>mutation</i>	mutasi
<i>penalty</i>	hukuman
<i>phenotype</i>	suatu karakteristik baik struktural, biokimiawi, fisiologis, dan perilaku yang dapat diamati dari suatu organisme yang diatur oleh <i>genotype</i> dan lingkungan serta interaksi keduanya.
<i>project</i>	proyek/ pembangunan/ rancangan
<i>random</i>	acak
<i>schedule</i>	jadwal yang memiliki urutan proses
<i>selection</i>	seleksi
<i>size</i>	ukuran
<i>soft computing</i>	segolongan metoda yang mampu mengolah data dengan baik walaupun di dalamnya terdapat ketidakpastian, ketidakakuratan maupun kebenaran parsial.
<i>soft-constraint</i>	aturan yang boleh dilanggar
<i>terminology</i>	istilah/ peristilahan
<i>timetable</i>	jadwal yang tidak memiliki urutan proses
<i>timetabling</i>	pembuatan <i>timetable</i>
<i>user</i>	pengguna

DAFTAR ISI

HALAMAN JUDUL	i
LEMBAR PENGESAHAN PEMBIMBING	ii
LEMBAR PRNGESAHAN PENGUJI.....	iii
LEMBAR PERNYATAAN KEASLIAN	iv
HASIL TUGAS AKHIR.....	iv
HALAMAN PERSEMBAHAN	v
HALAMAN MOTTO.....	vi
KATA PENGANTAR.....	vii
SARI	x
TAKARIR.....	xi
DAFTAR ISI.....	xiii
DAFTAR GAMBAR	xv
DAFTAR TABEL	xvii
BAB I PENDAHULUAN.....	1
1.1. Latar Belakang.....	1
1.2. Rumusan Masalah	2
1.3. Batasan Masalah	2
1.4. Tujuan Penelitian	3
1.5. Manfaat Penelitian	3
1.6. Metodologi.....	3
1.7. Sistematika Penulisan	5
BAB II LANDASAN TEORI	6
2.1. Timetable	6
2.2. Algoritma Genetika	8

BAB III ANALISIS KEBUTUHAN DAN PERANCANGAN PERANGKAT LUNAK	13
3.1. Analisis Kebutuhan Perangkat Lunak	13
3.2. Perancangan Perangkat Lunak.....	15
3.3. Metode Pengujian.....	29
BAB IV IMPLEMENTASI DAN ANALISIS KERJA PERANGKAT LUNAK	31
4.1. Implementasi Perangkat Lunak	31
4.2. Analisis Kinerja Perangkat Lunak	34
4.3. Kelebihan Perangkat Lunak.....	47
4.4. Kekurangan Perangkat Lunak	47
BAB V KESIMPULAN DAN SARAN	49
5.1. Kesimpulan	49
5.2. Saran	49
DAFTAR PUSTAKA	50

DAFTAR GAMBAR

Gambar 2.1. Penyilangan Satu Titik.....	11
Gambar 2.2. Gambar Contoh Mutasi.....	11
Gambar 2.3. <i>Flowchart</i> Algoritma Genetika	12
Gambar 3.1. Hubungan Antar Kelas A.....	18
Gambar 3.2. Hubungan Antar Kelas B.....	18
Gambar 3.3. Contoh Kromosom	19
Gambar 3.4. <i>Flowchart</i> Perhitungan Nilai Pelanggaran <i>Constraint</i> 1 (M1) ..	21
Gambar 3.5. <i>Flowchart</i> Perhitungan Nilai Pelanggaran <i>Constraint</i> 2 (M2) ..	21
Gambar 3.6. <i>Flowchart</i> Perhitungan Nilai Pelanggaran <i>Constraint</i> 3 (M3) ..	22
Gambar 3.7. <i>Flowchart</i> Perhitungan Nilai Pelanggaran <i>Constraint</i> 4 (M4) ..	22
Gambar 3.8. <i>Flowchart</i> Perhitungan Nilai Pelanggaran <i>Constraint</i> 5 (M5) ..	23
Gambar 3.9. <i>Flowchart</i> Seleksi Kromosom	25
Gambar 3.10. <i>Flowchart</i> Crossover (Penyilangan)	26
Gambar 3.11. <i>Flowchart</i> Mutasi.....	27
Gambar 3.12. Tampilan Menu Asisten.....	28
Gambar 3.13. Tampilan Menu Kelas.....	28
Gambar 3.14. Tampilan Menu Kelas yang Tidak Bisa Diampu Asisten.....	29
Gambar 4.1. Antar Muka Database Asisten	32
Gambar 4.2. Antar Muka <i>Database</i> Kelas	32
Gambar 4.3. Antar Muka <i>Database</i> Kendala Jadwal.....	33
Gambar 4.4. Antar Muka Pengaturan Nilai Prioritas Aturan	33
Gambar 4.5. Antar Muka Pembuatan Jadwal	34
Gambar 4.6. Antar Muka Daftar Fitnes Setiap Generasi.....	34

Gambar 4.7. Grafik <i>Fitness</i> Pengujian Nomor 2	39
Gambar 4.8. Grafik <i>Fitness</i> Pengujian nomor 3	40
Gambar 4.9. Grafik <i>Fitness</i> Pengujian Nomor 5	40
Gambar 4.10. Jadwal Terbaik Pengujian Nomor 2	41
Gambar 4.11. <i>Output</i> Laporan Pengujian Nomor 2.....	41
Gambar 4.12. Hasil Jadwal Pengujian Nomor 3.....	43
Gambar 4.13. <i>Output</i> Laporan Pengujian nomor 3	44

DAFTAR TABEL

Tabel 2.1.	Perbedaan Penelitian Sebelumnya.....	7
Tabel 3.1.	Tabel Contoh Daftar Kelas dan Asisten	19
Tabel 3.2.	Tabel Constraint.....	20
Tabel 4.1.	Data Asisten.....	35
Tabel 4.2.	Data Kelas.....	36
Tabel 4.3.	Kendala Kelas Setiap Asisten	36
Tabel 4.4.	Hasil Pengujian	38
Tabel 4.5.	Status Jadwal.....	38
Tabel 4.6.	Tabel <i>Likelihood of Optimality</i>	46
Tabel 4.7.	Tabel <i>Average Fitness</i>	47

BAB I

PENDAHULUAN

1.1. Latar Belakang

Saat ini teknologi sangat dibutuhkan oleh masyarakat untuk menyelesaikan permasalahan di kehidupan sehari-hari. Teknologi dapat memberikan kemudahan kepada setiap orang untuk menyelesaikan suatu masalah tertentu. Permasalahan yang diselesaikan dengan teknologi biasanya adalah permasalahan yang rumit dan membutuhkan waktu yang cukup lama.

Salah satu permasalahan yang sering dihadapi masyarakat khususnya di Institusi pendidikan adalah permasalahan penyusunan jadwal mengajar. Penyusunan jadwal mengajar biasanya membutuhkan waktu yang cukup lama karena setiap jadwal harus menyesuaikan kegiatan masing-masing pengajar agar jadwal yang dibuat dapat ditaati oleh setiap pengajar.

Setiap kelompok orang memiliki batasan (*constraint*) pembuatan jadwal yang berbeda-beda. Batasan (*constraint*) yang dimaksud adalah aturan-aturan penyusunan jadwal yang dimiliki institusi agar jadwal yang dibuat dapat dilaksanakan oleh orang-orang yang bersangkutan. Aturan-aturan tersebut sering menjadi sebab penyusunan jadwal membutuhkan waktu yang cukup lama. Penyusunan jadwal juga membutuhkan tingkat ketelitian yang tinggi agar tidak ada kesalahan sehingga dapat ditaati oleh orang-orang yang bersangkutan.

Kasus yang dijelaskan di atas sering dialami oleh bagian Administrasi penyusun jadwal mengajar Asisten di Laboratorium Universitas Islam Indonesia. Divisi Administrasi di Laboratorium Informatika khususnya Laboratorium Komputasi dan Sistem Cerdas memiliki anggota yang sangat terbatas. Banyaknya Asisten Divisi Administrasi Laboratorium tidak sebanding dengan banyaknya tugas yang diberikan. Penelitian ini diharapkan dapat menerapkan Algoritma Genetika dalam permasalahan pembuatan jadwal mengajar Asisten Laboratorium Komputasi dan Sistem Cerdas dengan hasil yang optimal sehingga dapat meringankan pekerjaan Divisi Administrasi Laboratorium.

1.2. Rumusan Masalah

Berdasarkan latar belakang yang telah dipaparkan di atas, penelitian ini memiliki rumusan masalah sebagai berikut:

1. Bagaimana cara membuat jadwal mengajar Asisten Laboratorium yang optimal dan tidak melanggar *constraint timetable* Laboratorium Komputasi dan Sistem Cerdas.
2. Bagaimana *encoding* fakta permasalahan *timetabling* menjadi kromosom pada Algoritma Genetika. Algoritma Genetika merupakan salah satu metode optimasi *metaheuristic*. *Metaheuristic* dapat di definisikan sebagai metode lanjut (*advanced*) berbasis *heuristic* untuk menyelesaikan persoalan optimasi secara efisien (Talbi, 2009).
3. Bagaimana implementasi Algoritma Genetika untuk menyelesaikan masalah penyusunan jadwal mengajar Asisten di Laboratorium Komputasi dan Sistem Cerdas Teknik Informatika Universitas Islam Indonesia.

1.3. Batasan Masalah

Penelitian dan pembuatan perangkat lunak pada tugas akhir ini memiliki batasan masalah sebagai berikut:

1. Aturan pembuatan jadwal yang digunakan adalah aturan Laboratorium Komputasi Sistem Cerdas Teknik Informatika Universitas Islam Indonesia, aturan-aturan tersebut yaitu:
 - a. Banyaknya kelompok mahasiswa yang diampu setiap asisten dalam satu kelas harus sama.
 - b. Jadwal mengajar Asisten di laboratorium tidak boleh mengganggu jadwal kuliah.
 - c. Perbandingan banyaknya mengajar antara asisten satu dengan yang lain adalah 1 : 1 atau mendekati.

- d. Perbandingan banyaknya asisten senior dan junior di dalam 1 kelas adalah 1 : 1 atau mendekati.
 - e. Seluruh Asisten harus mendapatkan giliran mengajar.
2. Tidak memperhitungkan penambahan aturan/ *constraint*.

1.4. Tujuan Penelitian

Penelitian pada tugas akhir ini dilakukan untuk menyelesaikan masalah penyusunan jadwal mengajar Asisten Laboratorium Komputasi dan Sistem Cerdas Universitas Islam Indonesia. Masalah tersebut akan diselesaikan dengan menggunakan metode Algoritma Genetika.

Algoritma Genetika yang digunakan pada penelitian ini akan disesuaikan dengan masalah terkait sehingga dapat memberikan solusi terbaik dan waktu penyusunan yang singkat. Jadwal yang dihasilkan diharapkan dapat membantu dan meringankan beban kerja Asisten divisi Administrasi Laboratorium Komputasi dan Sistem Cerdas Universitas Islam Indonesia.

1.5. Manfaat Penelitian

Penelitian ini diharapkan dapat bermanfaat untuk seluruh anggota Asisten Laboratorium khususnya Asisten Divisi Administrasi yang ditugaskan untuk menyusun jadwal mengajar Asisten. Beberapa manfaat tersebut yaitu:

1. Tugas divisi Administrasi Laboratorium dapat lebih ringan.
2. Waktu yang diperlukan untuk membuat jadwal lebih singkat sehingga Divisi Administrasi dapat menyelesaikan tugas yang lain dengan waktu yang lebih banyak.
3. Tidak ada kecemburuan tentang giliran mengajar antara Asisten satu dengan yang lain.
4. Asisten yang bukan termasuk Divisi Administrasi dapat langsung menyaksikan jadwal tersusun dengan cepat.

1.6. Metodologi

Terdapat 2 tahap yang dilakukan dalam melakukan penelitian Tugas Akhir ini, tahapan tersebut adalah:

1. Pengumpulan Data

Pengumpulan data dalam penelitian ini dilakukan dengan cara sebagai berikut:

- a. Studi pustaka, yaitu mengumpulkan data dan informasi dengan cara mencari beberapa buku dan dokumen penelitian yang berkaitan dengan kasus *timetabling*.
- b. Observasi, yaitu mengumpulkan beberapa data dan informasi dengan melakukan pengamatan langsung pada pembuatan jadwal secara manual oleh Asisten Divisi Administrasi.
- c. *Interview*, yaitu mengumpulkan beberapa data dan informasi dengan cara wawancara kepada anggota Asisten Laboratorium Komputasi dan Sistem Cerdas Divisi Administrasi yang ditugaskan mengatur jadwal mengajar Asisten.

2. Pengembangan Perangkat Lunak

Pengembangan perangkat lunak dilakukan sesuai dengan hasil pengumpulan data yang sudah dilakukan sebelumnya. Adapun tahap-tahap tersebut adalah sebagai berikut:

a. Analisis Kebutuhan

Menganalisa seluruh kebutuhan perangkat lunak sesuai dengan data yang sudah terkumpul pada tahap pengumpulan data.

b. Perancangan Perangkat Lunak

Tahapan ini berupa perancangan Algoritma Genetika yang akan digunakan. Perancangan tersebut antara lain perancangan alur program, perancangan tampilan, dan perancangan *database*.

c. Implementasi Perangkat Lunak

Tahap ini adalah tahap pembuatan perangkat lunak berdasarkan perancangan yang sudah dilakukan. Bahasa pemrograman yang digunakan adalah bahasa pemrograman JAVA.

d. Pengujian Perangkat Lunak

Tahap ini dilakukan untuk menguji sistem yang telah dibuat. Pengujian yang dilakukan adalah pengujian *bug* dan pengujian kebenaran Algoritma Genetika pada perangkat lunak.

1.7. Sistematika Penulisan

Sistematika penulisan tugas akhir ini dibuat dengan tujuan untuk memudahkan pembaca memahami laporan. Sistematika penulisan yang digunakan yaitu:

Bab I. Pendahuluan, membahas tentang latar belakang masalah dalam pembuatan jadwal mengajar Asisten Laboratorium Komputasi dan Sistem Cerdas.

Bab II. Landasan Teori, membahas landasan teori tentang Algoritma Genetika beserta aturan-aturan yang ditetapkan oleh pihak Laboratorium untuk pembuatan jadwal.

Bab III. Analisis Kebutuhan dan Perancangan, membahas tentang hasil analisa kebutuhan perangkat lunak berdasarkan data dan informasi yang sudah didapat serta membahas perancangan perangkat lunak yang akan dibuat.

Bab IV. Implementasi dan Pengujian, membahas tentang implementasi rancangan perangkat lunak dan menyampaikan hasil pengujian perangkat lunak.

Bab V. Kesimpulan dan Saran, membahas kesimpulan berupa rangkuman dari hasil dan pembahasan pada bab sebelumnya disertai dengan saran yang perlu disampaikan untuk perbaikan dan pengembangan berikutnya.

