IMPLEMENTASI HYBRID ALGORITMA GENETIKA UNTUK PENJADWALAN AUDITOR PADA AUDIT INTERNAL MUTU UNIVERSITAS BRAWIJAYA

SKRIPSI

Untuk memenuhi sebagian persyaratan mencapai gelar Sarjana Komputer

Disusun oleh :

AMELIA

105090600111021

PROGRAM STUDI INFORMATIKA / ILMU KOMPUTER

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

2014

LEMBAR PERSETUJUAN

IMPLEMENTASI HYBRID ALGORITMA GENETIKA UNTUK PENJADWALAN AUDITOR PADA AUDIT INTERNAL MUTU UNIVERSITAS BRAWIJAYA

SKRIPSI
KONSENTRASI KOMPUTASI CERDAS DAN VISUALISASI

Untuk memenuhi sebagian persyaratan untuk
Mencapai gelar Sarjana Komputer

Disusun Oleh:
AMELIA
105090600111021

Telah diperiksa dan disetujui oleh
Dosen Pembimbing

Pembimbing I,	**Pembimbing II,**
Rekyan Regasari M.P., ST., MT.	Diah Priharsari, ST., MT.
NIK. 77041406120253	

LEMBAR PENGESAHAN

IMPLEMENTASI HYBRID ALGORITMA GENETIKA UNTUK PENJADWALAN AUDITOR PADA AUDIT INTERNAL MUTU UNIVERSITAS BRAWIJAYA

SKRIPSI

LABORATORIUM KOMPUTASI CERDAS DAN VISUALISASI
Untuk memenuhi sebagian persyaratan untuk mencapai gelar Sarjana Komputer

Disusun oleh :

Amelia
105090600111021

Setelah dipertahankan di depan Majelis Penguji
pada tanggal 20 Agustus 2014
dan dinyatakan memenuhi syarat untuk memperoleh
gelar sarjana dalam bidang Ilmu Komputer

Penguji I,	**Penguji II,**
Drs. Marji, MT.	Ir. Sutrisno, MT.
NIP. 196708011992031001	**NIP. 195703251987011000**

Penguji III,

Budi Darma Setiawan, S.Kom., M.Cs.
NIP. 84101506110090

Mengetahui,
Ketu Program Studi Informatika / Ilmu Komputer

Drs. Marji, MT.
NIP. 196708011992031001

PERNYATAAN
ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah SKRIPSI ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata di dalam naskah SKRIPSI ini dapat dibuktikan terdapat unsur-unsur PLAGIASI, saya bersedia SKRIPSI ini digugurkan dan gelar akademik yang telah saya peroleh (SARJANA) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70.

Malang, 26 Juni 2014
Mahasiswa,

Amelia
NIM. 105090600111021

KATA PENGANTAR

Alhamdulillahi rabbil 'alamiin, puji syukur penulis panjatkan ke hadirat Allah SWT yang telah melimpahkan segala rahmat, karunia, hidayah serta bimbingan-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul: " **IMPLEMENTASI HYBRID ALGORITMA GENETIKA UNTUK PENJADWALAN AUDITOR PADA AUDIT INTERNAL MUTU UNIVERSITAS BRAWIJAYA** ".

Skripsi ini diajukan sebagai syarat ujian skripsi dalam rangka untuk memperoleh gelar Sarjana Komputer di Program Teknologi Informasi Dan Ilmu Komputer (PTIIK), Program Studi Informatika/Ilmu Komputer, Universitas Brawijaya Malang. Dalam pelaksanaan dan penulisan tugas akhir ini, penulis banyak sekali mendapatkan dukungan dan bantuan dari berbagai pihak. Dalam kesempatan ini penulis ingin mengucapkan terima kasih yang sebesar-besarnya kepada:

1. Ibu Rekyan Regasari Mardi Putri, ST., MT., selaku Dosen Pembimbing Skripsi pertama yang telah meluangkan waktu dan juga memberikan pengarahan bagi penulis.
2. Ibu Diah Priharsari, ST., MT., selaku Dosen Pembimbing Skripsi kedua yang telah meluangkan waktu dan juga memberikan pengarahan bagi penulis.
3. Drs. Marji, MT. selaku Ketua Program Studi Informatika/Ilmu Komputer Program Teknologi Informasi & Ilmu Komputer Universitas Brawijaya
4. Ir. Sutrisno, MT., selaku Ketua Program Teknologi Informasi & Ilmu Komputer Universitas Brawijaya.
5. Kedua orang tua penulis. Papa dan Mama tercinta atas segala kasih sayang, do'a yang tak henti, nasihat, pelajaran hidup yang berharga dan segala keikhlasan yang diberikan.
6. Alfian Nur Hidayat, atas segenap dukungan, motivasi dan bantuan yang telah diberikan selama pengerjaan skripsi ini.
7. Sahabat Penulis serta teman – teman seperjuangan Ilmu Komputer angkatan 2010 yang memotivasi dan saling menyemangati.

8. Segenap Bapak dan Ibu dosen yang telah mendidik dan mengajarkan ilmunya kepada Penulis selama menempuh pendidikan di Program Teknologi Informasi & Ilmu Komputer Universitas Brawijaya.
9. Segenap staff dan karyawan di Program Studi Teknologi Informasi & Ilmu Komputer Universitas Brawijaya yang telah banyak membantu dalam hal administrasi Penulis dalam pelaksanaan penyusunan skripsi ini.

Penulis berharap agar skripsi ini dapat memberikan sumbangan dan manfaat bagi semua pihak yang berkepentingan.

Penulis menyadari bahwa skripsi ini tentunya tidak terlepas dari ketidaksempurnaan. Oleh karena itu, dengan tangan terbuka penulis menerima segala kritik dan saran yang bersifat membangun demi perbaikan dimasa yang akan datang.

Malang, 26 Juni 2014

Penulis,

Amelia

NIM. 105090600111021

ABSTRAK

Amelia. 2014 : Implementasi Hybrid Algoritma Genetika untuk Penjadwalan Auditor pada Audit Internal Mutu Universitas Brawijaya.
Dosen Pembimbing : Rekyan Regasari M.P., ST., MT. dan Diah Priharsari, ST., MT.

Penjadwalan merupakan proses dalam menyusun suatu jadwal atau urutan proses yang diperlukan dalam suatu permasalahan. Guna meningkatkan mutu dan mempertahankan standar pelayanan Audit Internal Mutu, instansi harus memiliki sistem penjadwalan yang berkualitas dikarenakan padatnya sistem pelayanan yang ada di dalamnya. Penjadwalan auditor dipengaruhi oleh dua komponen yakni jumlah auditor dan jumlah *auditee* dengan sejumlah batasan-batasan tertentu, dimana batasan-batasan tersebut ada yang harus dipenuhi atau tidak boleh dilanggar. Salah satu algoritma yang dapat digunakan untuk menyelesaikan permasalahan penjadwalan auditor ini adalah *hybrid* algoritma genetika. *Hybrid* algoritma genetika merupakan hibridisasi dari algoritma genetika dimana metode yang digunakan sebagai kombinasi pada penelitian ini adalah *tabu search*. Pada prinsipnya, *tabu search* digunakan untuk memfilter kromosom yang mengalami *crossover* agar kromosom yang sama tidak dilakukan *crossover* berulang-ulang. Sehingga diharapkan dapat memberikan hasil optimum dan mengurangi waktu pencarian solusi. Data yang digunakan pada penelitian ini adalah data Audit Internal Mutu UKPPA Siklus 7 Universitas Brawijaya dengan jumlah *auditee* sebanyak 31 dan jumlah auditor sebanyak 46. Dari hasil pengujian yang dilakukan didapat bahwa metode *hybrid* algoritma genetika mampu memberikan solusi dengan rata-rata jumlah generasi lebih sedikit dibandingkan dengan metode algoritma genetika pada semua parameter pengujian yang digunakan walaupun tidak signifikan karena penggunaan data yang kecil. Namun secara keseluruhan *hybrid* algoritma genetika mampu menyelesaikan permasalahan penjadwalan lebih baik dibanding dengan algoritma genetika.

Kata Kunci*: Hybrid* Algoritma Genetika, Tabu Search, Optimasi, Penjadwalan, Kesediaan

ABSTRACT

Amelia. 2014 : Hybrid Genetic Algorithm Implementation to Generate Internal Quality Audit Schedule in Brawijaya University.
Advisor : Rekyan Regasari M.P., ST., MT. and Diah Priharsari, ST., MT.

Scheduling is the process of preparing a schedule or sequence of processes required in a problem. In order to improve the quality and maintain the standards of Internal Quality Audit service, the agency must have a quality system due to the tight scheduling service system in it. Scheduling of audit is affected by two components, that are the number of auditors and the number of auditees with certain constraints, which some of constraints must be met or should not be violated. One algorithm that can be used to solve scheduling of audit problems are hybrid genetic algorithm. Hybrid genetic algorithm is a hybridization of a genetic algorithm, which is used as a combination method in this study is tabu search method. In principle, tabu search method is used to filter chromosomes which will undergo crossover, so that the same chromosome does not undergo crossover repeatedly. Expected to give optimum results and reduce the search time solution. The data used in this study is from Internal Quality Audit data Cycle 7 UKKPA Brawijaya University with 31 of auditees and 46 of auditors. The result of tests shown that the hybrid genetic algorithm provide a solution to the average number of generations is less than genetic algorithm without hybridization on all parameters used in tests, although not significant due to the use of small data. But overall hybrid genetic algorithm is able to solve scheduling problems better than genetic algorithm without hybridization.

***Keywords**: Hybrid Genetic Algorithm, Tabu Search, Optimization, Scheduling, Availability*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR SOURCE CODE

BAB I
PENDAHULUAN

1.1. Latar Belakang

Untuk meningkatkan mutu secara berkelanjutan, relevansi dan efisiensi layanan Universitas Brawijaya (UB) di era global, maka pengelolaan UB membutuhkan Sistem Penjaminan Mutu (SPM). Penjaminan mutu merupakan serangkaian proses dan sistem yang terkait untuk mengumpulkan, menganalisis, dan melaporkan data mengenai kinerja mutu didik dan tenaga kependidikan program, dan lembaga. Proses penjaminan mutu mengidentifikasikan aspek pencapaian dan prioritas peningkatan, menyediakan data sebagai dasar perencanaan dan pengambilan keputusan, serta membantu membangun budaya peningkatan mutu secara berkelanjutan. Pusat Jaminan Mutu (PJM) Universitas Brawijaya (UB) menerapkan jaminan mutu sesuai Sistem Penjaminan Mutu Internal (SPMI). Audit Internal Mutu (AIM) adalah pemeriksaan sistematik dan independen untuk mengetahui apakah implementasi Sistem Penjaminan Mutu Internal (SPMI) efektif dan sesuai dengan perencanaan yang dilakukan oleh unit kerja di Universitas Brawijaya (UB).

Pada pelaksanaan Audit Internal Mutu (AIM) sering sekali dijumpai kendala dalam penyusunan jadwal audit. Dalam menyusun suatu rencana audit, auditor harus mempertimbangkan sumber daya audit yang dimilikinya dibandingkan dengan kebutuhan auditnya. Jika jumlah kebutuhan audit dan sumber daya audit yang harus dikelola oleh auditor cukup banyak maka proses ini tentunya akan menyulitkan jika dilakukan dengan cara manual karena terdapat banyak sekali kombinasi jadwal yang dapat dibuat. Jumlah auditor yang terdapat di Universitas Brawijaya saat ini sebanyak 157 auditor dengan jumlah *auditee* 122 untuk AIM UKPA (Unit Kerja Pelaksana Akademik) yang terdiri dari Fakultas, Program, Jurusan, Program Studi, dan 31 *auditee* untuk AIM UKKPA (Unit Kerja Penunjang Pelaksana Akademik) yang terdiri dari biro, lembaga, unit pelaksanaan teknis, dan unit-unit lain. Dengan keterbatasan waktu pelaksanaan audit serta jumlah setiap tim audit 2 orang yang terdiri dari ketua dan anggota, dapat

terbentuk lebih dari 1000 kombinasi jadwal dimana harus dipilih satu jadwal yang dianggap paling optimal (tidak melanggar *constraint* yang telah ditetapkan) membuat permasalahan penjadwalan auditor ini menjadi sangat kompleks. Sehingga dalam kasus penjadwalan auditor diperlukan algoritma yang dapat meyelesaikan masalah multi-kriteria dan multi-objektif. Salah satu algoritma yang dapat digunakan adalah algoritma genetika.

Algoritma Genetika (*genetic algorithm*) banyak dipakai pada aplikasi bisnis, teknik maupun bidang keilmuan. Algoritma ini dapat dipakai untuk mendapatkan solusi yang tepat untuk masalah optimal dari satu variabel atau multi variabel. Sebelum algoritma ini dijalankan, masalah apa yang ingin dioptimalkan itu harus dinyatakan dalam fungsi tujuan, yang dikenal dengan fungsi *fitness*. Jika nilai *fitness* semakin besar, maka sistem yang dihasilkan semakin baik. Walaupun pada awalnya semua nilai *fitness* kemungkinan sangat kecil (karena algoritma ini menghasilkannya secara random), sebagian akan lebih tinggi dari yang lain. Kromosom dengan nilai *fitness* yang tinggi ini akan memberikan probabilitas yang tinggi untuk bereproduksi pada generasi selanjutnya. Sehingga untuk setiap generasi pada proses evolusi, fungsi *fitness* yang mensimulasikan seleksi alam, akan menekan populasi kearah *fitness* yang meningkat [SUK-03].

Komang Setemen (2009), dalam tugas akhirnya menyelesaikan permasalahan penjadwalan mata kuliah dengan menggunakan kombinasi algoritma genetika dan *tabu search*. Hasil yang didapat pada jumlah data yang kecil (45 sampai 88 data), kombinasi algoritma genetika dan *tabu search* dapat menyelesaikan masalah penjadwalan dengan baik [SET-09]. Irfan Darmawan (2011), dalam penelitiannya juga menggunakan perpaduan antara algoritma genetika dan *tabu search* untuk menyelesaikan permasalahan penyeimbangan beban terhadap *resource* yang terhubung. Hasil penelitian menunjukkan bahwa penggabungan kedua algoritma tersebut yang disimulasikan dengan 21 job dan 7 *cluster* memberikan hasil lebih baik dibanding dengan algoritma genetika [DAR-11].

Dari beberapa penelitian yang telah dilakukan tersebut dapat dilihat bahwa penggabungan algoritma genetika dengan metode *tabu search* memiliki hasil yang lebih baik dibanding dengan penggunanaan algoritma genetika murni. Pada

prinsipnya, *tabu search* digunakan untuk memfilter kromosom yang mengalami *crossover* agar kromosom yang sama tidak dilakukan *crossover* berulang-ulang. Sehingga diharapkan dapat memberikan hasil optimum dan mengurangi waktu pencarian solusi. Maka dari itu, pada penelitian ini akan digunakan algoritma genetika serta mengkombinasikannya pada algoritma *tabu search* atau yang dikenal dengan *hybrid genetic algorithm* dengan harapan memperoleh hasil optimasi penjadwalan terbaik.