BAB II

LANDASAN TEORI

2.1. Timetable

Timetable adalah jadwal daftar waktu beberapa acara yang tidak memiliki urutan. Contohnya adalah waktu kedatangan dan keberangkatan transportasi di stasiun. *Timetable* memiliki pengertian yang berbeda dengan *schedule*. *Schedule* adalah rancangan daftar waktu yang memiliki urutan tertentu “*a plan of procedure*” (Ex: *project scheduling*) (Miller, et. al., 1974). Jadwal yang dibahas pada penelitian ini adalah *timetable*.

Saat ini sudah terdapat beberapa penelitian yang dilakukan untuk menyelesaikan masalah *timetable*, penelitian tersebut yaitu:

1. Penelitian dari Universitas Tarumanegara “Implementasi Vertex Graph Colouring, Particle Swarm Optimization, dan Constraint Based Reasoning untuk University Timetabling Problem” (Hiryanto dan Josselyn, 2013). Penelitian ini membahas tentang permasalahan *timetable* di Universitas Tarumanegara yang diselesaikan dengan aplikasi berbasis Web. Masalah *timetable* pada penelitian ini diselesaikan menggunakan 3 metode yaitu Vertex Graph Colouring, Particle Swarm Optimization, dan Constraint Based Reasoning.
2. Penelitian dari Universitas Indonesia yang berjudul “Implementasi Algoritma Integer Linear Programming Untuk Sistem Informasi Penjadwalan Ruangan di Fakultas Ilmu Komputer Universitas Indonesia” (Hutomo, et. al., 2011). Penelitian ini membahas tentang permasalahan *timetable* di Fakultas Ilmu Komputer Universitas Indonesia. Algoritma yang digunakan adalah Algoritma Integer Linear Programming. Penelitian ini juga membandingkan hasil pembuatan *timetable* menggunakan Algoritma Integer Linear Programming dan dengan 2 algoritma lainnya. 3 Algoritma tersebut adalah Tabu Search,

Simulated Annealing, dan Graph Coloring. Hasil perbandingan tersebut menunjukkan bahwa Algoritma Integer Linear Programming merupakan pendekatan terbaik untuk memodelkan masalah *timetabling* dibandingkan dengan Tabu Search, Simulated Annealing, dan Graph Coloring.

3. Penelitian dari *Department of Computer Science Konstanz German* yang berjudul “*A Genetic Algorithm Solving a Weekly course Time Timetabling Problem*” (Erben dan Keppler, 1995). Penelitian ini membahas tentang *University Timetabling Problem* (UTP) di Universitas Konstanz German. Metode yang digunakan untuk menyelesaikan UTP pada penelitian ini adalah Algoritma Genetika. Metode dengan menggunakan Algoritma tersebut diimplementasikan ke dalam aplikasi berbasis Desktop. Hasil yang ditunjukkan pada penelitian ini cukup baik walaupun belum menggunakan data asli dari Universitas.

Penelitian-penelitian tersebut memiliki beberapa perbedaan dalam menyelesaikan masalah *timetabling*. Perbedaan 3 penelitian tersebut dapat dilihat pada **Tabel 2.1**.

Tabel 2.1. Perbedaan Penelitian Sebelumnya

Peneliti	(Hiryanto dan Josselyn, 2013)	(Hutomo, et. al., 2011)	(Erben dan Keppler, 1995)
Banyak Metode	3	1	1
Metode	<i>Vertex Graph Colouring</i>	<i>Integer Linier Programming</i>	<i>Genetic Algorithm</i>
	<i>Particle Swarm Optimization</i>		
	<i>Constraint Based Reasoning</i>		
Jenis Metode	<i>Metaheuristic</i>	Kovensional	<i>Metaheuristic</i>
Kasus	Dinamis	Statis	Dinamis

Penelitian di atas sama-sama menjelaskan masalah tentang lamanya waktu yang dibutuhkan dalam pembuatan *timetable*. Permasalahan yang sering terjadi tersebut perlu diselesaikan dengan melakukan penelitian dan pembuatan perangkat lunak yang tepat agar waktu pembuatan *timetable* dapat lebih singkat.

Kasus penelitian ini memiliki kemiripan dengan kasus pada penelitian yang dibahas pada penelitian nomer 3 oleh *Department of Computer Science Konstanz German*. Penelitian tersebut membuktikan bahwa kasus *timetabling* dengan kasus serupa dapat dilakukan menggunakan *Genetic Algorithm* dengan hasil yang cukup optimal.

Penelitian ini dan penelitian yang dilakukan pada penelitian nomer 3 juga memiliki perbedaan. Perbedaan keduanya yaitu tentang *constraint* permasalahan *timetable* dan representasi kromosom pada Algoritma Genetika.

2.2. Algoritma Genetika

Algoritma genetika merupakan salah satu metode optimasi *metaheuristic*. *Metaheuristic* dapat di definisikan sebagai metode lanjut (*advanced*) berbasis *heuristic* untuk menyelesaikan persoalan optimasi secara efisien (Talbi, 2009).

Blum dan Roli (2003) menyatakan bahwa optimasi *Metaheuristic* memiliki beberapa karakteristik dasar. Karakteristik dasar tersebut yaitu:

1. *Metaheuristic* adalah strategi yang memandu proses pencarian.
2. Tujuan dari *metaheuristic* adalah untuk menjelajahi ruang pencarian secara *efficient* untuk menemukan solusi optimal.
3. Teknik *metaheuristic* berkisar dari prosedur pencarian *local* yang sederhana sampai proses pembelajaran yang kompleks.
4. *Metaheuristic* adalah metode pendekatan dan biasanya *non-deterministic/ stochastic*.
5. *Metaheuristic* dapat terdiri dari penggabungan beberapa mekanisme supaya proses pencarian tidak terjebak dalam daerah terbatas di ruang pencarian.
6. Konsep dasar dari *metaheuristic* memungkinkan deskripsi secara abstrak

7. *Metaheuristic* bersifat *general*/ umum sehingga dapat diterapkan dalam berbagai macam persoalan
8. *Metaheuristic* dapat menggunakan *domain* pengetahuan khusus dalam bentuk *heuristic* yang dikendalikan dengan strategi tingkat lanjut.
9. *Metaheuristic* dapat menggunakan pengalaman yang didapat selama proses pencarian untuk menuntun proses pencarian.

Pencarian *heuristic* memiliki perbedaan dengan pencarian *metaheuristic* pencarian *heuristic* bersifat *problem dependent* (Bergantung pada permasalahan tertentu). Contohnya adalah metode *Nearest Neighborhood* Metode hanya bisa dipakai pada permasalahan yang mengenal konsep *neighborhood*/ tetangga, misalnya pada *Traveling Salesman Problem* maupun *Vehicle Routing Problems*. Pencarian *metaheuristic* bersifat *problem independent* (tidak tergantung pada jenis permasalahan). Contohnya metode *Genetic Algorithm* (GA), *Particle Swarm Optimization* (PSO), dan *Ant Colony Optimization* (ACO) (Hindriyanto, 2012).

Algoritma Genetika (*Genetic Algorithm*) adalah suatu algoritma teknik optimasi dan pencarian berdasarkan prinsip genetika dan seleksi alam (Randy dan Sue, 2004). Algoritma Genetika pertama kali diperkenalkan oleh John Holland di Michigan University *United State of America*. John Holland menyatakan bahwa setiap permasalahan yang berbentuk adaptasi dapat diformulasikan menggunakan *Genetic Terminology*.

Gen dan Cheng (1997) menyatakan bahwa Algoritma Genetika memiliki 3 keunggulan utama. Keunggulan tersebut yaitu:

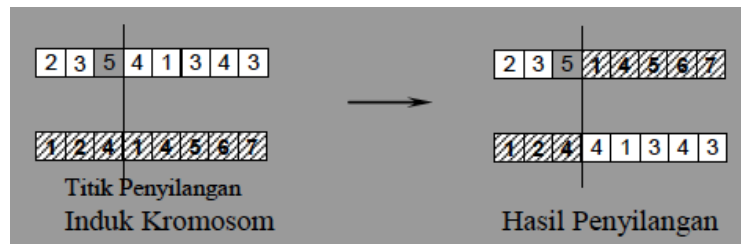
1. Algoritma Genetika tidak membutuhkan banyak perhitungan matematika untuk memecahkan masalah optimasi.
2. Memiliki operator evaluasi *ergodicity* (atribut sistem *stochastic*) yang menjadikan Algoritma Genetika sangat efektif dalam melakukan pencarian global (dalam probabilitas).
3. Algoritma Genetika memberikan fleksibilitas yang baik untuk melakukan integrasi dengan metode *heuristic* agar lebih efisien jika diimplementasikan pada detail masalah yang lebih spesifik.

Terdapat beberapa istilah yang digunakan dalam Algoritma Genetika, istilah-istilah tersebut yaitu:

1. *Chromosome* (kromosom): solusi (dalam bentuk *code/ coding*).
2. *Genes* (gen): bagian dari solusi.
3. *Locus*: posisi suatu gen.
4. *Alleles*: nilai suatu gen.
5. *Phenotype*: solusi yang sudah diterjemahkan.
6. *Genotype*: solusi dalam bentuk kode

Algoritma Genetika memiliki beberapa komponen/ operator utama yang digunakan. Operator tersebut yaitu:

1. Representasi Kromosom
Mengodekan suatu masalah menjadi sekumpulan gen (kromosom) untuk diubah menjadi solusi.
2. Inisialisasi
Pembuatan populasi awal yang terdiri dari kromosom dengan nilai gen acak sebagai solusi awal yang akan diperbaiki pada iterasi berikutnya.
3. Evaluasi
Perhitungan nilai *fitness* yang didapat dari nilai objektif berdasarkan *constraint* yang telah ditetapkan.
4. Seleksi
Pembuatan populasi baru dengan melakukan seleksi berdasarkan perhitungan nilai probabilitas menggunakan nilai *fitness* setiap kromosom.
5. *Crossover* (Penyilangan)
Penyilangan adalah pertukaran satu atau lebih gen antara suatu kromosom dengan kromosom yang lain sehingga menghasilkan dua kromosom baru. Contoh proses *crossover* dapat dilihat pada **Gambar 2.1**.



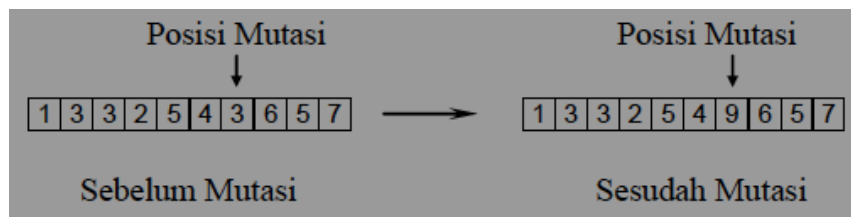
Gambar 2.1. Penyilangan Satu Titik

(Zukhri, 2014)

6. Mutasi

Perubahan nilai gen secara acak berdasarkan nilai probabilitas mutasi yang sudah ditetapkan sebelumnya. Contoh mutasi dapat dilihat pada

Gambar 2.2.



Gambar 2.2. Gambar Contoh Mutasi

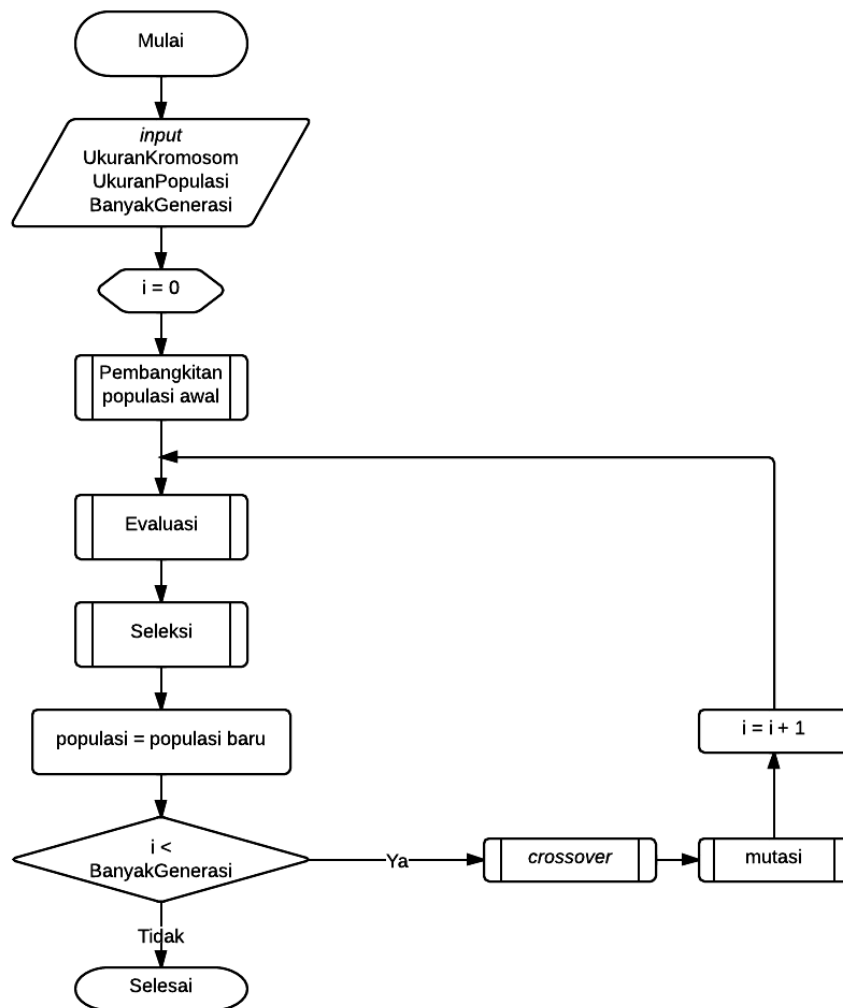
(Zukhri, 2014)

Algoritma Genetika juga merupakan salah satu algoritma *soft computing* sehingga proses maupun urutan pengerjaan tidak harus selalu sama. Implementasi Algoritma Genetika akan berbeda jika kasus yang diselesaikan juga berbeda. Kusumadewi dan Hari (2005) menyatakan bahwa alur pengerjaan Algoritma Genetika adalah sebagai berikut:

1. Generasi = 0
2. Inisialisasi Populasi Awal, $P(\text{generasi})$, secara acak
3. Evaluasi nilai fitness pada setiap individu dalam $P(\text{generasi})$
4. Kerjakan langkah-langkah berikut hingga generasi mencapai maksimum generasi:
 - a. $\text{Generasi} = \text{generasi} + 1$
 - b. Seleksi populasi untuk mendapatkan kandidat induk, $P'(\text{generasi})$
 - c. Lakukan crossover pada $P'(\text{generasi})$

- d. Lakukan mutasi pada P' (generasi) // optional
- e. Lakukan evaluasi fitness setiap individu pada P' (generasi)
- f. Bentuk populasi baru $P(\text{generasi}) = \{P(\text{generasi}-1) \text{ yang bertahan, } P' \text{Generasi}\}$

Flowchart dari urutan pengerjaan Algoritma Genetika di atas dapat dilihat pada **Gambar 2.3**. Gambar tersebut akan menjelaskan alur Algoritma Genetika secara umum beserta operator – operator yang harus disediakan.