1.2. Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, maka didapat permasalahan dalam penelitian ini adalah:

1. Bagaimana mengimplementasikan metode *hybrid* algoritma genetika dengan teknik *tabu search* untuk menyelesaikan masalah penjadwalan auditor.
2. Bagaimana perbandingan hasil yang diperoleh dari implementasi *hybrid* algoritma genetika dengan algoritma genetika dalam menyelesaikan permasalahaan penjadwalan.

1.3. Batasan Masalah

Berdasarkan latar belakang yang telah diuraikan, maka batasan masalah yang digunakan pada penelitian ini adalah:

1. Objek yang digunakan pada penelitian ini diperoleh dari data *auditee* dan auditor Audit Internal Mutu UKPPA Siklus 7 Universitas Brawijaya.
2. Parameter-parameter algoritma genetika yang digunakan adalah jumlah individu, probabilitas *crossover*, dan probabilitas mutasi (*mutation*).
3. Aplikasi yang dibuat tidak mempertimbangkan jam kegiatan mengajar dosen yang merupakan tim auditor.
4. Aplikasi yang dibuat tidak mencakup penanganan terhadap adanya perubahan jadwal auditor atau *auditee* secara tiba-tiba. Jika terdapat perubahan jadwal dari auditor atau *auditee*, maka sistem akan merubah hasil penjadwalan secara keseluruhan.

5. Aplikasi penjadwalan auditor dibangun menggunakan bahasa pemrograman Java.
6. *Database management System* yang digunakan adalah MySQL.

1.4. Tujuan

Berdasarkan latar belakang yang telah diuraikan, maka tujuan dari penelitian ini adalah:

1. Membangun dan merancang sistem penjadwalan auditor menggunakan metode *hybrid* algoritma genetika.
2. Membandingkan hasil dari implementasi metode *hybrid* algoritma genetika dengan metode algoritma genetika dalam menyelesaikan permasalahan penjadwalan audit.

1.5. Manfaat

Manfaat yang diharapkan dengan adanya penelitian ini adalah:

1. Meningkatkan efektifitas dan efisiensi dalam penyusunan jadwal auditor. Efektifitas berarti jadwal yang telah ada bisa dikerjakan dengan tepat, sedangkan efisien adalah dapat mengurangi waktu pencarian solusi jadwal.
2. Memberikan kontribusi dan acuan serta pertimbangan bagi pengelolaan sistem penjadwalan auditor pada suatu institusi perguruan tinggi.

1.6. Sistematika Penulisan

Sistematika penulisan penelitian ditujukan untuk memberikan gambaran serta uraian dari penyusunan tugas akhir secara garis besar yang meliputi beberapa bab, sebagai berikut:

BAB I PENDAHULUAN

Menjelaskan mengenai latar belakang permasalahan, perumusan masalah, tujuan penelitian yang ingin dicapai, manfaat dari hasil penelitian, batasan masalah yang digunakan, serta sistematika pembahasan.

BAB II KAJIAN PUSTAKA DAN DASAR TEORI

Menguraikan tentang kajian pustaka atau referensi yang mendasari dibuatnya penelitian, dan teori-teori yang berkaitan dengan penjadwalan dan hybrid algoritma genetika.

BAB III METODE PENELITIAN DAN PERANCANGAN

Menguraikan tentang metode dan langkah kerja yang digunakan dalam penelitian yang terdiri dari studi literatur, metode pengambilan data, analisis kebutuhan, perancangan aplikasi, implementasi, pengujian dan analisis serta pengambilan kesimpulan. Menguraikan perancangan aplikasi yang menjadi objek studi kasus penjadwalan auditor.

BAB IV IMPLEMENTASI

Membahas implementasi dari aplikasi penjadwalan auditor yang sesuai dengan perancangan aplikasi yang telah dibuat.

BAB V PENGUJIAN DAN ANALISIS

Memuat hasil pengujian serta hasil analisis terhadap aplikasi yang telah dibangun.

BAB VI PENUTUP

Mamuat kesimpulan yang diperoleh dari pembuatan dan pengujian aplikasi yang dibangun dalam penelitian ini, serta saran-saran guna pengembangan penelitian lebih lanjut.

BAB II
KAJIAN PUSTAKA DAN DASAR TEORI

2.1 Kajian Pustaka

Sebelumnya telah ada penelitian tentang implementasi *hybrid* algoritma genetika yang merupakan penggabungan antara algoritma genetika dan *tabu search*. Irfan Darmawan (2011), dalam penelitiannya menggunakan perpaduan antara algoritma genetika dan *tabu search* untuk menyelesaikan permasalahan penyeimbangan beban terhadap *resource* yang terhubung. Penggabungan algoritma genetika dan *tabu search* pada penelitian tersebut digunakan untuk menutup kelemahan eksploitasi pada algoritma genetika dengan menyisipkan metode *tabu search* pada posisi operasi pindah silang (*crossover*) dan atau pada operasi mutasi. Hasil penelitian menunjukkan bahwa penggabungan kedua algoritma tersebut yang disimulasikan dengan 21 job dan 7 *cluster* memberikan hasil lebih baik dibanding dengan algoritma genetika [DAR-11].

Komang Setemen (2009), melakukan penelitian untuk menyelesaikan permasalahan penjadwalan mata kuliah dengan menggunakan dua macam metode. Metode pertama yakni menggunakan algoritma genetika, dan metode kedua menggunakan kombinasi algoritma genetika dan *tabu search*. Hasil yang didapat pada penggunaan algoritma genetika murni memberikan nilai *fitness* 0 dan jumlah iterasi atau generasi maksimum sebanyak 5000 pada jumlah kelompok data yang kecil (53 sampai 88). Sedangkan untuk data yang dengan jumlah pemetaan 141 mata kuliah, algoritma genetika murni memberikan nilai *fitness* 0 dengan jumlah iterasi atau generasi maksimum sebanyak 10.000 generasi. Hasil yang didapat pada penggunaan kombinasi algoritma genetika dan *tabu search* pada jumlah data yang kecil (45 sampai 88 data), kombinasi algoritma genetika dan *tabu search* dapat menyelesaikan masalah penjadwalan dengan baik, dengan nilai *fitness* sama dengan 0 dan iterasi atau jumlah generasi maksimum 1000 generasi. Sedangkan untuk data yang cukup besar yaitu dengan jumlah pemetaan mata kuliah 160 mata kuliah, kombinasi algoritma ini masih mampu memperoleh *fitness* 0 tetapi dengan jumlah iterasi atau generasi maksimum 10.000 generasi [SET-09].

Kombinasi algoritma genetika dan *tabu search* dapat memeberikan hasil yang lebih baik di bandingkan hanya dengan menggunakan algoritma genetika saja. Hal ini dapat dilihat dari hasil jumlah iterasi, maupun kualitas solusi yang dihasilkan. Pada penelitian tersebut, *tabu search* digunakan untuk memfilter kromosom yang mengalami *crossover* agar kromosom yang sama tidak dilakukan *crossover* berulang-ulang. Saat iterasi pertama kali, semua kromosom yang mengalami *crossover* disimpan ke dalam *tabu list*, kemudian baru dilakukan *crossover*. Untuk iterasi selanjutnya, kromosom yang mengalami *crossover* terlebih dahulu di cek pada *tabu list*, apakah kromosom tersebut sudah ada atau belum. Jika kromosom tersebut sudah ada pada *tabu list*, maka kromosom tersebut akan dilakukan mutasi, jika belum maka kromosom tersebut disimpan pada *tabu list*, kemudian dilakukan *crossover* [SET-09].

2.2 Audit Internal Mutu

Audit Internal Mutu (AIM) adalah pemeriksaan sistematik dan independen untuk mengetahui apakah implementasi Sistem Penjaminan Mutu Internal (SPMI) efektif dan sesuai perencanaan yang dilakukan oleh unit kerja di Universitas Brawijaya (UB). AIM di UB dilakukan untuk memeriksa kesesuaian antara standar mutu yang ditetapkan, dalam Sistem Manajemen Mutu (SMM) Akademik, Keuangan dan Administrasi dengan pelaksanaannya oleh unit kerja di UB. Komponen paling penting dari pelaksanaan audit adalah adanya Auditor dan *Auditee*. Auditor adalah seseorang yang memiliki kemampuan (kompetensi) untuk melakukan audit. Sedangkan *Auditee* atau teraudit adalah organisasi atau orang yang diaudit. Audit Internal Mutu sangat bermanfaat untuk [ADM-11]:

1. Membantu mengidentifikasi setiap ketidaksesuaian yang telah dan atau sedang terjadi, serta hal-hal yang kemudian hari mempuyai kecenderungan untuk menimbulkan masalah (terjadi ketidaksesuaian).
2. Menjamin kesesuaian sistem terdokumentasi terhadap persyaratan standar yang diacu.
3. Menjamin kesesuaian aktivitas yang diterapkan dengan sistem terdokumentasi.

4. Menjamin konsistensi penerapan sistem.
5. Memastikan keefektifan penerapan sistem.
6. Meningkatkan/mengembangkan sistem.

AIM di UB dilakukan dalam dua tahap, yaitu [ADM-11]:

1. Audit Sistem: Audit terhadap kecukupan organisasi penjaminan mutu dan dokumen mutu untuk memenuhi persyaratan standar sistem audit mutu.
2. Audit Kinerja (Keputusan): Audit pada implementasi sistem penjaminan mutu yang telah ditetapkan/dijanjikan.

Prosedur pelaksanaan AIM untuk UKPA (Unit Kerja Pelaksana Akademik) dan UKPPA (Unit Kerja Penunjang Pelaksana Akademik) dilakukan langkah-langkah berikut:

1. PJM melalui Wakil Ketua Bidang (Wakabid) AIM menyiapkan ruang lingkup dan dokumen AIM UKPA atau AIM UKPPA, kemudian menyampaikan kepada Pembantu Rektor I.
2. Rektor melalui Pembantu Rektor I menyetujui dan mengesahkan ruang lingkup AIM UKPA atau AIM UKPPA.
3. PJM melalui Wakabid AIM menyusun jadwal sosialisasi dan pelaksanaan AIM, serta menentukan Tim AIM.
4. PJM meminta kesediaan dan alokasi waktu auditor internal.
5. Pembantu Rektor I didampingi oleh PJM melakukan sosialisasi dan koordinasi pelaksanaan AIM dengan pada *auditee*.
6. PJM melalui Wakabid AIM menyelenggarakan *refreshing* AIM UKPA atau AIM UKPPA untuk para auditor internal.
7. Pembantu Rektor I meminta kesediaan dan waktu pelaksanaan audit kepada *auditee*.
8. Wakabid AIM menyusun konsep Surat Tugas Tim AIM disertai distribusi auditor internal, jadwal dan tempat pelaksanaan AIM.
9. Rektor mengesahkan Surat Tugas Tim AIM.
10. Pembantu Rektor I menyampaikan pemberitahuan kepada *auditee* untuk mempersiapkan diri menghadapi AIM sesuai dengan ruang lingkup yang

ditetapkan dan menginstruksikan untuk mempersiapkan dokumen sesuai ruang lingkup AIM dan mengirimnya ke PJM.

11. Wakabid AIM menyampaikan *checklist* AIM kepada *auditee* dan auditor internal.
12. Sebelum pelaksanaan AIM, auditor internal dan MR setiap UKPA atau UKPPA mengkoordinasikan pelaksanaan AIM di masing-masing unit kerja.
13. Auditor internal melakukan AIM sesuai ruang lingkup SMM pada masing-masing unit kerja melalui pemeriksaan lapangan, wawancara, dan pemeriksaan dokumen.
14. Auditor internal mendiskusikan hasil temuan AIM dengan *auditee* untuk mendapatkan persetujuan, kemudian menyerahkan hasilnya (*softcopy* dan *hardcopy*) dan menyampaikan umpan balik pelaksanaan AIM kepada Wakabid AIM.
15. *Auditee* menyampaikan umpan balik pelaksanaan AIM kepada Wakabid AIM.
16. Wakabid AIM menyusun laporan hasil AIM kemudian menyampaikan kepada Rektor (melalui Ketua PJM) disertai dengan konsep permintaan tindakan koreksi (PTK).
17. Rektor menerbitkan dan menyampaikan surat permintaan tindakan koreksi kepada Pimpinan UKPA atau Pimpinan UKPPA.
18. Pimpinan UKPA atau Pimpinan UKPPA melakukan tindakan koreksi sesuai permintaan Rektor dan melaporkannya kepada Rektor melalui PJM.

2.3 Penjadwalan

Penjadwalan adalah pengalokasian waktu yang tersedia untuk melaksanakan masing-masing pekerjaan dalam rangka menyelesaikan suatu kegiatan hingga tercapai hasil yang optimal dengan mempertimbangkan keterbatasan-keterbatasan yang ada [HUS-08]. Sedangkan menurut Wren (1996), penjadwalan adalah proses pengorganisasian, pemilihan, dan penentuan waktu penggunaan sumber-sumber daya untuk mengerjakan semua aktifitas yang diperlukan yang memenuhi kendala aktifitas dan sumber daya [WRE-96].

Masalah penjadwalan tenaga kerja secara umum adalah masalah dalam penentuan berapa banyak pekerja yang harus ditempatkan terhadap setiap periode perencanaan dari waktu kerja pada sebuah organisasi. Karena sumber daya manusia adalah komponen yang sangat penting diantara semua organisasi baik di industri manufaktur ataupun jasa, menjadwalkan tenaga kerja adalah masalah yang umum bagi semua organisasi [GUN-99]. Kriteria untuk penjadwalan tenaga kerja didasarkan pada efisiensi penggunaan sumber daya tenaga kerja dengan tetap memperhatikan aturan dalam pekerjaan.

Masalah penjadwalan merupakan masalah yang sulit. Bahkan menjadi lebih sulit ketika harus mempertimbangkan tujuan yang banyak (*multiple objectives*). Adanya kombinasi dalam masalah penjadwalan mengakibatkan hal ini menjadi sulit untuk diselesaikan walaupun dengan cara teknik matematika. Hal-hal yang menjadi faktor pertimbangan dalam penyusunan jadwal yaitu adanya batasan waktu pengerjaan (*completion time*), aliran waktu (*flow time*), yang merupakan urutan pengerjaan aktivitas termasuk waktu tunggu (*waiting time*), dan waktu proses (*processing time*). Penjadwalan yang tepat akan mencapai tujuan yang optimal dengan memenuhi semua kenada [SIA-10].

2.1.1 Tujuan Penjadwalan

Beberapa tujuan dari penjadwalan adalah sebagai berikut [GIN-09]:

1. Meningkatkan penggunan sumber daya atau mengurangi waktu tunggunya, sehingga total waktu proses dapat berkurang, dan produktifitasnya dapat meningkat.
2. Mengurangi beberapa keterlambatan pada pekerjaan yang mempunyai batas waktu penyelesaian sehingga meminimasi *penalty cost* (biaya keterlambatan).

2.1.2 Manfaat Penjadwalan

Secara umum, manfaat dari dibuatnya penjadwalan adalah sebagai berikut [HUS-09]:

1. Memberikan pedoman terhadap pekerjaan/kegiatan mengenai batas-batas waktu untuk memulai dan akhir dari masing-masing tugas.

2. Memberikan alat bagi pihak manajemen untuk mengkoordinir secara sistematis dan realistis dalam penentuan alokasi prioritas terhadap sumber daya dan waktu.
3. Memberikan saran untuk menilai kemajuan pekerjaan.
4. Menghindari pemakaian sumber daya yang berlebihan.
5. Memberikan kepastian waktu pelaksanaan pekerjaan.

2.1.3 Performansi Penjadwalan

Menurut Baker , terdapat tiga tujuan pembuatan keputusan yang umum dalam penjadwalan dan ketiganya menunjukkan ukuran dasar performansi jadwal, yaitu [BAK-74]:

1. Pemanfaatan sumber daya yang efisien: minimum maksimum saat selesai.
2. Respon yang cepat terhadap permintaan konsumen: minimum rata-rata saat selesai (*completion time*), minimum rata-rata waktu tinggal (*flow time*), atau minimum rata-rata waktu tunggu (*waiting time*).
3. Sesuai dengan batas waktu yang ditentukan: minimum rata-rata keterlambatan (*tardiness*), minimum maksimum keterlambatan, dan minimum jumlah job yang terlambat (*the number of tardy jobs*).

2.4 Penjadwalan Auditor

Masalah penjadwalan karyawan banyak dijumpai pada industri jasa, salah satunya di instansi pendidikan. Penjadwalan merupakan proses dalam menyusun suatu jadwal atau urutan proses yang diperlukan dalam suatu permasalahan. Guna meningkatkan mutu dan mempertahankan standar pelayanan Audit Internal Mutu, instansi harus memiliki sistem penjadwalan yang berkualitas dikarenakan padatnya sistem pelayanan yang ada di dalamnya. Salah satu penjadwalan yang harus diperhatikan adalah penjadwalan auditor.

Penjadwalan auditor dipengaruhi oleh dua komponen yakni jumlah auditor dan jumlah *auditee* dengan sejumlah batasan-batasan tertentu, dimana batasan-batasan tersebut ada yang harus dipenuhi atau tidak boleh dilanggar. Batasan tersebut merupakan ukuran kualitas dari penjadwalan auditor, sehingga suatu

jadwal auditor yang optimal dapat terbentuk [ZAI-13]. Terdapat dua batasan dalam penyusunan jadwal auditor, yaitu *hard constraint* (harus terpenuhi) dan *soft constraint* (diupayakan untuk terpenuhi).

Hard constraint merupakan batasan-batasan yang harus diterapkan pada penjadwalan auditor dan harus dipenuhi. Suatu solusi hanya dapat dikatakan valid apabila dalam solusi tersebut sama sekali tidak ada *hard constraint* yang terlanggar [ZAI-13]. *Hard constraint* yang digunakan dalam penjadwalan auditor pada Audit Internal Mutu (AIM) UKPA dan UKPPA Universitas Brawijaya adalah sebagai berikut:

1. Jadwal yang dihasilkan harus sesuai dengan waktu kesediaan auditor dan *auditee*.
2. Seorang auditor tidak diperkenankan melakukan audit dalam ruang lingkup pekerjaanya sendiri.
3. Seorang auditor tidak diperkenankan melakukan audit dalam ruang lingkup yang dapat mengganggu independensi yang bersifat pribadi.
4. *Auditee* tidak diperkenankan memilih auditor sendiri.
5. Tim Audit tidak boleh terdiri dari auditor yang berasal dari unit kerja (fakultas) yang sama.
6. Tim Audit terdiri dari Ketua Auditor dan Anggota Auditor.
7. Auditor yang dapat dijadikan sebagai Ketua Tim Audit adalah auditor yang pernah melakukan audit minimal 2 kali siklus audit.
8. Sesama Anggota Tim Audit tidak dapat dipasangkan menjadi satu Tim Audit.
9. *Auditee* pada tingkat magister hanya dapat diaudit oleh auditor dengan lulusan minimal S2.

Berbeda dengan *hard constraint*, *soft constraint* merupakan kendala yang tidak selalu dapat terpenuhi dalam proses pembentukan jadwal. Akan tetapi, meskipun tidak harus terpenuhi, jadwal yang dihasilkan harus semaksimal mungkin berusaha memenuhi ketentuan *soft constraint* [ZAI-13]. Contoh *soft constraint* dalam penjadwalan auditor pada Audit Internal Mutu (AIM) UKPA dan UKPPA di Universitas Brawijaya antara lain:

1. Auditor hanya dapat melakukan audit satu kali dalam satu hari jadwal audit.
2. Auditor hanya dapat melakukan audit maksimal 5 kali dalam satu siklus audit.
3. Satu siklus Audit Internal Mutu dilakukan dalam jangka waktu 5 hari kerja.
4. Pembagian jadwal harus seimbang antar semua auditor.

Sebelum dilakukannya AIM, auditor dan *auditee* diminta untuk memberikan kesediaan alokasi waktu pelaksanaan audit sesuai dengan jadwal yang telah ditentukan oleh PJM sebelumnya. Untuk auditor diminta memilih minimal 4 (empat) waktu berbeda, sedangkan untuk *auditee* hanya dapat memilih 1 (satu) waktu untuk pelaksanaan audit. Berikut contoh alokasi waktu yang diberikan untuk pelaksanaan AIM:

- ☐ Senin, 15 Oktober 2012, jam 08.30 s.d 11.30 WIB
- ☐ Senin, 15 Oktober 2012, jam 13.00 s.d 16.00 WIB
- ☐ Selasa, 16 Oktober 2012, jam 08.30 s.d 11.30 WIB
- ☐ Selasa, 16 Oktober 2012, jam 13.00 s.d 16.00 WIB
- ☐ Rabu, 17 Oktober 2012, jam 08.30 s.d 11.30 WIB
- ☐ Rabu, 17 Oktober 2012, jam 13.00 s.d 16.00 WIB
- ☐ Kamis, 18 Oktober 2012, jam 08.30 s.d 11.30 WIB
- ☐ Kamis, 18 Oktober 2012, jam 13.00 s.d 16.00 WIB
- ☐ Jum'at, 19 Oktober 2012, jam 08.30 s.d 11.30 WIB
- ☐ Jum'at, 19 Oktober 2012, jam 13.00 s.d 16.00 WIB

2.5 Algoritma Genetika

Algoritma genetika adalah algoritma pencarian yang didasarkan pada mekanisme seleksi ilmiah dan genetika alamiah. Pada awalnya algoritma genetika

digunakan sebagai algoritma pencarian parameter-parameter optimal. Tetapi dalam pengembangannya, algoritma genetika bisa diaplikasikan untuk berbagai masalah lain, seperti *learning*, peramalan, pemrograman otomatis, dan sebagainya. Pada bidang *soft computing* algoritma genetika banyak digunakan untuk mendapat nilai-nilai parameter pada jaringan syaraf tiruan maupun sistem fuzzy [SUY-11].

Algoritma Genetika (*genetic algorithm*) banyak dipakai pada aplikasi bisnis, teknik maupun bidang keilmuan. Algoritma ini dapat dipakai untuk mendapatkan solusi yang tepat untuk masalah optimal dari satu variabel atau multi variabel. Sebelum algoritma ini dijalakan, masalah apa yang ingni dioptimalkan itu harus dinyatakan dalam fungsi tujuan, yang dikenal dengan fungsi *fitness*. Jika nilai *fitness* semakin besar, maka sistem yang dihasilkan semakin baik. Walaupun pada awalnya semua nilai *fitness* kemungkinan sangat kecil (karena algoritma ini menghasilkannya secara random), sebagian akan lebih tinggi dari yang lain. Kromosom dengan nilai *fitness* yang tinggi ini akan memberikan probabilitas yang tinggi untuk bereproduksi pada generasi selanjutnya. Sehingga untuk setiap generasi pada proses evolusi, fungsi *fitness* yang mensimulasikan seleksi alam, akan menekan populasi kearah *fitness* yang meningkat [SUK-03].

Keuntungan penggunaan algoritma genetika sangat jelas terlihat dari kemudahan implementasi dan kemampuannya untuk menemukan solusi yang bagus (bisa diterima) secara cepat untuk masalah-masalah berdimensi tinggi. Ciri-ciri permasalahan yang dapat dikerjakan dengan menggunakan algoritma genetika adalah [BAS-03]:

1. Mempunyai fungsi tujuan optimalisasi non linear dengan banyak kendala yang juga non linear.
2. Mempunyai kemungkinan solusi yang jumlahnya tak berhingga.
3. Membutuhkan solusi *real time* dalam arti solusi bisa didapatkan dengan cepat sehingga dapat diimplementasikan untuk permasalahan yang mempunyai perubahan yang cepat.
4. Mempunyai multi-objektif dan multi-kriteria, sehingga diperlukan solusi yang dapat secara bijak diterima oleh semua pihak.

Selain itu juga algoritma genetika mempunyai karakteristik-karakteristik yang perlu diketahui sehingga dapat terbedakan dari prosedur pencarian atau optimasi yang lain, yaitu:

1. Algoritma genetika bekerja dengan pengkodean dari himpunan solusi permasalahan berdasarkan parameter yang telah ditetapkan dan bukan parameter itu sendiri.
2. Algoritma genetika melakukan pencarian pada sebuah populasi dari sejumlah individu-individu yang merupakan solusi permasalahan bukan hanya dari sebuah individu.
3. Algoritma genetika merupakan informasi fungsi objektif (*fitness*), sebagai cara untuk mengevaluasi individu yang mempunyai solusi terbaik, bukan turunan dari suatu fungsi.
4. Algoritma genetika menggunakan aturan-aturan transisi peluang, bukan aturan-aturan deterministik.

2.3.1 Proses Pada Algoritma Genetika

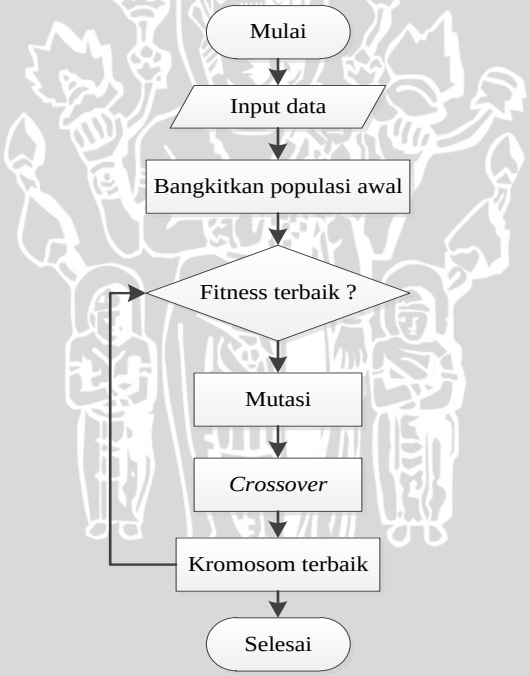
Algoritma genetika adalah algoritma pencarian yang berdasarkan pada mekanisme sistem natural yakni genetik dan seleksi alam. Dalam aplikasi algoritma genetika, variabel solusi dikodekan ke dalam struktur *string* yang merepresentasikan barisan gen, yang merupakan karakteristik dari solusi problem.berbeda dengan teknik pencarian konvensional. Algoritma genetika berangkat dari himpunan solusi yang dihasilkan secara acak. Himpunan ini disebut populasi. Sedangkan setiap individu dalam populasi disebut kromosom yang merupakan representasi dari solusi. Kromosom-kromosom berevolusi dalam suatu proses iterasi yang berkelanjutan yang disebut generasi. Pada setiap generasi, kromosom dievaluasi berdasarkan suatu fungsi evaluasi. Setelah beberapa generasi maka algoritma genetika akan konvergen pada kromosom terbaik, yang diharapkan merupakan solusi optimal.

Algoritma genetika adalah algoritma pencarian hasil yang terbaik, yang didasarkan pada persilangan dan seleksi gen secara alami.kombinasi persilangan ini dilakukan dengan proses acak (*random*). Dimana struktur gen hasil proses persilangan ini akan menghasilkan gen inovatif untuk diseleksi. Dalam setiap

generasi, *offspring* yang baru (hasil persilangan) diperoleh dari bit-bit dan bagian-bagian gen induk yang terbaik. Diharapkan dengan mengambil dari gen induk yang terbaik ini diperoleh gen anak yang lebih baik lagi. Meskipun pada kenyataannya tidak selalu tercipta gen anak yang lebih baik dari induknya. Ada kemungkinan lebih baik, sama baiknya, atau bahkan mungkin lebih buruk.

Tujuan dari algoritma genetika ini adalah menghasilkan populasi yang terbaik dari populasi awal. Sedangkan keuntungan dari algoritma genetika adalah sifat metode pencariannya yang lebih optimal, tanpa terlalu memperbesar ruang pencarian.

Dalam menyusun suatu algoritma genetika menjadi program, maka diperlukan beberapa tahapan proses yakni proses pembuatan generasi awal, proses genetika, proses seleksi, dan pengulangan proses sebelumnya.



Gambar 2.1 *Flowchart* Proses Algoritma Genetika

Sumber: [ABD-08]

Pada algoritma ini terdapat beberapa definisi penting yang harus dipahami sebelumnya, yaitu:

a. Gen

   Gen merupakan nilai yang menyatakan satuan dasar yang membentuk suatu arti tertentu dalam satu kesatuan gen yang dinamakan kromosom.

b. Kromosom / Individu

Kromosom merupakan gabungan dari gen-gen yang memberntuk nilai tertentu dan menyatakan solusi yang mungkin dari suatu permasalahan.

c. Populasi

Populasi merupakan sekumpulan individu yang akan diproses bersama dalam satu satuan siklus evolusi.

d. *Fitness*

Fitness menyatakan seberapa baik nilai dari suatu individu yang didapatkan.

e. Seleksi

Seleksi merupakan proses untuk mendapatkan calon induk yang baik.

f. *Crossover*

Crossover merupakan proses pertukaran atau pindah silang gen-gen dari dua induk tertentu.

g. Mutasi

Mutasi merupakan proses pergantian salah satu gen yang terpilih dengan nilai tertentu.

h. Generasi

Generasi merupakan urutan iterasi dimana beberapa kromosom bergabung.

i. *Offspring*

Offspring merupakan kromosom baru yang dihasilkan.

Untuk menyelesaikan suatu permasalahan menggunakan algoritma genetika, perlu diketahui beberapa macam *encoding* (pengkodean) guna menentukan operator *crossover* dan mutasi yang akan digunakan. Pengkodean tersebut tergantung pada permasalahan apa yang diangkat. Beberapa macam pengkodean akan dijelaskan di bawah ini:

1) *Binary Encoding* (Pengkodean Biner)

Pengkodean jenis ini adalah yang sering digunakan dalam penyelesaian masalah. Kromosom dari pengkodean biner ini berupa kumpulan dari nilai biner 0 dan 1.