Gambar 2.3. *Flowchart* Algoritma Genetika

BAB III

ANALISIS KEBUTUHAN DAN PERANCANGAN PERANGKAT LUNAK

3.1. Analisis Kebutuhan Perangkat Lunak

Analisis kebutuhan perangkat lunak dilakukan untuk mengetahui beberapa kebutuhan yang harus dipersiapkan sebelum perangkat lunak dibuat. Tahap analisis kebutuhan ini dilakukan berdasarkan beberapa data dan informasi yang telah didapat saat melakukan pengumpulan data dengan cara studi pustaka, observasi, dan *interview*.

3.1.1. Kebutuhan Masukan

Berdasarkan data hasil *interview* dengan Asisten Laboratorium Komputasi dan Sistem Cerdas Divisi Administrasi, pembuatan perangkat lunak *timetable* ini memiliki beberapa kebutuhan masukan. Kebutuhan tersebut yaitu:

1. Data *Constraint/ Aturan Timetable*

Data ini berisi tentang aturan-aturan pembuatan *timetable* Laboratorium Komputasi dan Sistem Cerdas yang digunakan untuk perhitungan nilai objektif dari kromosom yang dibangkitkan. Data *constraint* yang dibutuhkan dibagi menjadi 2 yaitu *hard-constraint* dan *soft-constraint*.

2. Data Asisten

Data Asisten adalah data Asisten Laboratorium Komputasi Sistem Cerdas pada tahun ajaran yang aktif.

3. Data Kelas Praktikum

Data ini adalah data kelas yang disediakan untuk praktikum di Laboratorium Komputasi dan Sistem Cerdas pada tahun ajaran yang aktif.

4. Data Kendala Kelas Asisten

Data ini adalah data yang kelas praktikum yang tidak dapat diikuti oleh setiap asisten karena beberapa kendala yang dimiliki oleh asisten bersangkutan. Kendala tersebut bisa berupa jadwal kelas praktikum

yang waktunya bersamaan dengan jadwal kuliah atau terdapat acara lain di luar perkuliahan.

3.1.2. Kebutuhan Keluaran

Perangkat lunak yang dibuat membutuhkan beberapa keluaran yang dibutuhkan oleh *user*. Kebutuhan tersebut yaitu:

1. Hasil *Timetable* Asisten
Keluaran ini berupa *timetable* mengajar asisten yang dihasilkan oleh perangkat lunak.
2. Laporan Pelanggaran *Constraint Timetable*
Isi laporan ini adalah beberapa pelanggaran yang masih terjadi pada *timetable* yang terbuat. Pelanggaran tersebut berupa pelanggaran *Hard-Constraint* dan *Soft-Constraint*. Berikut ini adalah laporan yang disebutkan:
 - a. Berapa banyak asisten yang di dalam 1 kelas namanya tercantum lebih dari 1 kali.
 - b. Berapa banyak jadwal mengajar asisten yang waktunya bersamaan dengan kuliah atau acara lain.
 - c. Banyaknya mengajar setiap Asisten dalam 1 minggu.
 - d. Perbandingan banyak Asisten Senior dan Junior dalam 1 kelas.

3.1.3. Kebutuhan Proses

Berdasarkan Studi Pustaka yang dilakukan, perangkat lunak *timetabling* dengan menggunakan Algoritma Genetika yang dibuat membutuhkan adanya beberapa proses. Proses tersebut yaitu:

1. *Encoding* kasus menjadi kromosom.
2. *Genetic Algorithm Operator*:
 - a. *Selection (Roulette Wheel Selection)*
 - b. *Crossover*
 - c. *Mutation*
3. *Decoding* kromosom terbaik menjadi solusi.

3.2. Perancangan Perangkat Lunak

Perangkat lunak pada tugas akhir ini dirancang berdasarkan hasil observasi yang telah dilakukan. Hasil observasi tersebut adalah suatu kesimpulan bahwa proses pembuatan *timetable* membutuhkan waktu yang cukup lama karena adanya *constraint* yang harus ditaati dan banyaknya data Asisten dan Kelas pada jadwal yang akan dibuat. Perancangan ini bertujuan untuk mempermudah peneliti dalam membangun perangkat lunak dan membuat perangkat lunak tersusun dengan baik sehingga dapat lebih mudah untuk dikembangkan.

3.2.1. Hubungan antar Class

Perangkat lunak yang dibangun menggunakan bahasa pemrograman JAVA, sehingga program akan diselesaikan oleh beberapa objek dari hasil instansiasi kelas-kelas yang dibuat. Kelas-kelas tersebut ada yang berhubungan dengan pengolahan database dan berhubungan dengan Algoritma Genetika. Kelas-kelas yang berhubungan dengan Database adalah sebagai berikut:

- a. Koneksi
Kelas ini digunakan untuk menghubungkan sistem yang dibuat dengan Database.
- b. Database
Kelas database merupakan induk kelas dari kelas DBkelas dan DBAsisten. Kelas ini digunakan sistem untuk berkomunikasi dengan database. Komunikasi tersebut berupa input data dan penghapusan data.
- c. DBKelas
Kelas DBKelas merupakan kelas yang digunakan sistem untuk berkomunikasi dengan database yang berhubungan dengan data kelas praktikum Laboratorium.
- d. DBAsisten
Kelas DBAsisten merupakan kelas yang digunakan sistem untuk berkomunikasi dengan database yang berhubungan dengan data Asisten Laboratorium.

Kelas-kelas di atas digunakan untuk menyelesaikan masalah dalam Koneksi database. Kelompok kelas di atas juga digunakan sebagai pembuatan objek yang berfungsi untuk berkomunikasi dengan database.

Pada kelas-kelas yang disebutkan di bawah ini berfungsi untuk membuat objek yang akan digunakan dalam menyelesaikan masalah-masalah yang berhubungan dengan Algoritma Genetika:

a. GenAsisten

Kelas GenAsisten merupakan kelas yang digunakan untuk mendapatkan data asisten dari database dan mengubah data tersebut menjadi Gen yang berupa bilangan-bilangan bulat.

b. GenKelas

Kelas GenKelas adalah kelas yang digunakan untuk mendapatkan data kelas-kelas yang tersedia dan kebutuhan asisten di masing-masing kelas dari database.

c. Populasi

Kelas populasi merupakan kelas yang digunakan untuk membentuk populasi dari beberapa kromosom. Kelas ini juga berisi operator-operator Algoritma Genetika.

d. GeneticAlgorithm

Kelas ini merupakan kelas yang digunakan untuk menjalankan operator-operator Algoritma Genetika dalam pembuatan jadwal.

e. Decoding

Kelas ini merupakan kelas yang digunakan untuk menerjemahkan kromosom menjadi jadwal yang dapat difahami oleh user.

f. Fitness

Kelas Fitness adalah kelas yang digunakan untuk menghitung banyaknya pelanggaran *constraint* dan menghitung nilai *fitness* pada setiap kromosom.

g. Graph

Kelas ini digunakan untuk pembuatan graph yang menunjukkan perkembangan nilai *fitness* pada setiap generasi.

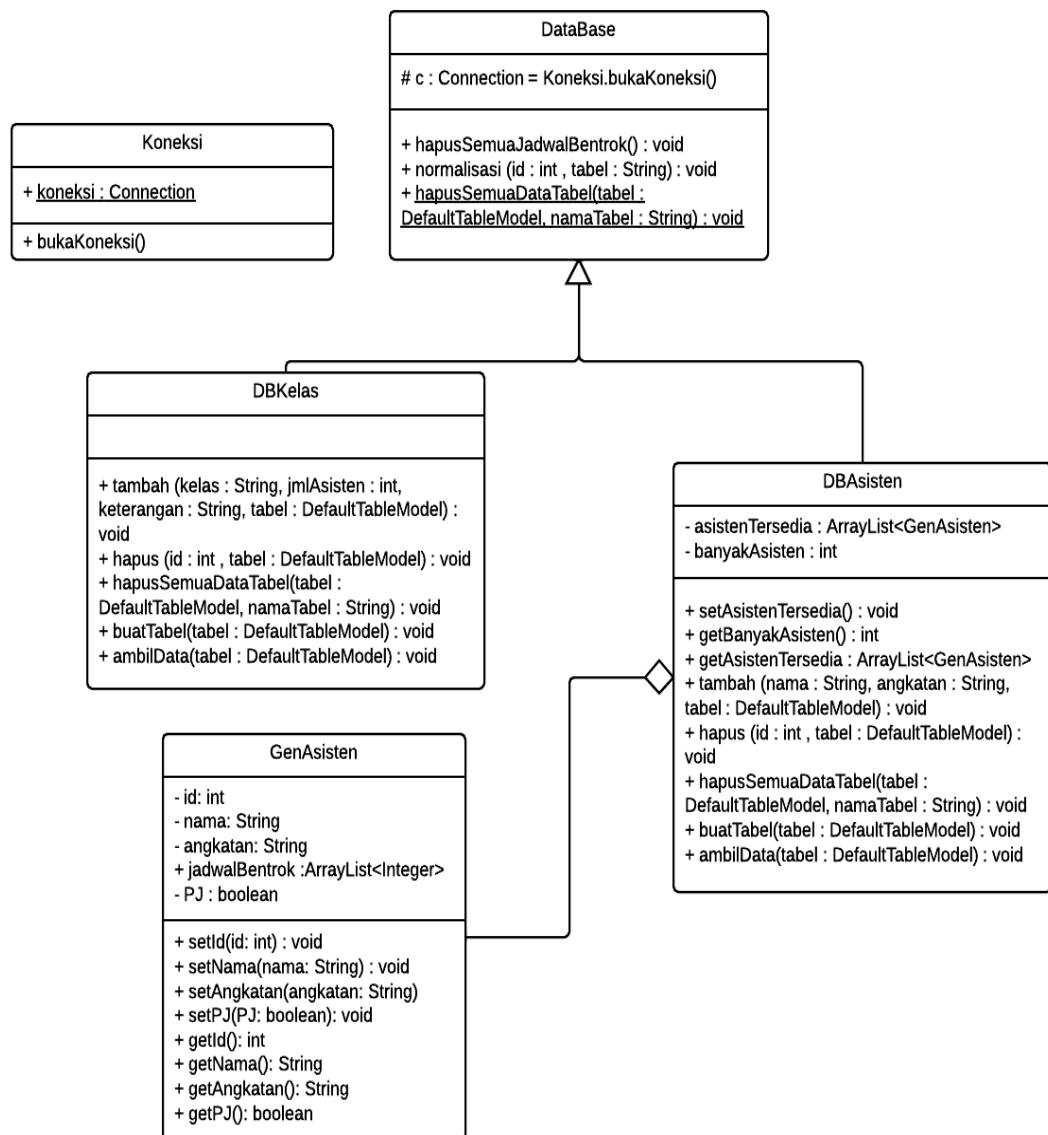
h. Kromosom

Kelas kromosom merupakan kelas yang digunakan untuk membuat kromosom yang digunakan dalam proses pembuatan jadwal dengan Algoritma Genetika.

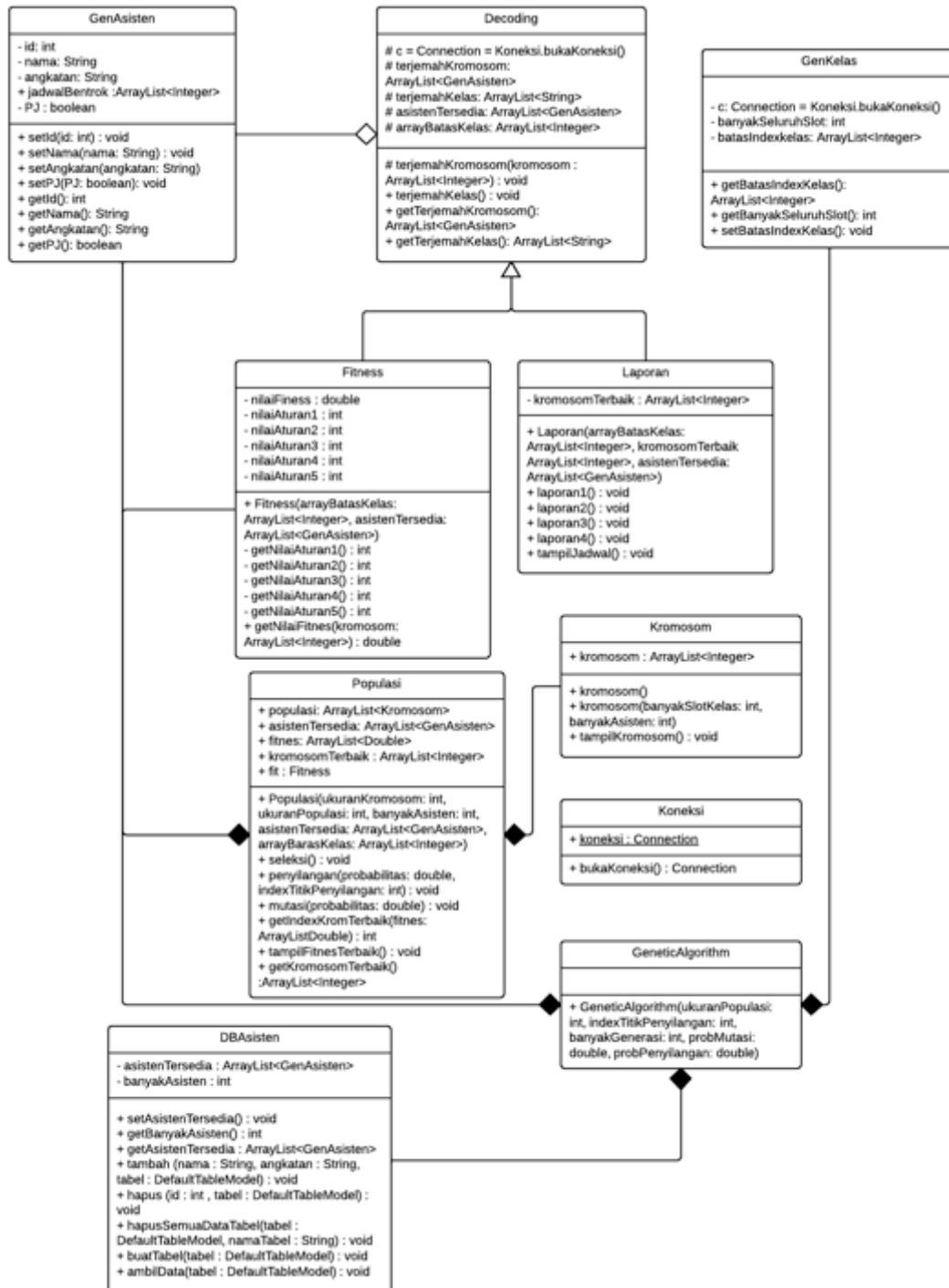
i. Laporan

Kelas ini adalah kelas yang berhubungan dengan laporan pelanggaran *constraint* yang terjadi pada jadwal yang telah dibuat.