Contoh:

Kromosom 1	1101100100110110
Kromosom 2	1101011000011110

Dalam pengkodean biner memungkinkan didapatkan kromosom dengan nilai *allele* yang kecil, tetapi kekurangannya tidak dapat digunakan untuk beberapa permsalahan dan terkadang diperlukan adanya koreksi setelah proses *crossover* dan mutasi. Salah satu permsalahan yang menggunakan pengkodean adalah menghitung nilai maksimal dari suatu fungsi.

2) *Permutation Encoding* (Pengkodean Permutasi)

Kromosom dari pengkodean permutasi ini berupa kumpulan dari nilai integer yang mewakili suatu posisi dalam sebuah urutan. Biasanya digunakan pada permsalahan TSP (*Travelling Salesman Problem*).

Contoh:

Kromosom 1	1 4 7 9 6 3 5 0 2 8
Kromosom 2	9 3 2 5 8 1 6 0 4 7

3) *Value Encoding* (Pengkodean Nilai)

Kromosom dari pengkodean nilai ini berupa kumpulan suatu nilai, yang bisa berupa macam-macam nilai sesuai dengan permsalahan yang dihadapi, seperti bilangan riil, karakter atau objek yang lai. Pengkodean ini merupakan pilihan yang bagus untuk beberapa permasalahan khusus, biasanya diperlukan metode khusus untuk memproses *crossover* dan mutasinya sesuai dengan permsalahan yang dihapadi.

Contoh:

Kromosom 1	A B E D B C A E D D
Kromosom 2	N W W N E S S W N N

4) *Tree Encoding* (Pengkodean Pohon)

Pengkodean pohon biasanya digunakan pada pemrograman genetika. Kromosom yang diguakan berupa sebuah pohon dari beberapa objek, seperti fungsi atau *command* pada pemrograman genetika.

2.3.1.1 Membuat Generasi Awal

Pendefinisian individu merupakan proses pertama yang harus dilakukan dalam algoritma genetika yang menyatakan salah satu solusi yang mungkin dari suatu permasalahan yang diangkat. Proses pembuatan generasi awal dilakukan dengan membangkitkan populasi secara acak, dimana populasi tersebut berisi beberapa kromosom yang telah didefinisikan sebelumnya. Dalam proses ini perlu diperhatikan syarat-syarat yang harus dipenuhi untuk menunjukkan suatu solusi dari permasalahan dan jumlah kromosom yang digunakan dalam satu populasi. Jika kromosom yang digunakan terlalu sedikit, maka individu yang dapat digunakan untuk proses *crossover* dan mutasi akan sangat terbatas, sehingga menyia-nyiakan proses yang ada. Tetapi jika jumlah kromosom yang digunakan terlalu banyak, akan memperlambat proses algoritma genetika yang dilakukan. Jumlah kromosom yang dianjurkan lebih besar dari jumlah gen yang ada dalam satu kromosom, tetapi juga harus disesuaikan dengan permasalahan, apabila jumlah gennya terlalu banyak, tidak juga dianjurkan seperti itu.

Pertama, dibentuk sebuah populasi untuk sejumlah gen. Susunan gen ini terbentuk dari kromosom yang disusun membentuk suatu *string*. Nilai *string* dibentuk secara *random* dengan memilih setiap kromosom dengan kode tertentu secara *random* pada *string*. Setiap *string* didekodekan menjadi satu set parameter yang dapat mewakilinya. Parameter ini merupakan model numerik ruang permasalahan. Model numerik ini memberikan pemecahan berdasarkan masukan parameter.

Sebagai dasar kualitas pemecahan, setiap *string* diberi nilai *fitness*. Dengan nilai *fitness* tersebut, operasi genetika (pindah silang, dan mutasi) dipergunakan untuk menciptakan generasi baru *string*. Set *string* baru tersebut didekodekan dan dievaluasi lagi, sampai generasi *string* yang baru terbentuk kembali. Proses ini diulang-ulang sampai sejumlah inputan generasi dari *user*.

2.3.1.2 Fungsi Tujuan (*Fitness*)

Fungsi tujuan (*fitness*) adalah fungsi yang dimiliki oleh masing-masing individu untuk menentukan tingkat kesesuaian individu tersebut dengan kriteria yang ingin dicapai [KAR-11]. Nilai *fitness* populasi kromosom menggambarkan

kualitas kromosom dalam populasi tersebut. Fungsi *fitnessi* tersebut sebagai berikut:

$$Fitness = \frac{1}{1+\Sigma penalty} \quad (2\text{-}1)$$

Dari persamaan di atas, nilai *fitness* ditentukan oleh nilai pinalti. Pinalti tersebut menunjukkan jumlah pelanggaran *constraint* pada suatu kromosom. Semakin tinggi nilai *fitness* akan semakin besar kemungkinan kromosom tersebut terpilih ke generasi berikutnya. Jadi nilai pinalti berbanding terbalik dengan nilai *fitness*, semakin kecil nilai pinalti (jumlah pelanggaran) semakin besar nilai *fitness*-nya.

2.3.1.3 Proses Seleksi

Operasi seleksi dilakukan dengan memperhatikan *fitness* dari tiap individu, manakah yang dapat dipergunakan untuk generasi selanjutnya. Seleksi ini digunakan untuk mendapatkan calon induk yang baik, semakin tinggi nilai *fitness*-nya maka semakin besar juga kemungkinan individu tersebut terpilih. Terdapat beberapa macam cara seleksi untuk mendapatkan calon induk yang baik, diantaranya adalah seleksi *roulette wheel*, *steady state*, *tournament* dan elitis. Proses seleksi yang biasa digunakan adalah mesin *roulette* (*roulette wheel*) atau metode elitis. Pada penelitian ini, metode seleksi yang digunakan adalah metode elitis.

Konsep elitis (*elitism*) dalam algoritma genetika berarti usaha mempertahankan individu-individu terbaik yang telah diperoleh di suatu generasi ke dalam generasi selanjutnya. Sehingga individu-individu terbaik ini akan muncul di populasi berikutnya. Langkah ini dilakukan dalam berbagai cara. Misalnya, melalui penyalinan individu terbaik atau dapat juga melalui kombinasi antara solusi-solusi turunan atau anak dengan induk. Terbukti bahwa penggunaan operator elitisme ini telah terbukti memiliki pengaruh yang sangat penting saat menggunakan algoritma genetika untuk menyelesaikan persoalan optimasi dengan tujuan tunggal [SIA-12].

2.3.1.4 Proses Mutasi

Mutasi juga merupakan salah satu operator penting dalam algoritma genetika selain *crossover*. Metode dan tipe mutasi yang dilakukan juga tergantung pada pengkodean yang digunakan dan permasalahan yang diangkat. Berdasarkan teknik pengkodeannya, terdapat beberapa macam cara mutasi diantaranya adalah sebagai berikut:

1) *Binary Encoding* (Pengkodean Biner)

   Mutasi pada pengkodean biner merupakan operasi yang sangat sederhana. Proses yang dilakukan adalah melakukan inversi pada bit yang terpilih, 0 menjadi 1 dan sebaliknya, 1 menjadi 0.

   Contoh:

Kromosom sebelum mutasi :	1	**1**	0	0	1	0	0	1
Kromosom setelah mutasi :	1	**0**	0	0	1	0	0	1

2) *Permutastion Encoding* (Pengkodean Permutasi)

   Proses mutasi dalam pengkodean permutasi dilakukan dengan memilih dua gen atau posisi dari kromosom dan menukarnya.

   Contoh:

Kromosom sbelum mutasi :	1	2	**3**	4	**5**	6	7	8
Kromosom setelah mutasi :	1	2	**5**	4	**3**	6	7	8

Karena proses mutasi juga merupakan salah satu operator dasar dalam algoritma genetika, sehingga sama dengan *crossover*, mutasi juga memerlukan probabilitas dengan proses yang sama seperti pada probabilitas *crossover*. Individu dengan nilai probabilitas yang lebih kecil dari probabilitas yang telah ditentukan yang akan melewati proses mutasi. Nilai probabilitas mutasi ini menunjukkan seberapa sering gen tertentu dari kromosom yang telah diproses dengan *crossover* akan melewati mutasi. Jika tidak ada proses mutasi, maka *offspring* yang dihasilkan akan sama dengan hasil individu setelah proses *crossover*, tanpa ada perubahan sedikitpun. Proses mutasi ini biasanya dilakukan untuk mencegah terjadinya lokal optimum, proses mutasi ini sebaiknya tidak terlalu sering dilakukan karena proses algoritma genetika akan cepat berubah

menjadi *random search*. Pada probabilitas mutasi, jika terlalu rendah akan mengakibatkan banyak gen yang berguna tidak sempat untuk dimanfaatkan dan jika terlalu besar akan menyebabkan *offspring* kehilangan sifat dari induknya dan tidak akan dapat memanfaatkan lagi proses evolusi alamiah.

2.3.1.5 Proses Pindah Silang (*Crossover*)

Proses pindah silang (*crossover*) adalah salah satu operator penting dalam algoritma genetika, metode dan tipe *crossover* yang dilakukan tergantung dari pengkodean dan permasalahan yang diangkat. Ada beberapa cara yang bisa digunakan untuk melakukan *crossover* sesuai dengan pengkodeannya yang dijelaskan sebagai berikut:

1) *Binary Encoding* (Pengkodean Biner)

   a. *Crossover* Satu Titik

   Proses *crossover* dilakukan dengan memilih satu titik tertentu, selanjutnya nilai biner sampai titik *crossover*-nya dari induk pertama digunakan dan sisanya dilanjutkan dengan nilai biner dari induk kedua.

   Contoh:

Parent 1	1	1	0	0	1	0	1	1
Parent 2	1	1	0	1	1	1	1	1
Offspring	1	1	0	0	1	1	1	1

   b. *Crossover* Dua Titik

   Proses *crossover* dilakukan dengan memilih dua titik tertentu, lalu nilai biner sampai titik *crossover* pertama pada induk pertama digunakan, dilanjutkan dengan nilai biner dari titik pertama sampai titik kedua dari induk kedua. Kemudian sisanya melanjutkan nilai biner dari titik kedua pada induk pertama lagi.

   Contoh:

Parent 1	1	1	0	0	1	0	1	1
Parent 2	1	1	0	1	1	1	1	1
Offspring	1	1	0	0	1	1	1	1

c. *Crossover Uniform*

Proses *crossover* dilakukan dengan memilih secara random dari kedua induk.

Contoh:

Parent 1	1	1	0	0	1	0	1	1
Parent 2	1	1	0	1	1	1	1	1
Offspring	1	1	0	0	1	1	1	1

d. *Crossover* Aritmatik

Proses *crossover* dilakukan dengan menggunakan operasi aritmatik antara induk pertama dan kedua untuk menghasilkan *offspring* yang baru.

Contoh (pada operasi AND):

Parent 1	1	1	0	0	1	0	1	1
Parent 2	1	1	0	1	1	1	1	1
Offspring	1	1	0	0	1	0	1	1

2) *Permutation Encoding* (Pengkodean Permutasi)

Memilih satu titik tertentu, nilai permutasi sampai titik *crossover* pada induk pertama digunakan lalu sisanya dilakukan *scan* terlebih dahulu, jika nilai permutasi pada induk kedua belum ada pada *offspring* nilai tersebut ditambahkan.

Contoh:

Parent 1	1	2	3	4	5	6	7	8
Parent 2	2	1	8	7	6	5	3	4
Offspring	1	2	3	4	5	8	7	6

3) *Value Encoding* (Pengkodean Nilai)

Semua metode pada *binary crossover* dapat digunakan pada pengkodean nilai.

2.3.1.6 Parameter Genetika

Pengoperasian algoritma genetika dibutuhkan 4 parameter, yaitu [JUN-03]:

- Probabilitas Persilangan (*Probability Crossover*)

  Menunjukkan kemungkinan *crossover* terjadi antara 2 kromosom. Jika tidak terjadi *crossover* maka keturunannya akan sama persis dengan kromosom induk, tetapi tidak berarti generasi yang baru akan sama persis dengan generasi yang lama. Jika probabilitas *crossover*-nya 100%, maka semua keturunannya dihasilkan dari *crossover*. *Crossover* dilakukan dengan harapan bahwa kromosom yang baru akan lebih baik.

- Probabilitas Mutasi (*Probability Mutation*)

  Menunjukkan kemungkinan mutasi terjadi pada gen-gen yang menyusun sebuah kromosom. Jika tidak terjadi mutasi maka keturunan yang dihasilkan setelah *crossover* tidak berubah. Jika terjadi mutasi bagian kromosom akan berubah. Jika probabilitasnya 100%, semua kromosom dimutasi. Jika probabilitasnya 0%, tidak ada yang mengalami mutasi.

- Jumlah Individu

  Menunjukkan jumlah kromosom yang terdapat dalam populasi (dalam satu generasi). Jika hanya sedikit kromosom dalam populasi maka algoritma genetika akan mempuyai sedikit variasi kemungkinan untuk melakukan *crossover* antara induk karena hanya sebagian kecil dari *search space* yang dipakai. Sebaliknya, jika terlalu banyak maka algoritma genetika akan berjalan lambat.

- Jumlah Populasi

  Menentukan jumlah populasi atau banyaknya generasi yang dihasilkan, digunakan sebagai batas akhir proses seleksi, persilangan, dan mutasi.

2.3.1.7 Kondisi Berhenti

Offspring merupakan kromosom baru yang dihasilkan setelah melalui proses-proses di atas. Kemudian pada *offspring* tersebut dihitung nilai *fitness*-nya apakah sudah optimal atau belum, jika sudah optimal berarti *offspring* tersebut merupakan solusi optimal, tetapi jika belum optimal maka akan diseleksi kembali, begitu seterusnya sampai terpenuhi kriteria berhenti.

2.6 *Tabu Search*

Tabu seacrh pertama kali diperkenalkan oleh Gover sekitar tahun 1986. *Tabu search* merupakan sauatu metode optimasi yang menggunakan *short-term memory* untuk menjaga agar proses pencarian tidak terjebak pada nilai optimum lokal. Metode ini menggunakan *tabu list* untuk menyimpan beberapa solusi yang baru saja dievaluasi. Selama proses optimasi, pada setiap iterasi, solusi yang akan dievaluasi akan dicocokkan terlebih dahulu dengan isi *tabu list* untuk melihat apakah solusi tersebut sudah ada pada *tabu list*. Apabila solusi itu sudah ada pada tabu *list*, maka solusi itu tidak akan dievaluasi lagi pada iterasi berikutnya. Apabila sudah tidak ada lagi solusi yang tidak menjadi anggota *tabu list*, maka nilai terbaik yang diperoleh merupakan solusi yang sebenarnya [KUS-05].

Pemilihan kandidat terbaik didasarkan nilai fungsi tujuan. Pemeriksaan nilai fungsi tujuan terlebih dahulu dilakukan sebelum pemeriksaan status *tabu*. Apabila nilai fungsi tujuan sebuah kandidat lebih baik dari yang lain, maka kandidat tersebut berotensi untuk diterima sehingga perlu diperiksa status *tabu*-nya. Pemilihan kandidat solusi terbaik yang dilakukan oleh *tabu search* menggunakan prinsip *global test strategy* bukan *first best strategy*. *Global test strategy* adalah strategi dimana algoritma akan mengganti solusi terbaik saat ini dengan solusi terbaik yang ada pada *neighborhood*. Sedangkan *first best strategy* adalah strategi dimana algoritma akan mengganti solusi terbaik saat ini secara langsung jika solusi yang lebih baik ditemukan.

Secara garis besar, algoritma *tabu search* dapat ditulis sebagai berikut [KUS-05]:

Langkah 0. Tetapkan:
X = Matriks input berukuran m x n
MaxIter = maksimum iterasi

Langkah1. S = bangkitkan solusi secara random

Langkah 2. GlobalMin = FCost(S)

Langkah 3. Best = S

Langkah 4. TabuList = [ ]

Langkah 5. Kerjakan dari k = 1 sampai MaxIter:

Langkah 6. BestSoFar = FCost(S)
Langkah 7. BestMove = S
Langkah 8. Kerjakan dari i = 1 sampai (n-1):
Langkah 9. Kerjakan dari j = 1 sampai n:
Langkah 10. L = Tukar(S[i] , S[j])
Langkah 11. Cost = FCost(L)
Langkah 12. Jika (L € TabuList) atau (Cost < GlobalMin), kerjakan:
Langkah 13. Jika (Cost < BestSoFar), kerjakan:
Langkah 14. BestSoFar = Cost
Langkah 15. BestMove = L
Langkah 16. S = BestMove
Langkah 17. Tambahkan S ke TabuList
Langkah 18. Jika BestSoFar < GlobalMin, kerjakan:
Langakh 19. GlobalMin = BestSoFar
Langkah 20. Best = BestMove

Solusi akhir adalah Best, dengan *cost* sebesar GlobalMin.

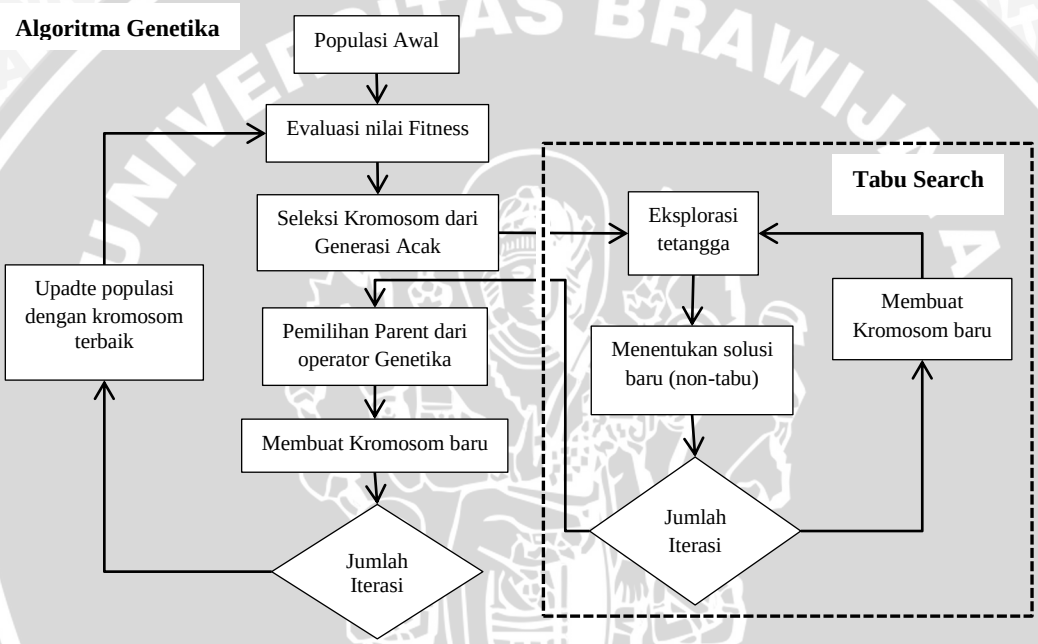
2.7 *Hybrid* Algoritma Genetika

Telah banyak penelitian yang membuktikan bahwa standar algoritma algorima genetika sering kali lebih mengarah pada solusi lokal optimum, khususnya pada masalah-masalah yang cukup besar. Atas dasar tersebut, beberapa peneliti mengkombinasikan metode-metode heuristik lain kedalam algoritma genetika atau biasa disebut *hybrid* algoritma genetika dengan harapan mampu meningkatkan kinerja algoritma genetika [SYA-08]. Pada prinsipnya hibridisasi ini diharapkan mampu memberikan solusi lain yang lebih baik disekitar lokal optimum yang diberikan oleh algoritma genetika atau dikenal dengan istilah *local search*.

Salah satu metode heuristik yang dapat dikombinasikan dengan algoritma genetika adalah *tabu search*. *Tabu search* merupakan salah satu metode pemecahan permasalahan optimasi kombinatorial yang tergabung ke dalam *local*

search methods. Metode ini bertujuan untuk mengefektifkan proses pencarian solusi terbaik dari suatu permasalahan optimasi kombinatorial yang berskala besar, contohnya permasalahan penjadwalan auditor, dengan waktu komputasi yang relatif lebih kecil, namun tanpa ada jaminan akan tercapainya solusi yang optimal [DAR-11].

Secara umum alur kombinasi algoritma genetika dan *tabu search* adalah sebagai berikut:

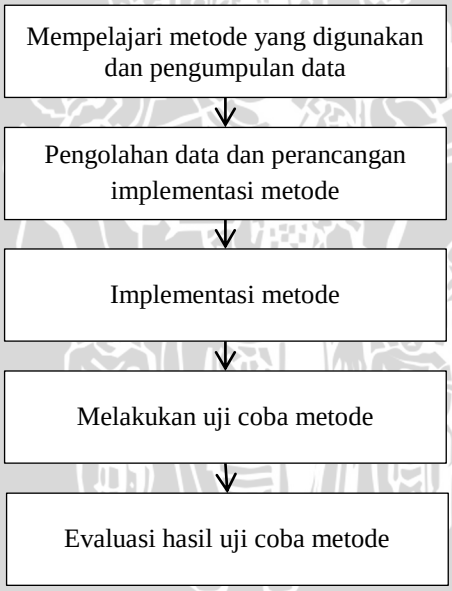


Gambar 2.2 *Flowchart* Penggabungan Algoritma Genetika dan *Tabu Search*
Sumber: [DAR-11]

BAB III
METODE PENELITIAN DAN PERANCANGAN

3.1 Metode Penelitian

Dalam melakukan penelitian, perlu adanya metode penelitian yang merupakan urutan langkah-langkah dan kerangka berpikir untuk merumuskan, menganalisis dan memecahkan masalah yang dihadapi. Alur pemikiran akan lebih terarah dan sistematis dengan adanya metode penelitian ini, sehingga tujuan dari penelitian tidak menyimpang dari permasalahan yang ada, serta memudahkan dalam proses analisis dan penarikan kesimpulan. Langkah penelitian ini digambarkan melalui blok diagram pada Gambar 3.1.



Gambar 3.1 Blok Diagram Penelitian

3.2.1 Mempelajari Metode yang Digunakan dan Pengumpulan Data

Tahap awal dalam penelitian ini adalah mempelajari metode yang digunakan serta pengambilan data. Metode yang digunakan pada penelitian ini adalah *hybrid genetic algorithm*, yang merupakan kombinasi dari algoritma genetika dan metode *tabu search*. Data yang digunakan merupakan data primer yang diambil dari Pusat Jaminan Mutu (PJM) Universitas Brawijaya. Adapun data

yang digunakan dalam penelitian ini adalah data auditor dan *auditee* pada proses Audit Internal Mutu (AIM) UKPPA Siklus 7.

3.2.2 Pengolahan Data dan Perancangan Implementasi Metode

Setelah dilakukan pengumpulan data, maka tahap selanjutnya adalah tahap pengolahan data. Data yang telah terkumpul selanjutnya akan diolah sehingga data siap untuk diproses dengan menerapkan metode yang digunakan. Untuk memperoleh data yang siap diolah, maka data yang diperoleh dikodekan terlebih dahulu. Selanjutnya adalah merancang implementasi metode yang digunakan dalam penjadwalan auditor. Pada tahap perancangan implementasi algoritma yang akan digunakan untuk penyusunan jadwal auditor, terdapat lima hal penting, yaitu: pengkodean kromosom, pembuatan populasi awal, proses regenerasi (mutasi, *crossover*), dan proses seleksi, serta pengulangan proses regenerasi dan seleksi.

3.2.3 Implementasi Metode

Tahap ini mengimplementasikan hasil perancangan metode yang digunakan serta sistem penjadwalan yang telah dibuat ke dalam komputer dengan menggunakan bahasa pemrograman dengan hasil akhir yang berupa program aplikasi.

3.2.4 Melakukan Uji Coba Metode

Pada tahap ini dilakukan uji coba dengan data-data penjadwalan auditor yang sudah didapat serta nilai-nilai parameter metode untuk mengetahui apakah metode yang digunakan telah memberikan hasil yang optimal. Hasil penjadwalan yang didapat kemudian dicocokkan dengan *constraint* yang digunakan sehingga tidak ada penjadwalan yang melanggar *constraint* khususnya *hard constraint*.

3.2.5 Evaluasi Hasil Uji Coba Metode

Tahap evaluasi hasil uji coba metode merupakan tahap terakhir pada penelitian. Pada tahap ini hasil *fitness* dari semua uji coba metode yang telah

dilakukan dievaluasi untuk mendapatkan nilai-nilai parameter yang terbaik dalam penjadwalan auditor.

3.2 Perancangan

Tahap perancangan yang akan dibangun pada penelitian ini meliputi tahap deskripsi umum sistem, perancangan perangkat lunak, perhitungan manual, dan juga perancangan antarmuka (*interface*).

3.2.1 Deskripsi Umum Sistem

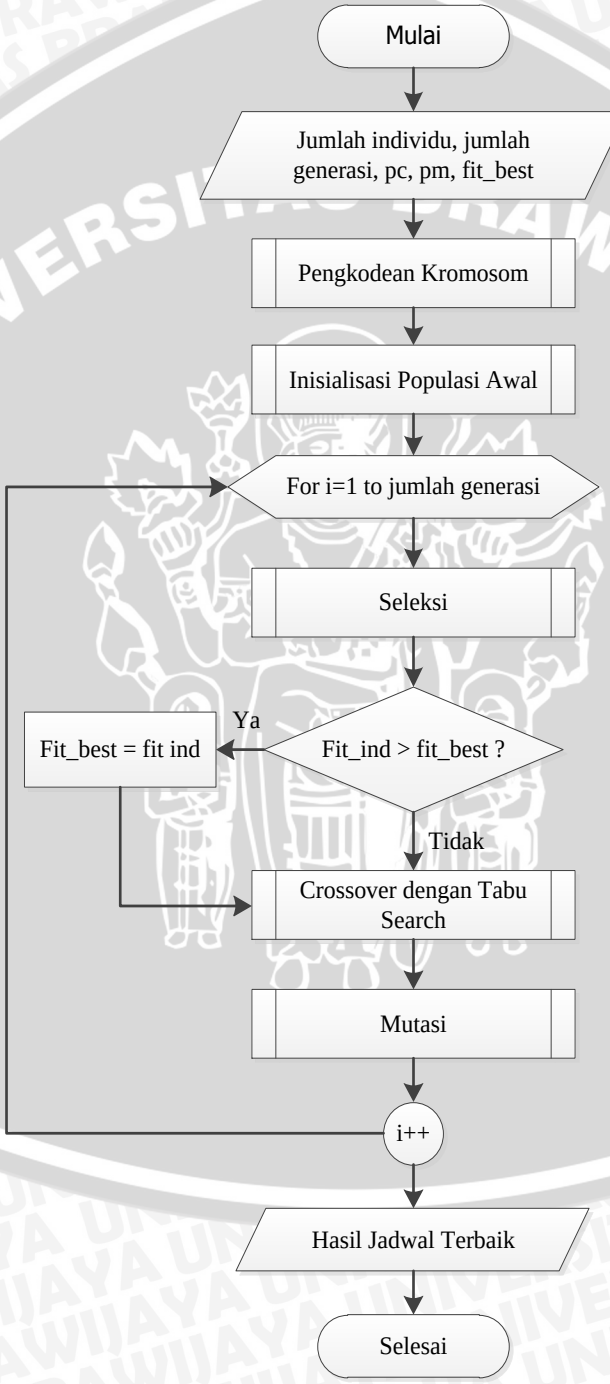
Secara umum, sistem yang akan dibangun berupa sebuah aplikasi perangkat lunak penjadwalan auditor yang nantinya digunakan untuk menyusun jadwal auditor secara otomatis. Hasil yang didapat kemudian dibandingkan untuk mengetahui efektivitas kinerja algoritma genetika dengan *tabu search*.

Proses penyusunan penjadwalan auditor AIM di PJM diawali dengan PJM meminta kesediaan dan alokasi waktu kepada auditor dan *auditee*. Setelah didapat daftar waktu kesediaan dari auditor dan *auditee*, selanjutnya dilakukan pengkodean data agar data dapat diproses di dalam sistem dengan mengimplementasikan algoritma genetika dan *tabu search* untuk mendapatkan hasil jadwal yang optimal. Jadwal yang dihasilkan oleh sistem dapat langsung disimpan dalam bentuk file CSV (*comma separated values*) untuk dapat digunakan lebih lanjut.

3.2.2 Perancangan Perangkat Lunak

Pada subbab ini akan dijelaskan mengenai perancangan perangkat lunak yang akan dibangun sesuai dengan deskripsi umum sistem yang telah dipaparkan sebelumnya. Aplikasi yang akan dibuat merupakan aplikasi berbasis desktop, dan bahasa pemrograman yang digunakan adalah java dengan IDE Netbeans serta menggunakan database MySQL sebagai media penyimpanan karena menggunakan file CSV sebagai data masukkannya. Data masukan tersebut kemudian diolah menggunakan metode algoritma genetika dan *tabu search* untuk mendapatkan hasil penjadwalan yang optimal. Parameter-parameter yang

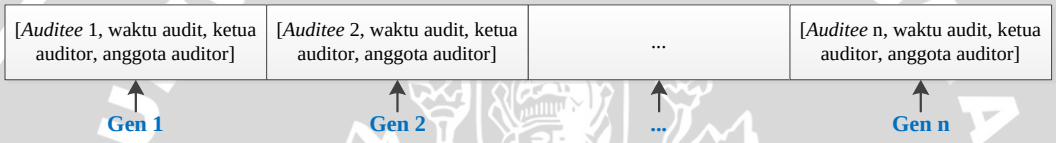
digunakan dalam proses algoritma genetika diantaranya adalah jumlah individu, jumlah generasi, nilai probabilitas mutasi, serta nilai probabilitas *crossover*. Gambar 3.2 di bawah ini adalah diagram alir secara umum proses algoritma genetika dan *tabu search* yang digunakan.