Perancangan hubungan antar kelas perangkat lunak dapat dilihat pada UML diagram pada Error! Reference source not found. dan Error! Reference source not found..



Gambar 3.1. Hubungan Antar Kelas A



Gambar 3.2. Hubungan Antar Kelas B

3.2.2. Representasi Kromosom

Representasi kromosom yang diterapkan pada kasus ini adalah representasi kromosom dengan menggunakan bilangan bulat nol dan positif. Satu gen pada kromosom diwakili satu bilangan bulat yang merepresentasikan seorang asisten. Gen memiliki nilai bilangan bulat maksimal yang disesuaikan dengan total banyak asisten (Ba). Rumus nilai maksimal gen dapat dilihat pada persamaan 3.1.

$$(Z_{max} = Ba - 1) \dots\dots\dots (3.1)$$

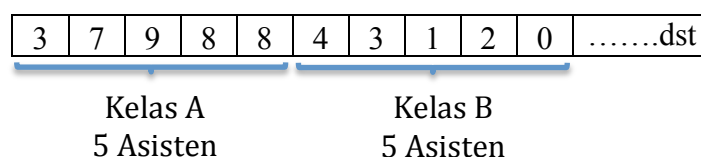
Posisi gen (*Locus*) merepresentasikan kelas praktikum yang disesuaikan dengan banyaknya kebutuhan asisten pada kelas tersebut (Ka_i). Representasi tersebut menghasilkan rumus *chromosome size* (Cs) pada persamaan 3.2. Salah satu contoh representasi kromosom dapat dilihat pada **Gambar 3.3**.

Tabel 3.1. Tabel Contoh Daftar Kelas dan Asisten

ID Kelas	Nama Kelas	Kebutuhan Asisten
0	A	5
1	B	5
2	C	6
3	D	4
4	E	5
5	F	6

ID Asisten	Asisten
0	Ivan
1	Yopi
2	Deni
3	Uzi
4	Lina
5	Dadang
6	Aul
7	Deka
8	Ito

$$Cs = \sum_{i=1}^{BykKelas} Ka_i \dots\dots\dots (3.2)$$



Gambar 3.3. Contoh Kromosom

3.2.3. Perhitungan Nilai *Fitness*

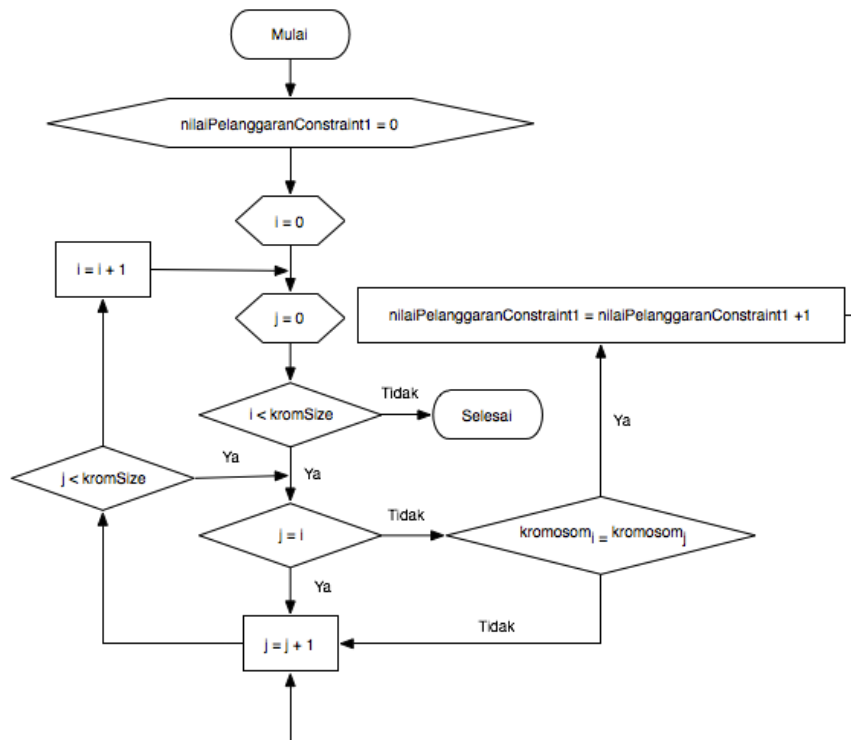
Perhitungan nilai *fitness* dipengaruhi nilai objektif yang didapat dari perhitungan pelanggaran *constraint* kromosom. Kasus *timetable* ini menggunakan 5 *constraint* yang diklasifikasikan menjadi 2 bagian yaitu *hard-constraint* dan *soft-constraint*. *Constraint* yang digunakan dapat dilihat pada **Tabel 3.2**.

Tabel 3.2. Tabel Constraint

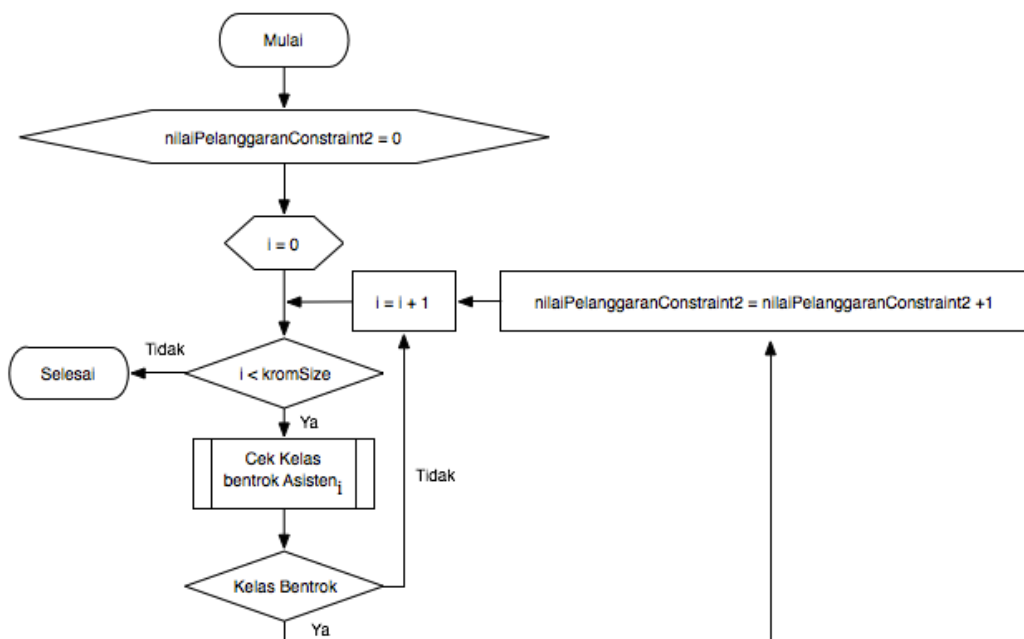
<i>Hard-constraint</i>	
1	Banyaknya kelompok mahasiswa yang diampu setiap asisten dalam satu kelas harus sama.
2	Jadwal mengajar asisten tidak boleh mengganggu jadwal kuliah.
3	Semua Asisten wajib mendapat jadwal mengajar.
<i>Soft-constraint</i>	
4	Perbandingan banyaknya mengajar setiap Asisten 1 : 1 atau mendekati.
5	Perbandingan banyaknya Asisten senior dan junior dalam 1 kelas adalah 1 : 1 atau mendekati.

Algoritma untuk mendapatkan nilai pelanggaran masing-masing *constraint* (M_i) dapat dilihat pada **Gambar 3.4**, **Gambar 3.5**, **Gambar 3.6**, **Gambar 3.7**, dan **Gambar 3.8**. Nilai pelanggaran adalah banyaknya pelanggaran *constraint* pada jadwal yang terbuat. Gambar tersebut menjelaskan penambahan nilai pelanggaran *constraint* yang terjadi pada setiap solusi yang dibuat.

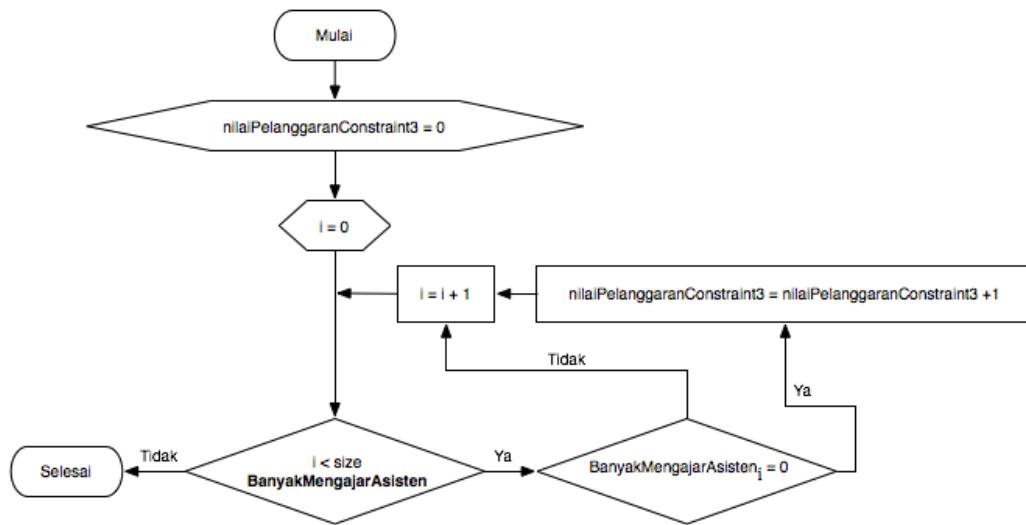
Nilai pelanggaran masing-masing *constraint* nantinya akan dikalikan dengan nilai *penalty* sesuai prioritas *constraint* yang diberikan. Pemberian nilai *penalty* ini akan di bahas pada pembahasan selanjutnya.



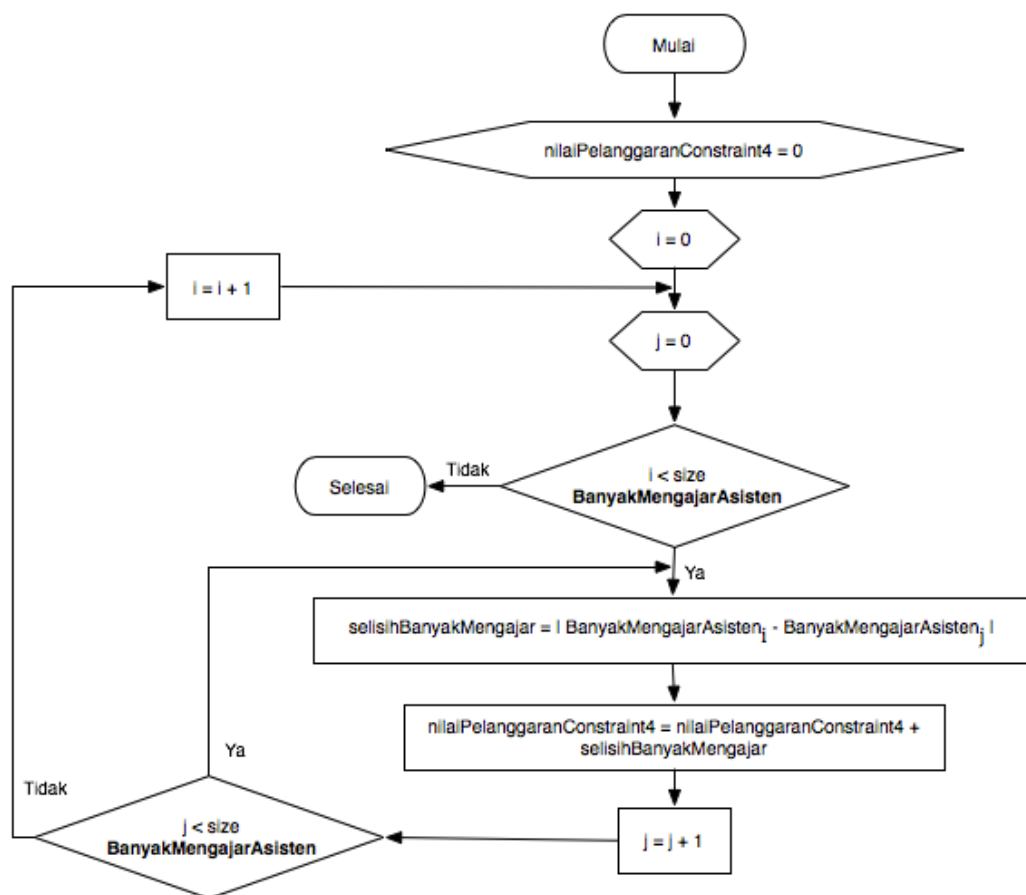
Gambar 3.4. Flowchart Perhitungan Nilai Pelanggaran Constraint 1 (M_1)



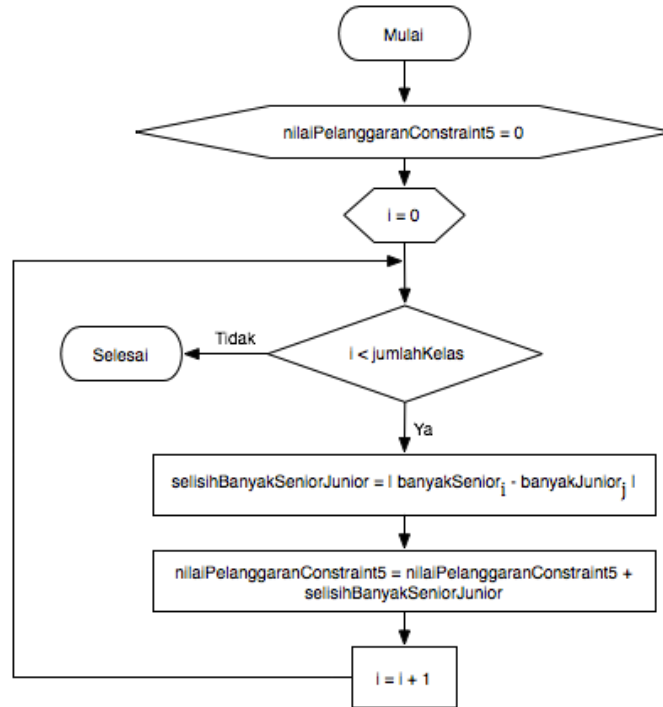
Gambar 3.5. Flowchart Perhitungan Nilai Pelanggaran Constraint 2 (M_2)



Gambar 3.6. Flowchart Perhitungan Nilai Pelanggaran Constraint 3 (M_3)



Gambar 3.7. Flowchart Perhitungan Nilai Pelanggaran Constraint 4 (M_4)



Gambar 3.8. Flowchart Perhitungan Nilai Pelanggaran Constraint 5 (M_5)

Nilai *constraint* (C_x) adalah nilai pelanggaran setiap *constraint* (M_i) pada kromosom (v_k) dikalikan nilai variabel *penalty* (pC_x). Persamaan perhitungan *constraint* (C_x) setiap kromosom dapat dilihat pada persamaan 3.3 di bawah ini.

$$C_x(v_k) = \sum_{i=1}^5 M(v_k)_i \times pC_x \quad \dots\dots\dots (3.3)$$

Nilai variabel *penalty* (pC_x) adalah nilai yang digunakan sebagai parameter prioritas *constraint* yang harus dipenuhi. Nilai pC_x semakin tinggi jika prioritas *constraint* semakin tinggi. Nilai pC_x dapat diisi sesuai dengan keinginan *user* namun pC_x sudah diberikan nilai *default* untuk memberikan gambaran perbandingan prioritas masing-masing *constraint* kepada *user*. Nilai *default* tersebut adalah sebagai berikut:

1. $pC_1 = 10$
2. $pC_2 = 7$
3. $pC_3 = 5$
4. $pC_4 = 0,1$
5. $pC_5 = 0,1$

$$Obj(v_k) = C_1(v_k) + C_2(v_k) + C_3(v_k) + C_4(v_k) + C_5(v_k) \dots\dots\dots (3.4)$$

Nilai-nilai *constraint* setiap kromosom digunakan untuk menghitung nilai objektif (*Obj*) dengan menggunakan persamaan 3.4. Nilai objektif akan mempengaruhi nilai *fitness* yang digunakan pada seleksi kromosom untuk generasi berikutnya. Rumus persamaan nilai *fitness* (*f*) dapat dilihat pada persamaan 3.5.

$$f(v_k) = \frac{1}{Obj(v_k)+1} \dots\dots\dots (3.5)$$

3.2.4. Seleksi

Metode seleksi yang digunakan adalah metode *Roulette Wheel Selection* dengan bilangan desimal dan elitisme. Elitisme adalah mengikut sertakan kromosom dengan nilai *fitness* terbaik ke dalam generasi berikutnya. *Roulette Wheel Selection* dilakukan dengan menggunakan nilai probabilitas masing-masing kromosom. Nilai probabilitas tersebut mewakili besarnya kemungkinan kromosom untuk bertahan di generasi berikutnya. Persamaan 3.6, 3.7, dan 3.8 adalah formulasi yang digunakan untuk menghitung nilai total *fitness* (*F*), probabilitas kromosom (*p_k*), dan probabilitas kumulatif kromosom (*q_k*):

$$F = \sum_{k=1}^{popSize} f(v_k) \dots\dots\dots (3.6)$$

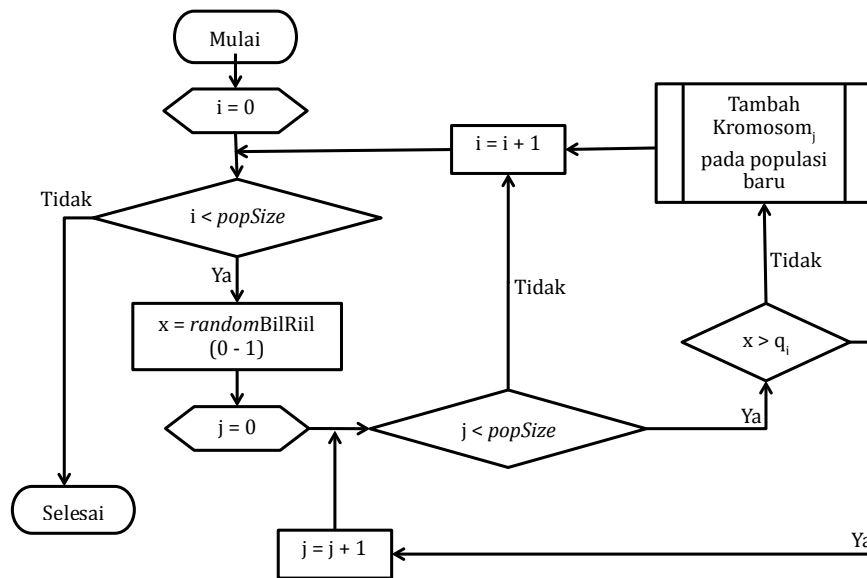
$$p_k = \frac{f(v_k)}{F} \dots\dots\dots (3.7)$$

$$q_k = \sum_{j=1}^k p_j \dots\dots\dots (3.8)$$

Seleksi dilakukan dengan cara mengacak bilangan riil dari 0 sampai 1. Setiap bilangan yang dibangkitkan akan menunjuk indeks kromosom terpilih berdasarkan nilai probabilitas kumulatif kromosom. Kromosom-kromosom yang terpilih dijadikan populasi baru untuk diproses pada generasi berikutnya.

Tahapan seleksi memberikan kesempatan lebih besar pada kromosom yang memiliki nilai probabilitas tinggi untuk bertahan pada generasi berikutnya.

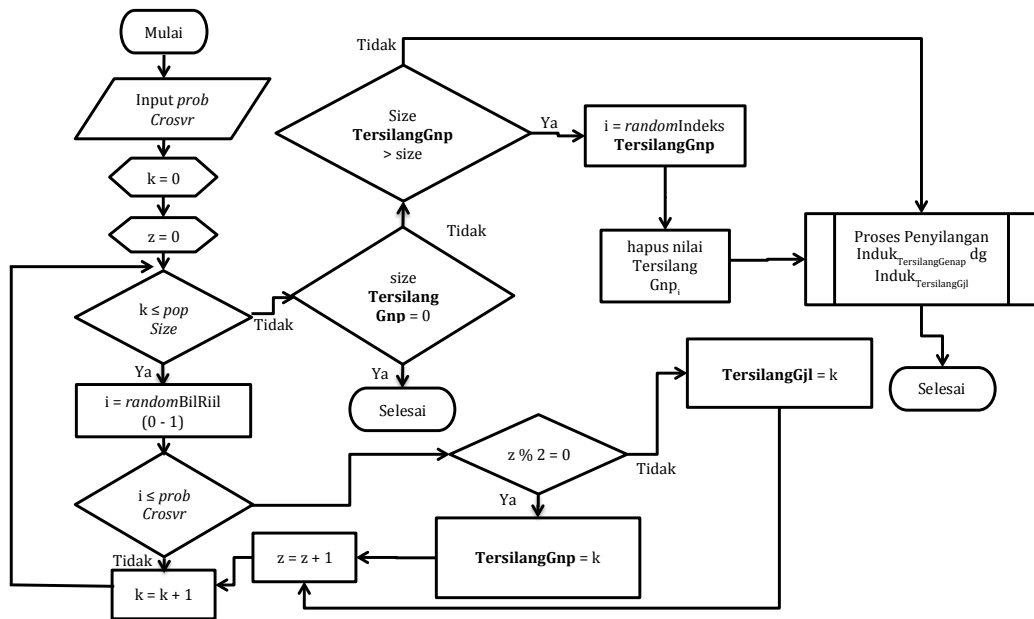
Kromosom dengan probabilitas tinggi tersebut menyebabkan kromosom dengan nilai probabilitas rendah hilang secara perlahan karena memiliki kemungkinan terseleksi yang rendah.



Gambar 3.9. *Flowchart Seleksi Kromosom*

3.2.5. Penyilangan/ Kawin Silang (*Crossover*)

Penyilangan yang digunakan adalah penyilangan 1 titik. Penyilangan 1 titik sudah dijelaskan pada bab landasan teori **Gambar 2.1**. Tujuan dilakukan *crossover* adalah untuk menghasilkan individu/ kromosom baru yang tetap mewarisi sifat 2 induknya. Kromosom baru tersebut diharapkan menjadi individu-individu yang memiliki nilai *fitness* lebih baik dari 2 induknya. *Flowchart crossover* dapat dilihat pada **Gambar 3.10**.

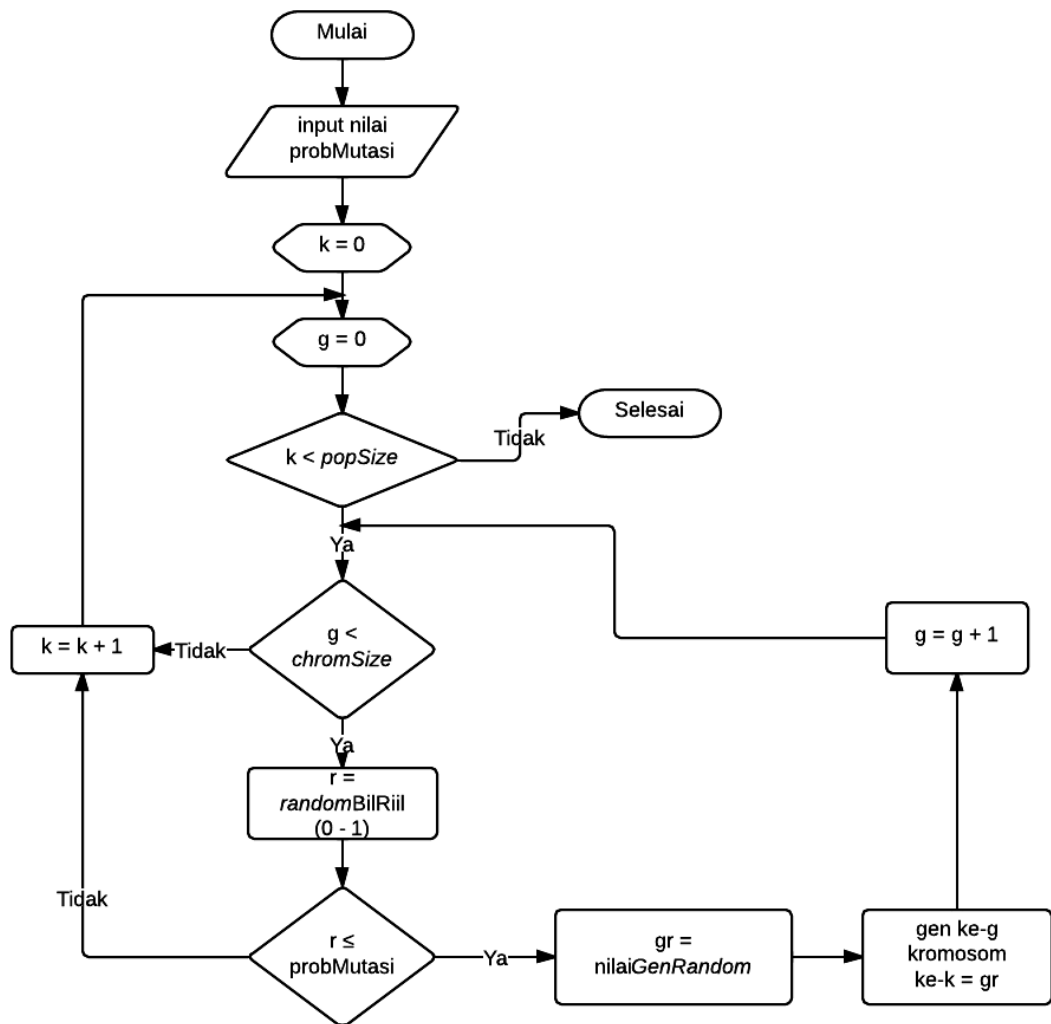


Gambar 3.10.Flowchart Crossover (Penyilangan)

3.2.6. Mutasi

Mutasi bertujuan untuk merubah sifat individu/ kromosom pada suatu generasi. Mutasi dilakukan dengan cara merubah nilai gen dengan bilangan *random*. Mutasi gen pada kromosom diharapkan dapat memberikan perubahan sifat pada kromosom tersebut sehingga memiliki nilai *fitness* yang lebih baik dari sebelumnya.

Perubahan kromosom yang terjadi karena mutasi dapat memberikan pengaruh pada kinerja Algoritma Genetika. Banyak generasi yang dibutuhkan untuk mendapatkan nilai *fitness* terbaik dapat lebih sedikit jika nilai probabilitas yang diberikan tepat, namun jika pemberian nilai probabilitas mutasi tidak tepat maka peningkatan nilai *fitness* akan menjadi lambat bahkan memiliki kemungkinan nilai *fitness* tidak dapat berkembang. Flowchart mutasi gen dapat dilihat pada **Gambar 3.11**.



Gambar 3.11.Flowchart Mutasi

3.2.7. Perancangan Interface

Rancangan *interface* perangkat lunak *timetabling* dibuat *user-friendly* agar *user* dapat menggunakan perangkat lunak dengan mudah. *Interface* yang *user-friendly* juga dapat mempermudah *user* untuk mendapatkan informasi pada perangkat lunak. Rancangan *interface timetabling* adalah sebagai berikut:

1. Tampilan Menu Asisten

Tampilan Menu Asisten merupakan tampilan yang digunakan untuk mengatur daftar Asisten yang aktif. Menu ini memberikan fasilitas *input* dan *delete* data Asisten di dalam *database*. Rancangan tampilan menu Asisten dapat dilihat pada **Gambar 3.12**.

Nama:
 Angkatan:

Tabel Daftar Asisten

Gambar 3.12.Tampilan Menu Asisten

2. Tampilan Menu Kelas

Tampilan menu kelas merupakan tampilan menu untuk menampilkan daftar kelas praktikum yang disediakan oleh Laboratorium. Menu ini juga memberikan fasilitas *input dan delete* data kelas di dalam *database*. Rancangan menu kelas dapat dilihat pada **Gambar 3.13**.

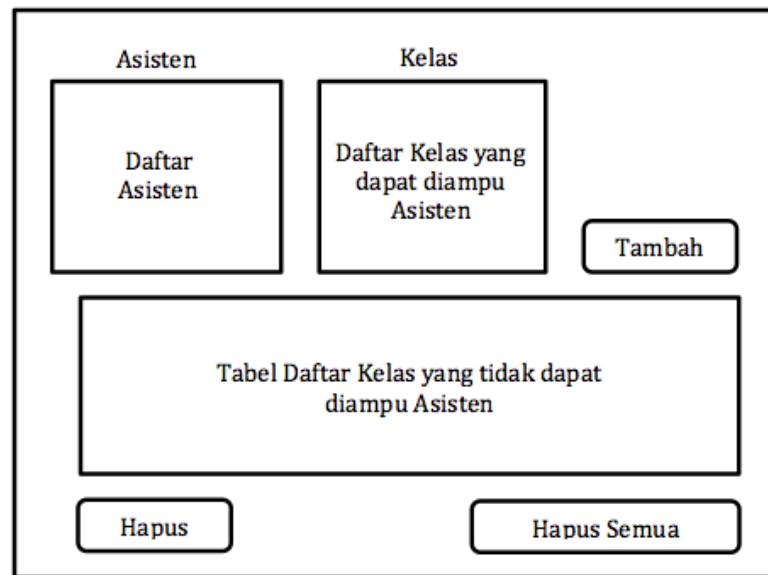
Nama Kelas:
 Jumlah Asisten:
 Keterangan:

Tabel Daftar Kelas

Gambar 3.13.Tampilan Menu Kelas

3. Tampilan Menu Kendala Kelas Asisten

Tampilan menu ini adalah tampilan kelas-kelas praktikum yang tidak bisa diambil oleh Asisten bersangkutan. Menu ini memberikan fasilitas *input* dan *delete* data kelas yang tidak bisa diampu oleh Asisten. Rancangan menu ini dapat dilihat pada **Gambar 3.14**.