Gambar 3.2 *Flowchart* Algoritma Genetika menggunakan Tabu Search Pada Proses Penjadwalan Auditor

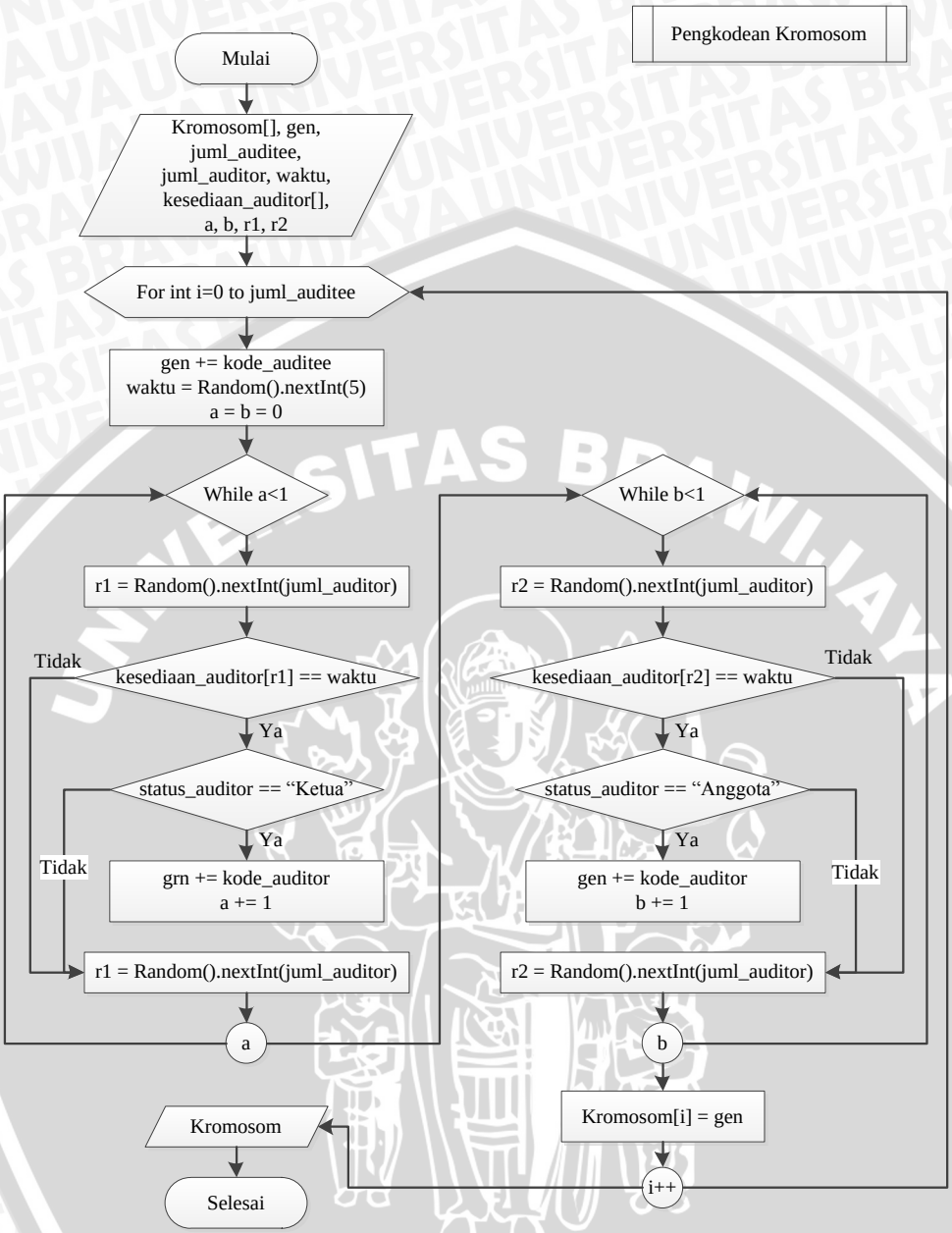
3.2.2.1 Proses Pengkodean Kromosom

Pengkodean kromosom yang akan digunakan pada penjadwalan auditor ini adalah pengkodean nilai. Inisialisasi kromosom direpresentasikan dalam bentuk larik (*array*) yang berisi data pendukung proses penjadwalan. Panjang dari kromosom adalah sebanyak gen yang ada, dalam hal ini setiap gen mewakili masing-masing komponen penjadwalan. Komponen yang digunakan yakni kode *auditee*, waktu audit, kode ketua tim auditor, serta kode anggota dari tim auditor. Jumlah kromosom yang dibentuk pada satu individu adalah sebanyak jumlah *auditee* karena fungsi objektif dari penjadwalan auditor adalah semua *auditee* terjadwalkan untuk proses audit.



Gambar 3.3 Komponen Pembentukan Kromosom

Proses pengkodean pada sub-gen pertama yakni kode *auditee* tidak dilakukan secara random, namun diambil langsung dari tabel data *auditee* sesuai dengan urutan dari kode *auditee* dan langsung dimasukkan dalam rancangan kromosom seperti pada Gambar 3.3. Untuk sub-gen kedua, pengkodeaan waktu audit diambil salah satu waktu secara random dari data kesediaan *auditee*. Kemudian untuk sub-gen ketiga diambil secara random dari tabel data auditor, dimana auditor yang terpilih memiliki waktu kesediaan yang sama dengan waktu audit terpilih (sub-gen kedua) dan status dari auditor adalah Ketua. Proses pembentukan sub-gen keempat sama dengan pembentukan sub-gen ketiga, namun dengan status auditor adalah Anggota. Diagram alir pembentukan kromosom yang digunakan pada sistem penjadwalan auditor dapat dilihat pada Gambar 3.4.



Gambar 3.4 *Flowchart* Pengkodean Kromosom Pada Proses Penjadwalan Auditor

Contoh hasil pengkodean kromosom dapat dilihat pada Tabel 3.5.

Tabel 3.1 Contoh Data Auditee

Kode *Auditee*	***Auditee***
E01	Fak. Ekonomi dan Bisnis (FEB)
E02	Fak. Teknik
E03	Prog, Teknologi Informasi dan Ilmu Komputer

E04	Fak. Ilmu Sosial dan Ilmu Politik
E05	Fak. Kedokteran

Tabel 3.2 Contoh Data Kesediaan Auditee

Kode *Auditee*	Kesediaan *Auditee*	Representasi Hari Kesediaan *Auditee*
E01	00101	3, 5
E02	01100	2, 3
E03	10010	1, 4
E04	10000	1
E05	01000	2

Tabel 3.3 Contoh Data Auditor

Kode Auditor	Nama Auditor	Status	Unit Kerja	Constraint
A01	Abdul Rouf Alghofari, Dr. Drs., MSc.	Ketua	Fak. MIPA	-
A02	Achmad Wicaksono, Ir., M.Eng., Ph.D	Ketua	Fak. Teknik	-
A03	Agoes Soehardjono MD, Prof. Dr. Ir., MS.	Anggota	Fak. Teknik	-
A04	Asih Purwanti, S.IP., M.IP	Anggota	FISIP	
A05	Coryna R. Amelia, S.ST.	Anggota	Fak. Kedokteran	-

Tabel 3.4 Contoh Data Kesediaan Auditor

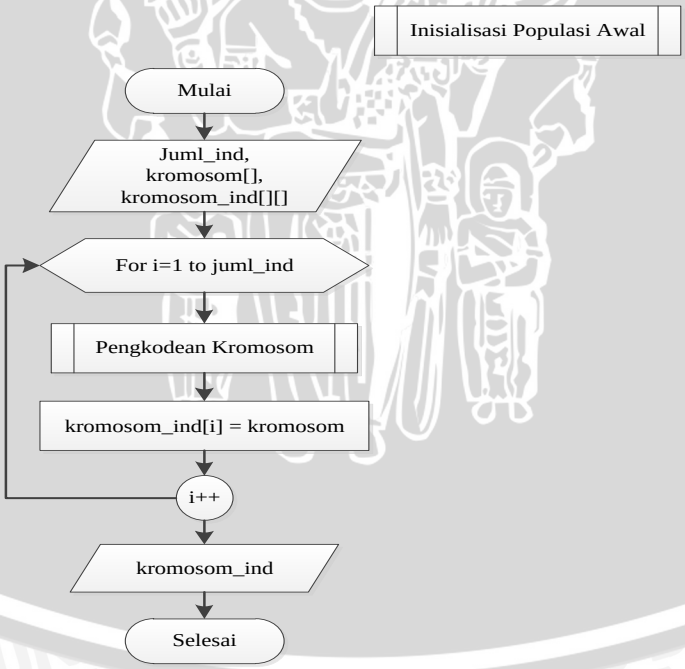
Kode Auditor	Kesediaan Auditor	Representasi Hari Kesediaan Auditor
A01	01101	2, 3, 5
A02	10110	1, 3 ,4
A03	11100	1, 2, 3
A04	00111	3, 4, 5
A05	01001	2, 5

Tabel 3.5 Contoh Hasil Pembentukan Kromosom Untuk 1 Individu

Kode *Auditee*	Waktu Audit	Kode Ketua Auditor	Kode Anggota Auditor	Gen
E01	3	A02	A04	E013A01A04
E02	2	A01	A05	E022A01A05
E03	1	A02	A03	E031A02A03
E04	1	A02	A03	E041A02A03
E05	2	A01	A05	E052A01A05
Kromosom: E013A01A04 E022A01A05 E031A02A03 E041A02A03 E052A01A05				

3.2.2.2 Inisialisasi Populasi Awal

Pada penelitian ini, satu individu adalah satu solusi jadwal. Sehingga inisialisasi atau pembentukan populasi awal adalah proses pembentukan individu atau pembentukan solusi jadwal sebanyak jumlah masukan yang telah ditentukan oleh pengguna. Diagram alir dari inisialisasi populasi awal dapat dilihat pada Gambar 3.5.



Gambar 3.5 *Flowchart* Proses Inisialisasi Populasi Awal

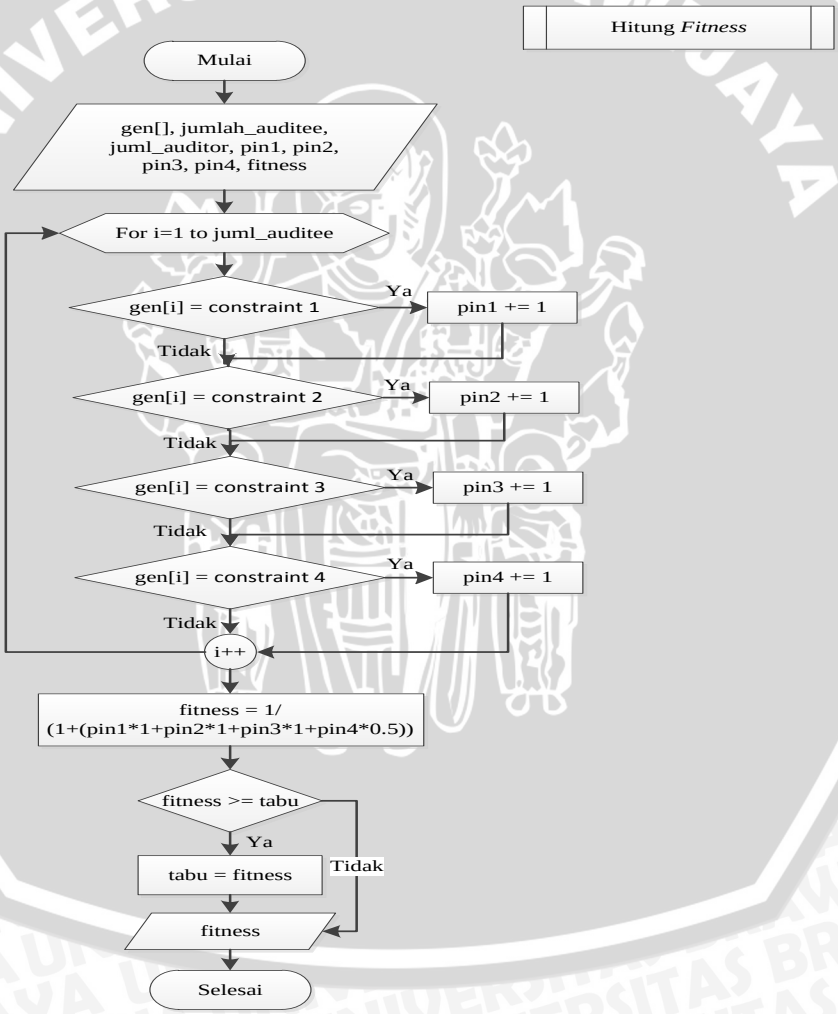
3.2.2.3 Perhitungan Nilai *Fitness*

Perhitungan nilai *fitness* dilakukan pada setiap individu atau solusi jadwal. Perhitungan dilakukan dengan memberikan pinalti untuk setiap aturan atau

constraint yang digunakan. Pelanggaran terhadap *hard constraint* akan dikenakan pinalti 1, sedangkan untuk pelanggaran terhadap *soft constraint* dikenakan pinalti 0.5. Berikut perhitungan fungsi *fitness* yang digunakan:

$$Fitness = 1/(1 + (aturan1 * pinalti1 + aturan2 * pinalti2 + \cdots)) \quad (3\text{-}1)$$

Dari fungsi *fitness* yang digunakan tersebut dapat disimpulkan bahwa semakin sedikit *constraint* yang dilanggar, maka semakin besar nilai *fitness*-nya dan semakin optimal pula jadwal yang dihasilkan. Diagram alir yang digunakan dalam perhitungan nilai *fitness* dapat dilihat pada Gambar 3.6.



Gambar 3.6 Flowchart Perhitungan Fitness Tiap Individu

Nilai pinalti yang digunakan pada setiap pelanggaran *constraint* dapat dilihat pada Tabel 3.6.

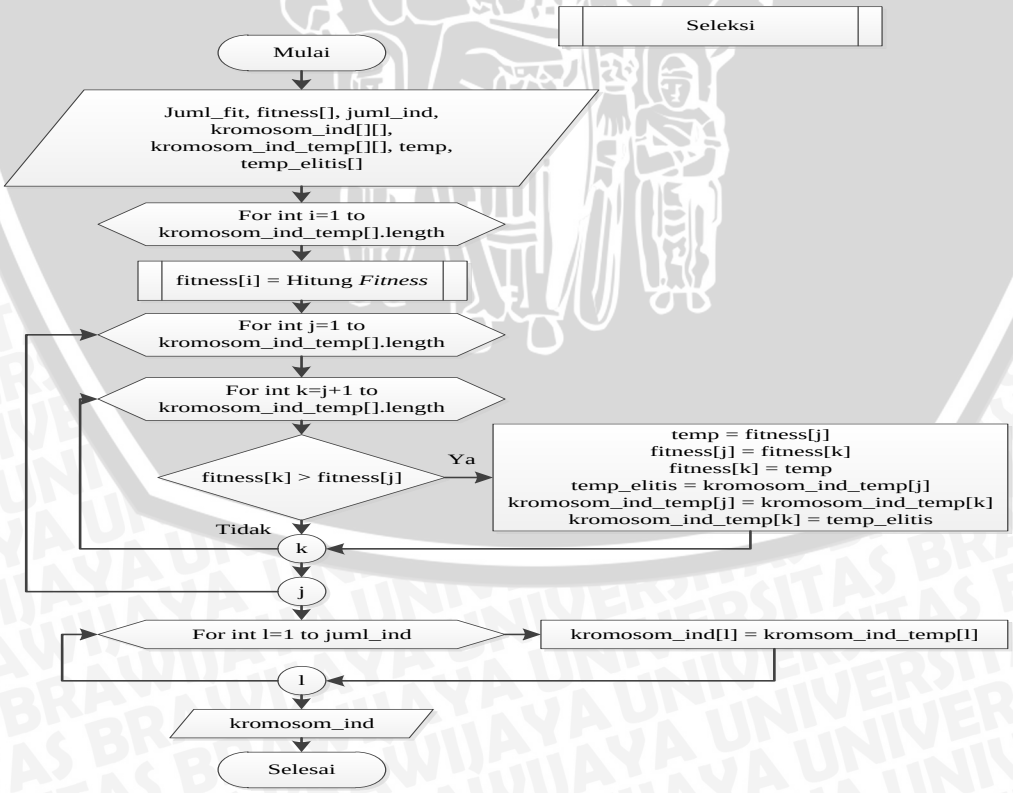
Tabel 3.6 Nilai Pinalti untuk Setiap Pelanggaran Constraint

No	*Constraint*	Pinalti
1	Auditor tidak boleh dari unit kerja yg sama dengan *auditee*	1
2	Ketua dan Anggota tidak boleh dari unit kerja yg sama	1
3	Auditor tidak dapat mengaudit lebih dari 1 kali sehari	1
4	Auditor sebaiknya hanya melakukan audit maksimal 3 kali dalam satu siklus	0.5

Penggunaan *constraint* nomor 1 dan nomor 2 sebagai *hard constraint* bertujuan untuk menjaga keobjektifan dari proses audit. Untuk *constraint* nomor 3 digunakan agar tidak terdapat bentrok jadwal dari semua auditor. Sedangkan *constraint* nomor 4 ditujukan agar sebaran auditor merata untuk masing-masing auditor.

3.2.2.4 Proses Seleksi

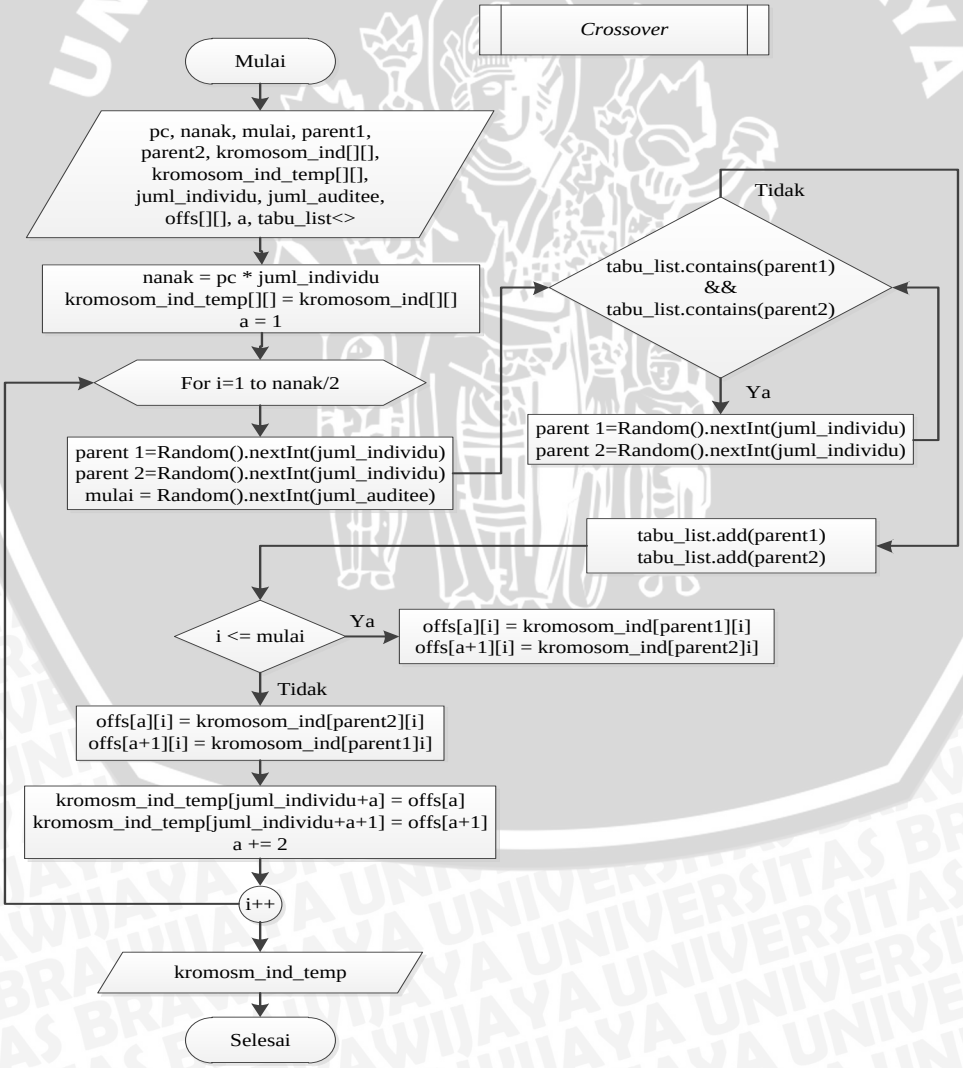
Proses pembuatan populasi baru merupakan proses penting dalam algoritma genetika, karena pada proses ini akan dipilih induk yang digunakan untuk menghasilkan populasi baru. Proses seleksi dalam pemilihan induk yang digunakan pada penelitian ini adalah seleksi elitis.



Gambar 3.7 Proses Seleksi dengan Metode Elitis

3.2.2.5 Proses *Crossover* dengan *Tabu Search*

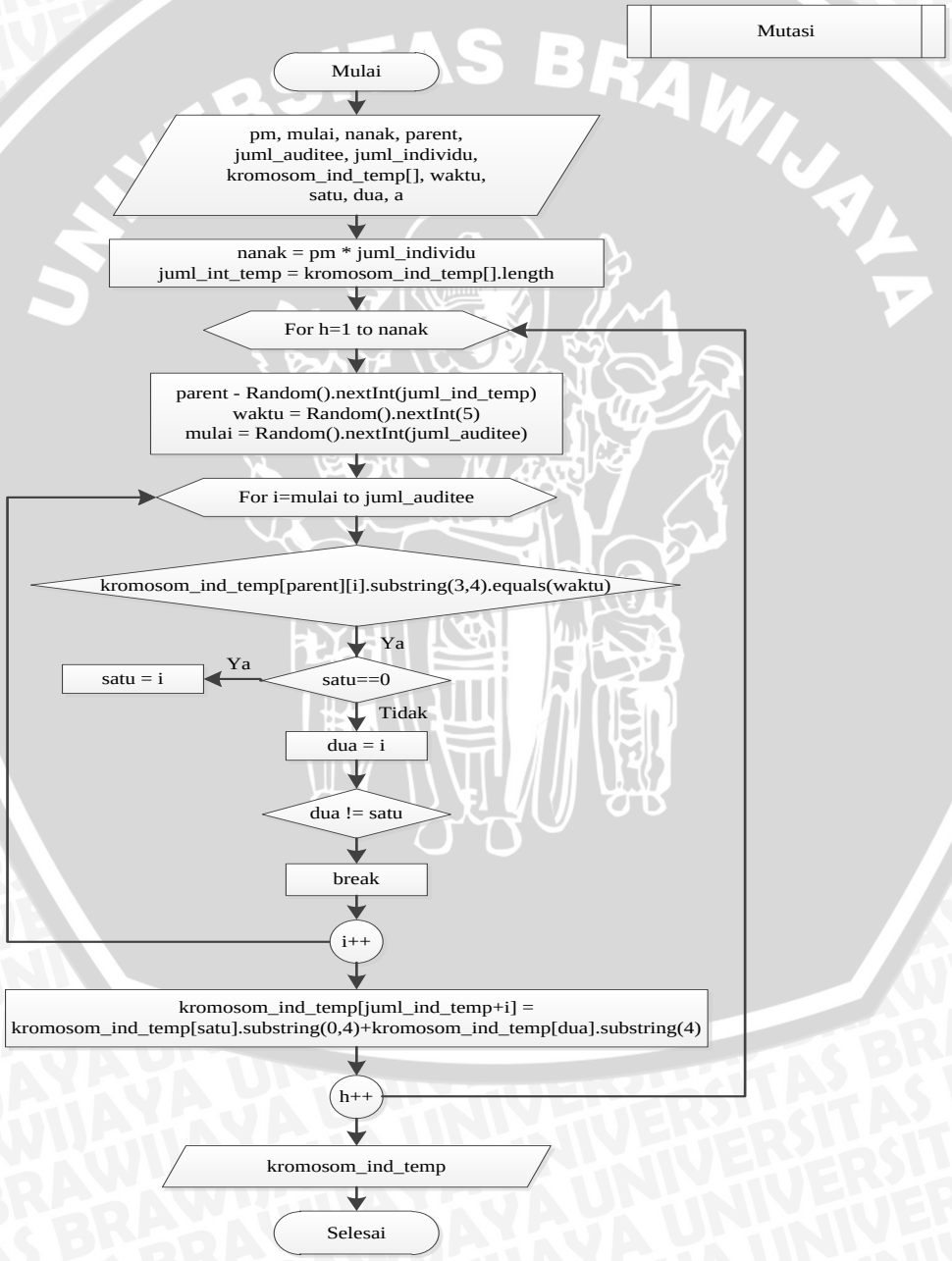
Proses *crossover* merupakan proses pertukaran atau pindah silang kromosom dari dua induk tertentu dengan tujuan untuk menghasilkan keturunan baru. Proses *crossover* yang digunakan pada penelitian ini adalah *crossover* satu titik (*one cut point*). Pada *crossover* ini dilakukan dengan cara menukar nilai gen-gen pada posisi kromosom yang sama dari kedua induk. Jumlah anak yang dibentuk sesuai dengan nilai dari probabilitas *crossover* yang digunakan. *Tabu search* pada proses *crossover* ini digunakan untuk memfilter kromosom induk yang akan mengalami *crossover* agar kromosom yang sama tidak dilakukan *crossover* berulang-ulang. Diagram alir proses *crossover* pada sistem penjadwalan auditor dapat dilihat pada Gambar 3.8.



Gambar 3.8 Flowchart Proses *Crossover* dengan *Tabu Search*

3.2.2.6 Proses Mutasi

Proses mutasi adalah proses perubahan materi genetika (gen atau kromosom) yang terpilih secara acak dengan tujuan untuk mencegah terjadinya konvergensi prematur. Proses mutasi dilakukan dengan cara menukar nilai gen-gen pada posisi kromosom tertentu yang memiliki waktu audit yang sama. Jumlah anak yang dibentuk sesuai dengan nilai dari probabilitas mutasi yang digunakan. Diagram alir proses mutasi yang digunakan dapat dilihat pada Gambar 3.9.



Gambar 3.9 Flowchart Proses Mutasi

3.2.3 Contoh Perhitungan Manual

Tahap pertama dalam proses penjadwalan auditor menggunakan algoritma genetika dan *tabu search* berdasarkan diagram alir yang ditunjukkan pada Gambar 3.2 adalah pembentukan populasi awal (pembentukan solusi jadwal) secara acak. Data auditor, *auditee*, kesediaan auditor dan kesediaan *auditee* yang digunakan adalah data pada Tabel 3.1 sampai dengan Tabel 3.4.

Generasi Awal

Misalkan jumlah individu (kromosom) yang digunakan adalah 5, maka hasil proses inisialisasi populasi awalnya dapat dilihat pada Tabel 3.7 sampai dengan Tabel 3.11.

Tabel 3.7 Individu 1

Kode *Auditee*	Waktu Audit	Kode Ketua Auditor	Kode Anggota Auditor	Gen
E01	3	A01	A04	E013A01A04
E02	2	A01	A05	E022A01A05
E03	1	A02	A03	E031A02A03
E04	1	A02	A03	E041A02A03
E05	2	A01	A05	E052A01A05
Kromosom: E013A01A04 E022A01A05 E031A02A03 E041A02A03 E052A01A05				

Tabel 3.8 Individu 2

Kode *Auditee*	Waktu Audit	Kode Ketua Auditor	Kode Anggota Auditor	Gen
E01	3	A01	A04	E013A01A04
E02	3	A01	A05	E023A01A05
E03	1	A02	A03	E031A02A03
E04	1	A02	A03	E041A02A03
E05	2	A01	A05	E052A01A05
Kromosom: E013A01A04 E023A01A05 E031A02A03 E041A02A03 E052A01A05				

Tabel 3.9 Individu 3

Kode *Auditee*	Waktu Audit	Kode Ketua Auditor	Kode Anggota Auditor	Gen
E01	5	A01	A05	E015A01A05
E02	3	A02	A03	E023A02A03

E03	4	A02	A04	E034A02A04
E04	1	A02	A03	E041A02A03
E05	2	A01	A05	E052A01A05
Kromosom: E015A01A05 E023A02A03 E034A02A04 E041A02A03 E052A01A05				

Tabel 3.10 Individu 4

Kode *Auditee*	**Waktu Audit**	**Kode Ketua Auditor**	**Kode Anggota Auditor**	**Gen**
E01	5	A01	A05	E015A01A05
E02	2	A01	A03	E022A01A03
E03	4	A02	A04	E034A02A04
E04	1	A02	A03	E041A02A03
E05	2	A01	A05	E052A01A05
Kromosom: E015A01A05 E022A01A03 E034A02A04 E041A02A03 E052A01A05				

Tabel 3.11 Individu 5

Kode *Auditee*	**Waktu Audit**	**Kode Ketua Auditor**	**Kode Anggota Auditor**	**Gen**
E01	3	A02	A04	E013A02A04
E02	2	A01	A05	E022A01A05
E03	4	A02	A04	E034A02A04
E04	1	A02	A03	E041A02A03
E05	2	A01	A03	E052A01A03
Kromosom: E013A02A04 E022A01A05 E034A02A04 E041A02A03 E052A01A03				