Gambar 3.14. Tampilan Menu Kelas yang Tidak Bisa Diampu Asisten

3.3. Metode Pengujian

Metode pengujian yang dilakukan adalah dengan menguji sistem sebanyak 6 kali. Pengujian dilakukan dengan mengubah nilai probabilitas mutasi dan probabilitas penyilangan. Pengubahan nilai probabilitas tersebut digunakan untuk mencari nilai probabilitas yang paling tepat agar dapat menghasilkan jadwal yang tidak melanggar *hard-constraint*/ optimal. Perubahan nilai probabilitas penyilangan dan mutasi tersebut dilakukan sebanyak 3 dan setiap nilai probabilitas yang sama diuji sebanyak 2 kali.

Selain pengujian di atas performa sistem ini juga akan diuji. Sugihara (1997) menyatakan bahwa performa Algoritma Genetika dapat dilihat dari sudut pandang kualitas solusi tanpa memperhatikan konvergensi. Berikut ini adalah sudut pandang yang digunakan dalam pengukuran performa Algoritma Genetika:

1. *Likelihood of Optimality*

Likelihood of Optimality merupakan probabilitas solusi optimal yang dihasilkan oleh Algoritma Genetika. Algoritma Genetika dijalankan sebanyak n kali dengan k generasi. m adalah banyak solusi optimal yang dihasilkan dengan menggunakan k generasi. *Likelihood of Optimality* $Lopt(k)$ di generasi ke- k probabilitas nya dapat dicari dengan rumus m/n .

2. *Average Fitness Value*

Average Fitness Value adalah rata-rata nilai fitness $f(k)$ di generasi ke- k yang telah dijalankan sebanyak n kali.

3. *Likelihood of Evolution Leap*

Suatu generasi dikatakan *leap* (melompat) jika solusi yang dihasilkan pada suatu generasi lebih baik daripada solusi terbaik yang diperoleh generasi-generasi sebelumnya. Algoritma Genetika dieksekusi dengan k generasi setiap n kali dijalankan dan l adalah rata-rata banyaknya *leap* dengan k generasi. Probabilitas *Likelihood of Evolution Leap* $Lel(k)$ dapat pada generasi ke- k dapat dihitung menggunakan rumus l/n .

Pengukuran performa pada penelitian ini menggunakan 2 sudut pandang. Sudut pandang yang di gunakan adalah *Likelihood of Optimality* dan *Average Fitness Value*. Pengukuran performa dengan menggunakan 2 sudut pandang tersebut dapat memberikan kesimpulan rekomendasi generasi yang sebaiknya digunakan saat menjalankan Algoritma Genetika agar dapat menghasilkan solusi yang optimal (jadwal yang tidak melanggar *hard-constraint*). Adanya rekomendasi tersebut dapat mencegah kinerja komputer yang terlalu berat yang disebabkan oleh terlalu banyaknya generasi yang diberikan saat menjalankan Algoritma Genetika (Sugihara, 1997).

BAB IV

IMPLEMENTASI DAN ANALISIS KERJA PERANGKAT LUNAK

4.1. Implementasi Perangkat Lunak

Perancangan perangkat lunak diimplementasikan dalam perangkat lunak yang siap digunakan oleh *user*. Perangkat lunak dinilai baik jika perangkat lunak yang dibangun bebas dari kesalahan apapun dan mampu memberikan manfaat sesuai dengan tujuan perangkat lunak dibuat.

4.1.1 Batasan Implementasi Perangkat Lunak

Perangkat lunak yang dibangun memiliki beberapa batasan implementasi. Batasan implementasi tersebut yaitu:

1. Proses pembuatan jadwal dapat diberhentikan sebelum memenuhi batas banyak generasi, namun setelah proses pembuatan jadwal diberhentikan proses tersebut diulangi dari generasi awal.
2. *Heap memory* maksimal yang diberikan pada perangkat lunak adalah 2Gb.
3. Hasil pembuatan jadwal hanya bisa disimpan dalam bentuk *file text* dengan ekstensi *file* “.txt”.

4.1.2 Antar Muka

Perangkat lunak dibangun dengan antar muka yang dapat mempermudah *user* saat menggunakannya. Antar muka yang baik juga akan memberikan kenyamanan *user* saat menggunakan perangkat lunak. Paragraf selanjutnya akan menjelaskan antar muka perangkat lunak yang telah dibangun.

Terdapat 3 menu utama pada perangkat lunak yaitu Menu “Program”, Menu “Data” dan Menu “Jadwal”. Isi submenu “Semua Data” pada menu “Data” adalah antar muka yang digunakan untuk akses data Asisten, kelas, dan kendala jadwal di dalam database. Tampilan untuk akses data tersebut dapat dilihat pada **Gambar 4.1** , **Gambar 4.2** , dan **Gambar 4.3**.

Asisten Kelas Kendala Jadwal

Asisten Laboratorium KSC

Nama :

Angkatan :

ID Gen	Nama	Angkatan	ID Asisten
0	Ivan	Senior	1
1	Yopi	Senior	2
2	Dadang	Senior	3
3	Uzi	Senior	4
4	Deni	Senior	5
5	Izum	Senior	6
6	Lina	Senior	7
7	Aah	Junior	8

Gambar 4.1. Antar Muka Database Asisten

Asisten Kelas Kendala Jadwal

Kelas Praktikum

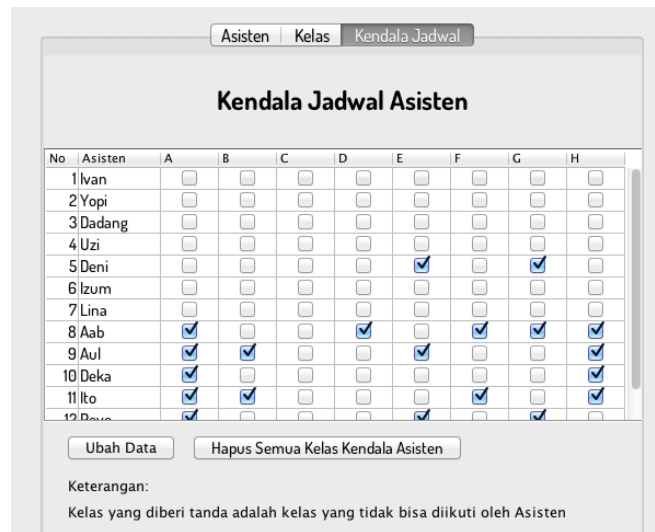
Nama Kelas :

Jumlah Asisten :

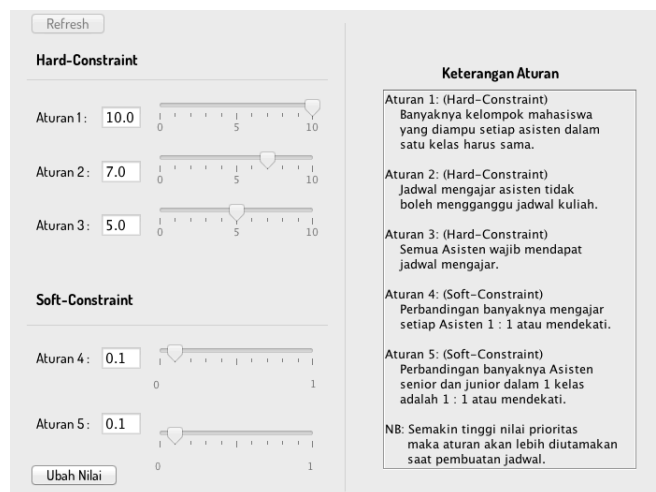
Keterangan :

No	Kelas	Kebutuhan A...	Keterangan	ID Kelas
1	A	6	Rabu, 13.00-1...	1
2	B	6	Rabu, 15.00-1...	2
3	C	6	Kamis, 08.0...	3
4	D	6	Kamis, 10.00...	4
5	E	6	Kamis, 13.00...	5
6	F	6	Kamis, 15.00...	6
7	G	6	Jumat, 07.3...	7

Gambar 4.2. Antar Muka *Database* Kelas



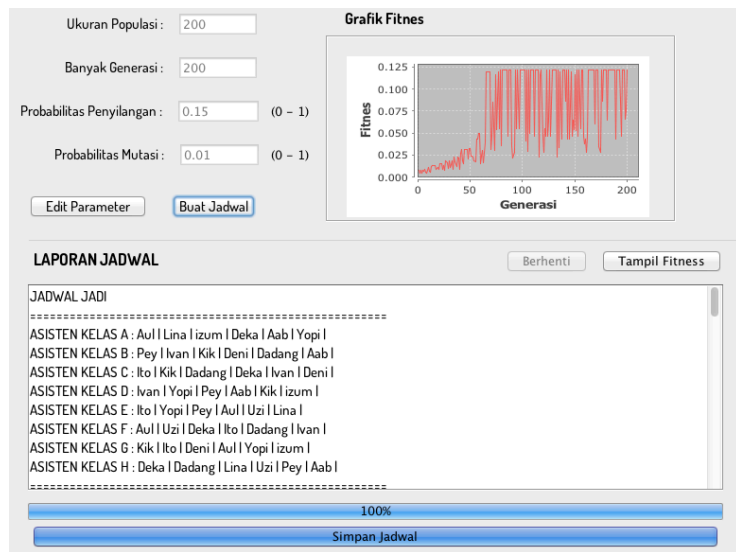
Gambar 4.3. Antar Muka *Database* Kendala Jadwal



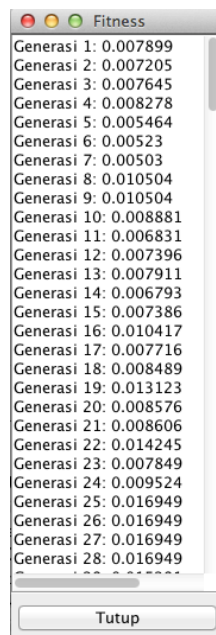
Gambar 4.4. Antar Muka Pengaturan Nilai Prioritas Aturan

Antar muka perangkat lunak yang berhubungan dengan pembuatan jadwal dan nilai parameter algoritma genetika terdapat pada menu “Jadwal”. Submenu dari menu jadwal adalah “Buat Jadwal” dan “Pengaturan Nilai Aturan” yang dapat dilihat pada **Gambar 4.4** dan **Gambar 4.5**.

Nilai *fitness* setiap generasi yang terdapat pada grafik tidak terlihat dengan rinci. Rincian *fitness* tersebut dapat ditampilkan seperti yang terlihat **pada Gambar 4.6**. dengan menekan tombol “Tampil Fitnes” pada submenu “Buat Jadwal” pada **Gambar 4.5**. setelah jadwal selesai terbuat.



Gambar 4.5. Antar Muka Pembuatan Jadwal



Gambar 4.6. Antar Muka Daftar Fitness Setiap Generasi

4.2. Analisis Kinerja Perangkat Lunak

Bab ini merupakan penjelasan dari analisis hasil kinerja perangkat lunak untuk membuat jadwal mengajar Asisten Laboratorium Komputasi dan Sistem

Cerdas. Analisis bertujuan untuk membuktikan bahwa perangkat lunak dapat bekerja dengan baik dan optimal sesuai dengan tujuan perangkat lunak dibuat.

4.2.1 Data Pengujian

Data yang digunakan dalam pengujian adalah sebagai berikut:

1. Data Asisten

Data Asisten yang digunakan untuk pengujian adalah data Asisten Laboratorium Komputasi dan sistem Cerdas yang aktif pada tahun ajaran 2013/ 2014. Asisten dibedakan menjadi 2 yaitu Asisten “Senior” dan Asisten “Junior”. Daftar Asisten tersebut dapat dilihat pada **Tabel 4.1**.

2. Data Kelas Praktikum

Data Kelas yang digunakan untuk pengujian adalah data kelas Praktikum Pemrograman Berorientasi Objek pada tahun ajaran 2013/ 2014. Data kelas ini terdiri dari nama kelas, banyak kebutuhan asisten setiap kelas, dan waktu kelas berlangsung yang berupa hari dan jam. Data tersebut dapat dilihat pada **Tabel 4.2**.

Tabel 4.1. Data Asisten

No	Nama Asisten	Angkatan
1	Ivan	Senior
2	Yopi	Senior
3	Dadang	Senior
4	Uzi	Senior
5	Deni	Senior
6	Izum	Senior
7	Lina	Senior
8	Aab	Junior
9	Aul	Junior
10	Deka	Junior
11	Ito	Junior
12	Peye	Junior
13	Kikik	Junior

Tabel 4.2. Data Kelas

No	Kelas	Banyak Asisten	Hari	Jam
1	A	6	Rabu	13.00-14.30
2	B	6	Rabu	15.00-16.30
3	C	6	Kamis	08.00-09.30
4	D	6	Kamis	10.00-11.30
5	E	6	Kamis	13.00-14.30
6	F	6	Kamis	15.00-16.30
7	G	6	Jum'at	07.30-09.00
8	H	6	Jum'at	09.30-11.00

3. Data Kendala Kelas Setiap Asisten

Data kendala kelas adalah data kelas praktikum yang tidak bisa diikuti oleh Asisten bersangkutan. Terdapat beberapa Asisten yang masih memiliki kewajiban kuliah saat diwajibkan untuk mengajar praktikum di Laboratorium. Data yang digunakan adalah data pada tahun ajaran 2013/ 2014 yang dapat dilihat pada **Tabel 4.3.** Tanda silang “X” pada kolom kelas menunjukkan bahwa kelas praktikum tersebut tidak bisa diikuti oleh Asisten bersangkutan.

Tabel 4.3. Kendala Kelas Setiap Asisten

No	Nama Asisten	Kendala Kelas							
		A	B	C	D	E	F	G	H
1	Ivan								
2	Yopi								
3	Dadang								
4	Uzi								
5	Deni					X		X	
6	Izum								
7	Lina								
8	Aab	X			X			X	X
9	Aul	X	X			X			X
10	Deka	X							X
11	Ito	X	X				X		X
12	Peye	X				X		X	
13	Kikik	X	X			X	X		

4.2.2 Pengujian Nilai Probabilitas

Pengujian dilakukan dengan data yang sudah dicantumkan pada subbab sebelumnya. Pengujian dilakukan dengan metode pengujian yang sudah dijelaskan pada bab 3 untuk mengetahui hasil optimasi perangkat lunak berupa jadwal mengajar yang dapat digunakan oleh Asisten Laboratorium Komputasi dan Sistem Cerdas.

Jadwal yang tersusun oleh perangkat lunak dapat berupa jadwal yang “dapat digunakan” dan jadwal yang “tidak dapat digunakan”. 2 kriteria jadwal tersebut dibedakan oleh pelanggaran *hard-constraint* yang terjadi pada jadwal yang tersusun. Jadwal dapat digunakan jika tidak terjadi pelanggaran *hard-constraint*, namun jika jadwal yang tersusun masih terdapat pelanggaran *hard-constraint* maka jadwal tidak dapat digunakan. Hasil pengujian dan status jadwal hasil pengujian dapat dilihat pada **Tabel 4.4** dan **Tabel 4.5** Berikut ini adalah nilai probabilitas penyilangan dan probabilitas mutasi yang diujikan:

- a. Nilai Probabilitas 1
 - Nilai Probabilitas Penyilangan: 0,15
 - Nilai Probabilitas Mutasi: 0,01
- b. Nilai Probabilitas 2
 - Nilai Probabilitas Penyilangan: 0,2
 - Nilai Probabilitas Mutasi: 0,1
- c. Nilai Probabilitas 3
 - Nilai Probabilitas Penyilangan: 0,001
 - Nilai Probabilitas Mutasi: 0,001

Penjelasan:

- a. Nilai Probabilitas 1: perkiraan nilai pertama yang ditentukan untuk melakukan pengujian yang diharapkan dapat menghasilkan jadwal yang optimal.
- b. Nilai Probabilitas 2: menambah nilai probabilitas penyilangan dan probabilitas mutasi.

- c. Nilai Probabilitas 3: mengurangi nilai probabilitas penyilangan dan probabilitas mutasi.

Tabel 4.4. Hasil Pengujian

Nomor Pengujian	Ukuran Populasi	Banyak Generasi	Probabilitas Penyilangan	Probabilitas Mutasi	Asisten Sama dalam 1 Kelas	Jadwal Bentrok Kuliah	Banyak Asisten yang tidak Mengajar	Selisih banyak mengajar	Selisih Banyak Asisten Senior dan Junior dalam 1 Kelas	Fitness Akhir
1	200	200	0,15	0,01	0	0	0	72	10	0,10869
2	200	200	0,15	0,01	0	0	0	72	8	0,11111
3	200	200	0,2	0,1	3	5	0	220	16	0,01116
4	200	200	0,2	0,1	3	8	0	272	16	0,00863
5	200	200	0,001	0,001	0	1	0	72	14	0,06024
6	200	200	0,001	0,001	0	0	0	128	8	0,06849

Tabel 4.5. Status Jadwal

Nomor Pengujian	Pelanggaran Hard-Constraint			Status Jadwal
	Aturan 1	Aturan 2	Aturan 3	
1	-	-	-	Bisa Digunakan
2	-	-	-	Bisa Digunakan
3	✓	✓	-	Tidak Bisa Digunakan
4	✓	✓	-	Tidak Bisa Digunakan
5	-	✓	-	Tidak Bisa Digunakan
6	-	-	-	Bisa Digunakan

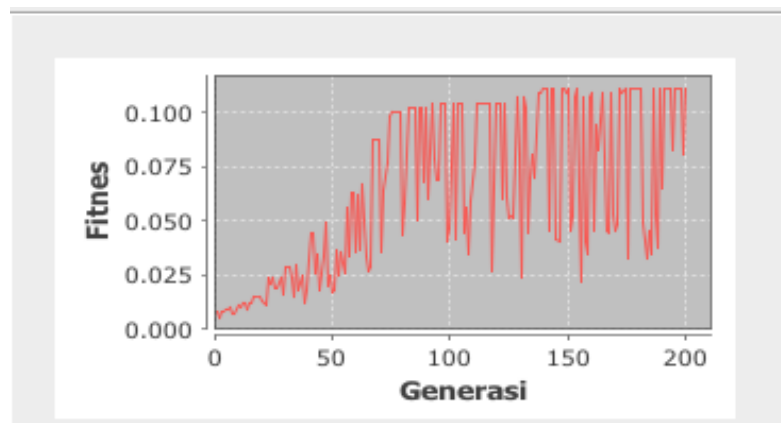
Pengujian pada **Tabel 4.4.** dilakukan tanpa merubah nilai prioritas aturan pembuatan jadwal mengajar Asisten Laboratorium Komputasi dan Sistem Cerdas.

Nilai prioritas yang digunakan adalah sebagai berikut:

1. Aturan 1: 10,0
2. Aturan 2: 7,0
3. Aturan 3: 5,0
4. Aturan 4: 0,1
5. Aturan 5: 0,1

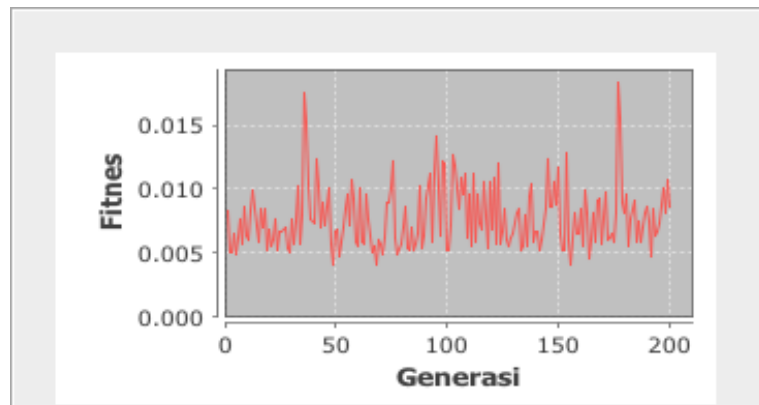
Beberapa kesimpulan yang dapat diambil dari pengujian pada **Tabel 4.4.** adalah sebagai berikut:

1. Pembuatan jadwal dengan nilai parameter Algoritma Genetika yang sama belum tentu memberikan hasil yang sama. Kesimpulan tersebut dapat terlihat pada nomor pengujian 1 dan 2, 3 dan 4, dan 5 dan 6. Pasangan nomor tersebut memiliki nilai parameter Algoritma Genetika yang sama namun memberikan nilai *fitness* yang berbeda sehingga jadwal yang dihasilkan juga berbeda.
2. Jadwal yang memiliki nilai *fitness* paling tinggi ditunjukkan pada pengujian nomor 1 dan 2 dengan menggunakan nilai probabilitas penyilangan 0,15 dan probabilitas mutasi 0,01. Hasil ini menunjukkan bahwa rekomendasi nilai probabilitas penyilangan adalah 0,15 dan probabilitas mutasi adalah 0,01. Grafik pengujian nomor 2 dapat dilihat pada **Gambar 4.7**.



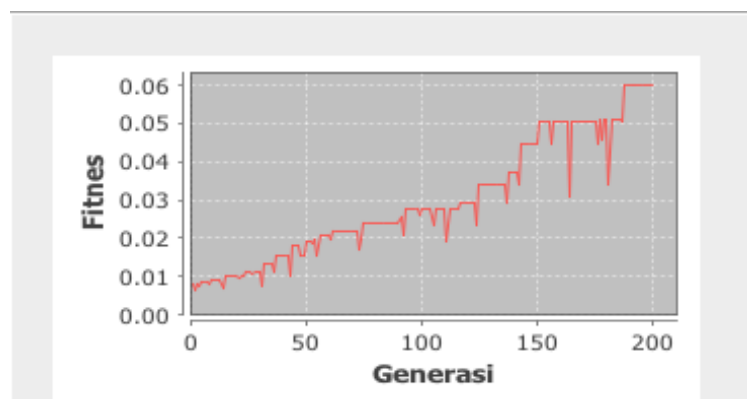
Gambar 4.7. Grafik *Fitness* Pengujian Nomor 2

3. Nilai probabilitas penyilangan dan mutasi yang terlalu tinggi menghasilkan nilai fitness yang sulit berkembang dan sulit untuk mempertahankan kromosom terbaik di generasi sebelumnya. Kesulitan tersebut terjadi karena setiap kromosom di dalam generasi memiliki kemungkinan perubahan sifat yang tinggi. Pengujian ini dapat dilihat pada pengujian nomor 3 dan nomor 4. Perkembangan nilai *fitness* pengujian nomor 3 dapat dilihat pada **Gambar 4.8**.



Gambar 4.8. Grafik *Fitness* Pengujian nomor 3

4. Nilai probabilitas penyilangan dan mutasi yang terlalu rendah dapat lebih mudah mempertahankan kromosom terbaik di generasi sebelumnya, namun perkembangan nilai *fitness* menjadi lebih lambat. Perkembangan *fitness* yang lambat tersebut disebabkan oleh kemungkinan yang kecil dalam perubahan sifat kromosom di setiap generasi. Pengujian ini terdapat pada pengujian nomor 5 dan 6. Grafik *fitness* pengujian nomor 5 dapat dilihat pada **Gambar 4.9**.



Gambar 4.9. Grafik *Fitness* Pengujian Nomor 5

Pengujian yang telah dilakukan menunjukkan bahwa pengujian pada nomor 2 pada **Tabel 4.4** adalah jadwal terbaik yang berhasil dibuat oleh perangkat lunak. Jadwal terbaik tersebut dapat dilihat pada **Gambar 4.10**. Nilai probabilitas penyilangan yang digunakan adalah 0,15 dan nilai probabilitas mutasi yang

digunakan adalah 0,01. Nilai tersebut dapat dijadikan rekomendasi nilai probabilitas untuk pembuatan jadwal dengan sistem yang telah dibuat.

```
JADWAL JADI
=====
ASISTEN KELAS A : Ivan | Yopi | Dadang | Lina | Deni | Uzi |
ASISTEN KELAS B : Ivan | izum | Dekka | Aab | Peya | Deni |
ASISTEN KELAS C : Dadang | Lina | Peya | Ito | Dekka | Uzi |
ASISTEN KELAS D : Kiki | Aul | Ito | Deni | izum | Uzi |
ASISTEN KELAS E : Ito | Dadang | Lina | Dekka | Aab | izum |
ASISTEN KELAS F : Ivan | Deni | Yopi | Aul | Aab | Dekka |
ASISTEN KELAS G : Kiki | Ito | Yopi | izum | Aul | Ivan |
ASISTEN KELAS H : Peya | Lina | Dadang | Yopi | Uzi | Kiki |
=====
```

Gambar 4.10. Jadwal Terbaik Pengujian Nomor 2

```
1. Asisten sama dalam 1 Kelas:
Tidak Ada Asisten double dalam 1 kelas

2. Jadwal Asisten Bentrok Kuliah:
Tidak Ada Jadwal Asisten yang Bentrok Kuliah

3. Banyak Asisten Mengajar dalam 1 Minggu:
Ivan: 4x
Yopi: 4x
Dadang: 4x
Uzi: 4x
Deni: 4x
izum: 4x
Lina: 4x
Aab: 3x
Aul: 3x
Dekka: 4x
Ito: 4x
Peya: 3x
Kiki: 3x

4. Perbandingan Jumlah Angkatan per Kelas
=====
Kelas : A
Senior: 6
Junior: 0
=====
Kelas : B
Senior: 3
Junior: 3
=====
Kelas : C
Senior: 3
Junior: 3
=====
Kelas : D
Senior: 3
Junior: 3
=====
Kelas : E
Senior: 3
Junior: 3
=====
Kelas : F
Senior: 3
Junior: 3
=====
Kelas : G
Senior: 3
Junior: 3
=====
Kelas : H
Senior: 4
Junior: 2
=====

Nilai Fitness Akhir: 0.1111111111111111
```

Gambar 4.11. Output Laporan Pengujian Nomor 2

Gambar 4.11 adalah laporan dari perangkat lunak tentang pelanggaran *constraint* yang terjadi pada jadwal terbaik pada **Gambar 4.10**. Berikut ini adalah penjelasan *output* Laporan pada **Gambar 4.11**.

a. Laporan Nomor 1

Laporan nomor 1 menunjukkan bahwa setiap kelas tidak terdapat asisten namanya tercantum lebih dari 1 kali.

b. Laporan Nomor 2

Laporan nomor 2 menunjukkan bahwa seluruh asisten dapat dikelasnya masing-masing sesuai dengan jadwal yang telah dibuat.

c. Laporan Nomor 3

Laporan nomor 3 pada pengujian ini menunjukkan bahwa perbandingan banyaknya mengajar asisten mendekati 1 : 1 yaitu antara 3 sampai 4 kali.

d. Laporan Nomor 4

Laporan nomor 4 menunjukkan bahwa hampir setiap kelas memiliki perbandingan banyak asisten senior dan junior 1 : 1.

e. Laporan Nilai Fitness

Laporan nomor 4 menunjukkan bahwa nilai *fitness* dari jadwal yang telah terbuat adalah 0,1111111111.

Laporan di atas menunjukkan bahwa jadwal pada **Gambar 4.10** tidak melanggar 3 *hard-constraint* yang sudah dijelaskan pada bab sebelumnya. Tidak adanya pelanggaran *hard-constraint* menunjukkan bahwa jadwal dapat digunakan oleh Asisten Laboratorium Komputasi dan Sistem Cerdas walaupun masih ditemukan pelanggaran *soft-constraint*.

Banyaknya pelanggaran *soft-constraint* menunjukkan kualitas jadwal yang tersusun. Semakin banyak pelanggaran *soft-constraint* maka kualitas jadwal semakin buruk. Pembuatan jadwal sebaiknya diulangi jika kualitas jadwal masih tidak sesuai oleh yang diinginkan Laboratorium, karena masih terdapat kemungkinan perangkat lunak dapat memberikan jadwal dengan pelanggaran *soft-constraint* yang lebih sedikit.

Sedikitnya pelanggaran *soft-constraint* dapat dilihat pada nilai *fitness* akhir pembuatan jadwal. Pengujian nomor 1 dan nomor 2 pada **Tabel 4.4** sama-sama

menghasilkan jadwal yang dapat digunakan karena tidak melanggar *hard-constraint*, namun 2 pengujian tersebut memiliki nilai *fitness* yang berbeda. Perbedaan nilai *fitness* tersebut dapat dijadikan parameter untuk menentukan perbandingan kualitas jadwal yang telah tersusun. Kualitas jadwal dapat dikatakan lebih baik jika nilai *fitness* akhir pembuatan jadwal lebih tinggi dari nilai *fitness* jadwal lainnya.

Jadwal terbaik yang berhasil tersusun tentu memiliki perbedaan yang sangat signifikan dengan jadwal terburuk yang dihasilkan oleh perangkat lunak. Pengujian nomor 3 pada **Tabel 4.4** menunjukan bahwa jadwal yang tersusun tidak dapat digunakan dan memiliki kualitas buruk. Buruknya kualitas jadwal pengujian nomor 3 ditunjukkan dengan nilai *fitness* akhir 0,01116. Nilai tersebut sangat kecil karena banyak terjadi pelanggaran *hard-constraint* dan *soft-constraint* pada jadwal yang tersusun. Jadwal yang dihasilkan oleh pengujian nomor 3 dapat dilihat pada **Gambar 4.12**.

JADWAL JADI

```
=====
ASISTEN KELAS A : Lina | Ivan | Yopi | Ito | Uzi | izum |
ASISTEN KELAS B : Peye | izum | Deka | Ito | Ivan | Dadang |
ASISTEN KELAS C : Aul | Deka | Uzi | Ivan | Kiki | Ito |
ASISTEN KELAS D : Deni | Aul | Yopi | Peye | Aul | Deka |
ASISTEN KELAS E : Peye | izum | Dadang | Lina | Lina | Yopi |
ASISTEN KELAS F : Ito | Aab | Ivan | Uzi | izum | Deni |
ASISTEN KELAS G : Uzi | Ivan | Kiki | Yopi | Aul | Ivan |
ASISTEN KELAS H : Peye | Ivan | Deni | Aab | Yopi | Kiki |
=====
```

Gambar 4.12. Hasil Jadwal Pengujian Nomor 3

```

1. Asisten sama dalam 1 Kelas
Aul --> D | Lina --> E | Ivan --> G |
2. Jadwal Asisten Bentrok Kuliah:
Ito --> A
Ito --> B
Peye --> E
Ito --> F
Aab --> H

3. Banyak Asisten Mengajar dalam 1 Minggu:
Ivan: 7x
Yopi: 5x
Dadang: 2x
Uzi: 4x
Deni: 3x
izum: 4x
Lina: 3x
Aab: 2x
Aul: 4x
Deka: 3x
Ito: 4x
Peye: 4x
Kiki: 3x

4. Perbandingan Jumlah Angkatan per Kelas
=====
Kelas : A
Senior: 5
Junior: 1
=====
Kelas : B
Senior: 3
Junior: 3
=====
Kelas : C
Senior: 2
Junior: 4
=====
Kelas : D
Senior: 2
Junior: 4
=====
Kelas : E
Senior: 5
Junior: 1
=====
Kelas : F
Senior: 4
Junior: 2
=====
Kelas : G
Senior: 4
Junior: 2
=====
Kelas : H
Senior: 3
Junior: 3
=====

Nilai Fitness Akhir: 0.011160714285714286

```

Gambar 4.13. Output Laporan Pengujian nomor 3

Gambar 4.13 adalah laporan dari perangkat lunak tentang pelanggaran *constraint* yang terjadi pada jadwal pengujian nomor 3 yang dapat dilihat pada **Gambar 4.12** Berikut ini adalah penjelasan *output* laporan pada **Gambar 4.13**.

a. Laporan Nomor 1

Laporan nomor 1 menunjukkan bahwa terdapat beberapa Asisten yang namanya tercantum lebih dari 1 kali di dalam kelas yang sama.

b. Laporan Nomor 2

Laporan nomor 2 menunjukkan terdapat beberapa jadwal mengajar yang tidak dapat ditaati oleh beberapa Asisten karena adanya kendala yang dimiliki oleh masing-masing Asisten.

c. Laporan Nomor 3

Laporan nomor 3 pada pengujian ini menunjukkan bahwa perbandingan banyaknya mengajar asisten jauh dari 1 : 1 yaitu antara 2 sampai 7 kali.

d. Laporan Nomor 4

Laporan nomor 4 menunjukkan bahwa setiap kelas tidak ada yang memiliki perbandingan banyak asisten senior dan junior 1 : 1.

e. Laporan Nilai Fitness

Laporan nomor 4 menunjukkan bahwa nilai *fitness* dari jadwal yang telah terbuat adalah 0,011160714285

Laporan di atas menunjukkan bahwa jadwal pada pengujian nomor 3 memiliki beberapa pelanggaran *hard-constraint*. Perangkat lunak melaporkan bahwa pada jadwal tersebut terdapat 3 pelanggaran pada aturan nomor 1 dan 5 pelanggaran pada aturan nomor 2. Pelanggaran *hard-constraint* tersebut yang menyebabkan jadwal tidak dapat digunakan. Jadwal pada pengujian nomor 3 tersebut juga memiliki banyak pelanggaran *soft-constraint*. Pelanggaran *soft-constraint* dapat dilihat pada perbandingan banyak mengajar asisten dalam 1 minggu pada laporan nomor 3 dan perbandingan banyak angkatan Asisten per kelas pada laporan nomor 4. Laporan tersebut dapat dijadikan alasan jadwal pada pengujian nomor 3 tidak dapat digunakan.

Laporan yang dihasilkan oleh perangkat lunak dapat digunakan *user* untuk menentukan jadwal mana yang akan digunakan. Jadwal yang dapat digunakan karena tidak melanggar *hard-constraint* belum tentu memiliki kualitas yang baik. *User* direkomendasikan memilih jadwal dengan nilai *fitness* tertinggi karena jadwal dengan nilai *fitness* tertinggi menunjukkan bahwa pelanggaran *constraint* yang terjadi pada jadwal lebih sedikit daripada jadwal yang lainnya.

4.2.3 Pengukuran Performa Algoritma Genetika

Pengukuran performa Algoritma Genetika dilakukan dengan data yang sama dengan pengujian sebelumnya. Pengukuran performa ini menggunakan 2 sudut pandang yang telah dijelaskan pada bab 3. 2 sudut pandang tersebut adalah *Likelihood of Optimality* dan *Average Fitness Value*. Pengukuran ini menggunakan nilai probabilitas penyilangan dan mutasi yang telah direkomendasikan berdasarkan pengujian sebelumnya. Nilai probabilitas penyilangan yang digunakan adalah 0,15 dan nilai probabilitas mutasi adalah 0,01.

Pengukuran ini dilakukan dengan 200 populasi dan menggunakan generasi sebanyak 25, 50, 75, 100, 125, 150, 175, dan 200. Masing-masing banyak generasi dijalankan sebanyak 10 kali. Data pengukuran Algoritma Genetika dengan sudut pandang *Likelihood of Optimality* dan *Average Fitness Value* dapat dilihat pada **Tabel 4.6** dan **Tabel 4.7**.

Tabel 4.6. Tabel *Likelihood of Optimality*

No	Banyak Generasi (<i>k</i>)	Status Solusi										Banyak Solusi Optimal (<i>m</i>)	Probabilitas Optimal <i>Lopt(k)</i>
	Run ke- (<i>n</i>)	1	2	3	4	5	6	7	8	9	10		
1	25	-	-	-	-	-	-	-	-	-	-	0	0
2	50	-	-	-	-	-	-	-	-	-	-	0	0
3	75	-	-	-	-	-	-	-	-	-	-	0	0
4	100	✓	✓	✓	✓	✓	-	-	-	✓	✓	7	0,7
5	125	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	9	0,9
6	150	✓	✓	✓	-	-	-	-	✓	✓	✓	6	0,6
7	175	✓	✓	✓	✓	-	-	✓	-	✓	✓	7	0,7
8	200	-	-	✓	✓	-	✓	✓	✓	✓	✓	7	0,7

Keterangan Status Solusi:

✓ : Solusi Optimal

- : Solusi Tidak Optimal

Tabel 4.7. Tabel *Average Fitness*

No	Banyak Generasi (<i>k</i>)	<i>Fitness (f)</i>										Rata-rata Fitness
		<i>Run ke- (n)</i>	1	2	3	4	5	6	7	8	9	
1	25	0,010660	0,020408	0,014285	0,012594	0,015773	0,012165	0,011236	0,014837	0,014493	0,018248	0,014470
2	50	0,019841	0,020576	0,025906	0,035211	0,029586	0,018315	0,022936	0,020161	0,018587	0,031847	0,024297
3	75	0,047169	0,047619	0,058139	0,046296	0,061728	0,047170	0,014085	0,025126	0,035211	0,043103	0,042565
4	100	0,094339	0,106382	0,098039	0,104166	0,064935	0,028571	0,050505	0,039683	0,106383	0,068493	0,076150
5	125	0,102040	0,081967	0,111111	0,104166	0,047170	0,074627	0,108695	0,102041	0,106383	0,104167	0,094237
6	150	0,104166	0,111111	0,078125	0,032467	0,044643	0,026316	0,041322	0,108696	0,102041	0,111111	0,076000
7	175	0,104166	0,108695	0,111111	0,104166	0,044248	0,051024	0,106383	0,045455	0,108695	0,106383	0,089033
8	200	0,051546	0,052631	0,111111	0,104166	0,051546	0,106383	0,108696	0,106383	0,111111	0,111111	0,091468

Data pada **Tabel 4.6** dan **Tabel 4.7** merupakan data proses pengukuran performa yang telah dilakukan. Berdasarkan data tersebut dapat disimpulkan bahwa performa Algoritma Genetika berada pada performa yang baik saat banyak generasi yang diberikan adalah 100 dan di atas 100 generasi. Kesimpulan tersebut dapat diambil karena nilai probabilitas solusi optimal mulai meningkat secara signifikan pada saat Algoritma Genetika dijalankan menggunakan 100 generasi.

Pada **Tabel 4.7** dapat dilihat bahwa nilai rata-rata *fitness* tertinggi adalah saat Algoritma Genetika menggunakan generasi sebanyak 125 dengan nilai *fitness* 0,094237. Berdasarkan data tersebut dapat disimpulkan bahwa banyak generasi yang direkomendasikan untuk Algoritma Genetika adalah 125 generasi. Adanya rekomendasi banyak generasi ini pengguna sistem tidak perlu menggunakan banyak generasi di atas 125 yang dapat memberatkan kinerja komputer.

4.3. Kelebihan Perangkat Lunak

Perangkat lunak *timetabling* yang dibangun memiliki kelebihan sebagai berikut:

1. Perangkat lunak dapat menghasilkan jadwal mengajar asisten yang dapat digunakan oleh Asisten Laboratorium Komputasi dan Sistem Cerdas tanpa melanggar *hard-constraint* dan *soft-constraint*.
2. Setiap jadwal yang dibuat oleh perangkat lunak dapat bervariasi sehingga asisten dapat memilih jadwal sesuai dengan yang diinginkan.

4.4. Kekurangan Perangkat Lunak

Perangkat lunak yang dibangun juga memiliki kekurangan, kekurangan tersebut antara lain:

1. Kemampuan Algoritma Genetika dalam perangkat lunak untuk mempertahankan kromosom terbaik di generasi sebelumnya masih kurang optimal.
2. Jadwal yang dihasilkan masih dimungkinkan terjadi pelanggaran *hard-constraint*.
3. Perangkat lunak membutuhkan *Heap memory* yang banyak untuk membuat jadwal dengan Algoritma Genetika.

BAB V

KESIMPULAN DAN SARAN

5.1. Kesimpulan

Perangkat lunak *timetabling* yang dibangun sudah dilakukan beberapa pengujian. Berikut ini adalah beberapa kesimpulan yang didapat:

1. Algoritma Genetika dapat digunakan untuk melakukan optimasi penyusunan jadwal/ *timetabling*.
2. Jadwal yang dihasilkan sangat dipengaruhi oleh nilai parameter Algoritma Genetika dan nilai parameter prioritas *constraint*.
3. Jadwal yang dibangun memiliki hasil yang bervariasi walaupun jadwal dibuat dengan nilai parameter Algoritma Genetika dan nilai parameter *constraint* yang sama.
4. Waktu kompilasi yang dibutuhkan sangat tergantung pada nilai parameter Algoritma Genetika dan banyaknya data.
5. Nilai parameter Algoritma Genetika yang tinggi dapat menghabiskan *heap memory* dengan cepat.

5.2. Saran

Perangkat lunak yang dibuat pada Tugas Akhir ini memiliki beberapa keterbatasan yang dapat dikembangkan lebih lanjut di penelitian berikutnya. Beberapa saran yang diberikan untuk penelitian berikutnya adalah sebagai berikut:

1. Perangkat lunak dapat lebih meningkatkan kemampuan dalam mempertahankan kromosom terbaik di generasi sebelumnya.
2. Perangkat lunak dapat mencapai nilai *fitness* terbaik dengan ukuran populasi dan generasi yang lebih kecil.
3. Perangkat lunak dapat lebih meminimalisir pelanggaran *hard-constraint* terjadi.
4. Perangkat lunak yang dibangun dapat lebih menghemat *heap memory* jika dikembangkan lagi dengan bahasa pemrograman JAVA.

DAFTAR PUSTAKA

- Gen, M., & Cheng, R. (1997). *Genetic Algorithms and Engineering Design*. Ashikaga: John Wiley & Son, Inc.
- Hindriyanto. (2012, September 29). *Antara Optimasi, Heuristik, dan Metaheuristik*. Retrieved February 22, 2014, from Duniaku: <http://hindriyanto.wordpress.com/2012/09/26/antara-optimisasi-heuristik-dan-metaheuristik/>
- Erben, W., & Keppler, J. (1995). A Genetic Algorithm Solving a Weekly Course-Timetabling Problem. *The College of Information Sciences and Technology* .
- Miller, D., Nusbaum, D., & Fine, H. (1974, March 14). *The Free Library*. (Farlex, Inc) Retrieved February 25, 2014, from The Free Library: <http://www.thefreedictionary.com/timetable>
- Blum, C., & Roli, A. (2003). *Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison*. Broadway, New York: Dept., ACM, Inc.
- Randy L, H., & Sue, H. E. (2004). *Practical Genetic Algorithm* (Vol. 2). Canada: A John Wiley & Sons, Inc.
- Hutomo, A., Fitrananda, A., & etc. (2011, April). Implementasi Algoritma Integer Linear Programming Untuk Sistem Informasi Penjadwalan Ruangan Di Fakultas Ilmu Komputer Universitas Indonesia. *Journal of Information System* , 26-33.
- Hiryanto, L., & Josselyn, S. (2013). Implementasi Vertex Graph Colouring, Particle Swarm Optimization, Dan Constraint Based Reasoning Untuk University Timetabling Problem Implementasi Vertex Graph Colouring, Particle Swarm Optimization, Dan Constraint Based Reasoning Untuk University Timetabling Problem. *Ilmu Komputer dan Sistem Informasi* , 97-104.
- Talbi, E. (2009). *Metaheuristik: from design to implementation*. Hoboken, New Jersey: John Wiley & Son, Inc.
- Kusumadewi, S., & Purnomo, H. (2005). *Penyelesaian Masalah Optimasi Menggunakan Teknik-teknik Heuristik* . Yogyakarta, Yogyakarta: Graha Ilmu.
- Sugihara, K. (1997). Measures for Performance Evaluation of Genetic Algorithms. *Conference on Information Sciences* .
- Zukhri, Z. (2014). *Algoritma Genetika, Metode Komputasi Evolusioner untuk Menyelesaikan Masalah Optimasi*. Yogyakarta, Yogyakarta: Andi.