Tahap kedua adalah menghitung nilai *fitness* dari masing-masing individu hasil pembentukan populasi awal. Perhitungan *fitness* untuk hasil pembentukan populasi awal tiap individu pada Tabel 3.7 sampai Tabel 3.11 adalah sebagai berikut:

Tabel 3.12 *Fitness* Individu 1

No	Gen	Pelanggaran *Constraint* 1	Pelanggaran *Constraint* 2	Pelanggaran *Constraint* 3
1	E013A01A04	0	0	0
2	E022A01A05	0	0	dengan gen 5
3	E031A02A03	0	1	dengan gen 4
4	E041A02A03	0	1	dengan gen 3
5	E052A01A05	1	0	dengan gen 2

Σ Pinalti 1 = 1 Σ Pinalti 3 = 2

Σ Pinalti 2 = 2 Σ Pinalti 4 = 1 (pada auditor A01)

$$Fitness\ individu\ 1 = \frac{1}{1 + (1 * 1 + 2 * 1 + 2 * 1 + 1 * 0.5)} = 0.15384$$

Tabel 3.13 *Fitness* Individu 2

No	Gen	Pelanggaran *Constraint* 1	Pelanggaran *Constraint* 2	Pelanggaran *Constraint* 3
1	E013A01A04	0	0	dengan gen 2
2	E023A01A05	0	0	dengan gen 1
3	E031A02A03	0	1	dengan gen 4
4	E041A02A03	0	1	dengan gen 3
5	E052A01A05	1	0	0

Σ Pinalti 1 = 1 Σ Pinalti 3 = 2

Σ Pinalti 2 = 2 Σ Pinalti 4 = 1 (pada auditor A01)

$$Fitness\ individu\ 2 = \frac{1}{1 + (1 * 1 + 2 * 1 + 2 * 1 + 1 * 0.5)} = 0.15384$$

Tabel 3.14 *Fitness* Individu 3

No	Gen	Pelanggaran *Constraint* 1	Pelanggaran *Constraint* 2	Pelanggaran *Constraint* 3
1	E015A01A05	0	0	0
2	E023A02A03	1	1	0
3	E034A02A04	0	0	0
4	E041A02A03	0	1	0
5	E052A01A05	1	0	0

Σ Pinalti 1 = 2 Σ Pinalti 3 = 0

Σ Pinalti 2 = 2 Σ Pinalti 4 = 1 (pada auditor A02)

$$Fitness\ individu\ 3 = \frac{1}{1 + (2 * 1 + 2 * 1 + 0 * 1 + 1 * 0.5)} = 0.18181$$

Tabel 3.15 *Fitness* Individu 4

No	Gen	Pelanggaran *Constraint* 1	Pelanggaran *Constraint* 2	Pelanggaran *Constraint* 3
1	E015A01A05	0	0	0
2	E022A01A03	1	0	dengan gen 5

3	E034A02A04	0	0	0
4	E041A02A03	0	1	0
5	E052A01A05	1	0	dengan gen 2

Σ Pinalti 1 = 2 Σ Pinalti 3 = 1

Σ Pinalti 2 = 1 Σ Pinalti 4 = 1 (pada auditor A01)

$$Fitness\ individu\ 4 = \frac{1}{1 + (2 * 1 + 1 * 1 + 1 * 1 + 1 * 0.5)} = 0.18181$$

Tabel 3.16 *Fitness* Individu 5

No	Gen	Pelanggaran *Constraint* 1	Pelanggaran *Constraint* 2	Pelanggaran *Constraint* 3
1	E013A02A04	0	0	0
2	E022A01A05	0	0	dengan gen 5
3	E034A02A04	0	0	0
4	E041A02A03	0	1	0
5	E052A01A03	0	0	dengan gen 2

Σ Pinalti 1 = 0 Σ Pinalti 3 = 1

Σ Pinalti 2 = 1 Σ Pinalti 4 = 1 (pada auditor A02)

$$Fitness\ individu\ 5 = \frac{1}{1 + (0 * 1 + 1 * 1 + 1 * 1 + 1 * 0.5)} = 0.28571$$

Setelah didapat nilai *fitness* dari masing-masing individu, tahap selanjutnya adalah menentukan individu terbaik untuk dimasukkan ke dalam variabel *best*. Pada generasi awal, variabel *best* masih kosong sehingga individu dengan nilai *fitness* terbaik dapat langsung dimasukkan dalam *best*. Individu yang memiliki nilai *fitness* terbaik adalah individu ke-5 dengan nilai *fitness* 0.28571.

Best = Individu ke 5

Fitness = 0.28571

Tahap keempat dalam algoritma genetika dengan *tabu search* adalah melakukan seleksi. Langkah awal pada proses seleksi ini adalah menghitung nilai peluang kumulatif dari masing-masing individu.

$$\Sigma fitness = fitness\ individu1 + fitness\ individu2 + fitness\ individu3 + fitness\ individu4 + fitness\ individu5$$
$$= 0.15384 + 0.15384 + 0.18181 + 0.18181 + 0.28571$$
$$= 0.95701$$

Tabel 3.17 Peluang Kumulatif Generasi Awal

Individu (Kromosom)	Fitness	Peluang	Peluang Kumulatif
1	0.15384	0.16075	0.16075
2	0.15384	0.16075	0.3215
3	0.18181	0.18998	0.51148
4	0.18181	0.18998	0.70146
5	0.28571	0.29854	1

Pada generasi pertama, tidak dilakukan seleksi karena belum terjadi proses rekombinasi individu.

Tahap kelima adalah proses *crossover*. Pada tahap ini akan dibentuk individu-individu baru hasil kombinasi antara *parent-parent* yang ditentukan secara random dengan harapan terbentuk individu dengan nilai *fitness* baik. *Parent-parent* yang terpilih terlebih dahulu akan diseleksi, apakah terdapat dalam tabu_list atau tidak untuk menghindari terjadinya proses *crossover* pada kromosom yang sama. Jumlah individu baru yang dibentuk ditentukan dari probabilitas *crossover* yang digunakan. Berikut langkah-langkah proses *crossover* dengan menggunakan *tabu search*:

tabu_list: <kosong>

- Ditentukan probabilitas *crossover* (pc) = 0.4
- Jumlah anak = 0.4 x 5 = 2 anak (*offspring*)
- *Generate parent* 1 dan *parent* 2 secara random. Misalkan didapat individu 5 dan individu 3.
- Cek apakah individu 5 dan individu 3 terdapat pada tabu_list.
- Individu 5 dan individu 3 tidak terdapat pada tabu_list sehingga data *parent* dapat dimasukkan ke dalam tabu_list.

tabu_list: <5,3>

- *Generate* titik awal proses *crossover* = 2
  (Maka titik *crossover* adalah gen ke 2)

Parent 1 (Individu 5)

No.	Gen
1.	E013A02004
2.	E022A01005
3.	E034A02004
4.	E041A02003
5.	E052A01003

Parent 2 (Individu 3)

No.	Gen
1.	E015A01A05
2.	E023A02A03
3.	E034A02A04
4.	E041A02A03
5.	E052A01A05

Tabel 3.18 *Offspring* 1

No.	Gen
1.	E013A02A04
2.	E022A01A05
3.	E034A02A04
4.	E041A02A03
5.	E052A01A05

Tabel 3.19 *Offspring* 2

No.	Gen
1.	E015A01A05
2.	E023A02A03
3.	E034A02A04
4.	E041A02A03
5.	E052A01A03

Tahap keenam adalah tahap rekombinasi atau proses mutasi. Sama halnya dengan proses *crossover*, *parent* yang digunakan pada proses mutasi ditentukan secara random. Jumlah anak atau *offspring* dari proses mutasi ditentukan dari nilai probabilitas mutasi yang digunakan. Berikut langkah-langkah dari proses mutasi:

- Ditentukan probabilitas mutasi (pm) = 0.2
- Jumlah anak = 0.2 x 5 = 1 anak (*offspring*)
- *Generate parent* random. Misalkan didapat individu 4 sebagai *parent*.
- Generate titik awal proses mutasi. Misalkan didapat titik = 2 (mulai dari gen 2)
- Cari gen pertama setelah gen-2 yang memiliki sub-gen waktu sama dengan gen-2. Didapat gen ke-5 = E052A01A05
- Tukar sub-gen ke-3 dan ke-4 dari kedua gen

No.	Gen
1.	E015A01A05
2.	E022A01A03

3.	E034A02A04
4.	E041A02A03
5.	E052A01A05

Tabel 3.20 Hasil Mutasi *Offspring* 3

No.	Gen
1.	E015A01A05
2.	E022A01A05
3.	E034A02A04
4.	E041A02A03
5.	E052A01A03

Generasi 1

Tahap awal pada generasi pertama adalah menghitung nilai *fitness offspring-offspring* hasil bentukan pada generasi sebelumnya. Setelah didapat nilai *fitness*, maka dapat dilakukan tahap seleksi dan tahap-tahap selanjutnya seperti pada generasi awal.

Tabel 3.21 *Fitness Offspring* 1

No	Gen	Pelanggaran *Constraint* 1	Pelanggaran *Constraint* 2	Pelanggaran *Constraint* 3
1	E013A02A04	0	0	0
2	E022A01A05	0	0	dengan gen 5
3	E034A02A04	0	0	0
4	E041A02A03	0	1	0
5	E052A01A05	1	0	dengan gen 2

Σ Pinalti 1 = 1 Σ Pinalti 3 = 1

Σ Pinalti 2 = 1 Σ Pinalti 4 = 1 (pada auditor A02)

$$Fitness\ offspring\ 1 = \frac{1}{1 + (1 * 1 + 1 * 1 + 1 * 1 + 1 * 0.5)} = 0.22222$$

Tabel 3.22 *Fitness Offspring* 2

No	Gen	Pelanggaran *Constraint* 1	Pelanggaran *Constraint* 2	Pelanggaran *Constraint* 3
1	E015A01A05	0	0	0

2	E023A02A03	1	1	0
3	E034A02A04	0	0	0
4	E041A02A03	0	1	0
5	E052A01A03	0	0	0

Σ Pinalti 1 = 1 Σ Pinalti 3 = 0

Σ Pinalti 2 = 2 Σ Pinalti 4 = 1 (pada auditor A02)

$$Fitness\ offspring\ 2 = \frac{1}{1 + (1 * 1 + 2 * 1 + 0 * 1 + 1 * 0.5)} = 0.22222$$

Tabel 3.23 Fitness *Offspring* 3

No	Gen	Pelanggaran *Constraint* 1	Pelanggaran *Constraint* 2	Pelanggaran *Constraint* 3
1	E015A01A05	0	0	0
2	E022A01A05	0	0	dengan gen 5
3	E034A02A04	0	0	0
4	E041A02A03	0	1	0
5	E052A01A03	0	0	dengan gen 2

Σ Pinalti 1 = 0 Σ Pinalti 3 = 1

Σ Pinalti 2 = 1 Σ Pinalti 4 = 1 (pada auditor A01)

$$Fitness\ offspring\ 3 = \frac{1}{1 + (0 * 1 + 1 * 1 + 1 * 1 + 1 * 0.5)} = 0.28571$$

Berikut nilai fitness dari semua individu dan *offspring* yang telah terbentuk.

Tabel 3.24 Fitness Populasi Generasi 1

Individu	***Fitness***
1	0.15384
2	0.15384
3	0.18181
4	0.18181
5	0.28571
Offspring 1	0.22222
Offspring 2	0.22222
Offspring 3	0.28571

Nilai *fitness* individu dari pembentukan populasi baru tidak ada yang lebih besar dari nilai *fitness* pada *best*, sehingga isi *best* tidak mengalami perubahan.

Best = Individu ke 5

Fitness = 0.28571

Tahap selanjutnya adalah proses seleksi menggunakan metode elitis. Proses seleksi adalah dengan cara mengurutkan individu yang memilliki *fitness* tertinggi hingga *fitness* terendah, setelah itu ambil sebanyak 5 individu (sejumlah inisialisasi jumlah individu) dengan urutan *fitness* terbaik untuk dijadikan sebagai populasi baru dalam proses generasi selanjutnya.

Tabel 3.25 Hasil Pengurutan Individu Terhadap Nilai **Fitness** Terbaik

Individu	***Fitness***
5	0.28571
Offspring 3	0.28571
Offspring 1	0.22222
Offspring 2	0.22222
3	0.18181
4	0.18181
1	0.15384
2	0.15384

Tabel 3.26 Populasi Baru Generasi 1

Individu (Kromosom)	Fitness
1	0.28571
2	0.28571
3	0.22222
4	0.22222
5	0.18181

Tahap kelima adalah proses *crossover*. Pada tahap ini akan dibentuk individu-individu baru hasil kombinasi antara *parent-parent* yang ditentukan secara random dengan harapan terbentuk individu dengan nilai *fitness* baik. *Parent-parent* yang terpilih terlebih dahulu akan diseleksi, apakah terdapat dalam tabu_list atau tidak untuk menghindari terjadinya proses *crossover* pada

kromosom yang sama. Jumlah individu baru yang dibentuk ditentukan dari probabilitas *crossover* yang digunakan. Berikut langkah-langkah proses *crossover* dengan menggunakan *tabu search*:

tabu_list: <kosong>

- Ditentukan probabilitas *crossover* (pc) = 0.4
- Jumlah anak = 0.4 x 5 = 2 anak (*offspring*)
- *Generate parent* 1 dan *parent* 2 secara random. Misalkan didapat individu 1 dan individu 2.
- Cek apakah individu 1 dan individu 2 terdapat pada tabu_list.
- Individu 1 dan individu 2 tidak terdapat pada tabu_list sehingga data *parent* dapat dimasukkan ke dalam tabu_list.

tabu_list: <1,2>

- *Generate* titik awal proses *crossover* = 3
  (Maka titik *crossover* adalah gen ke 3)

Parent 1 (Individu 1)

No.	Gen
1.	E013A02A04
2.	E022A01A05
3.	E034A02A04
4.	E041A02A03
5.	E052A01A03

Parent 2 (Individu 2)

No.	Gen
1.	E015A01A05
2.	E022A01A05
3.	E034A02A04
4.	E041A02A03
5.	E052A01A03

Tabel 3.27 *Offspring* 1 Generasi 1

No.	Gen
1.	E013A02A04
2.	E022A01A05

3.	E034A02A04
4.	E041A02A03
5.	E052A01A03

Tabel 3.28 *Offspring* 2 Generasi 1

No.	Gen
1.	E015A01A05
2.	E022A01A03
3.	E034A02A04
4.	E041A02A03
5.	E052A01A03

Tahap keenam adalah tahap rekombinasi atau proses mutasi. Sama halnya dengan proses *crossover*, *parent* yang digunakan pada proses mutasi ditentukan secara random. Jumlah anak atau *offspring* dari proses mutasi ditentukan dari nilai probabilitas mutasi yang digunakan. Berikut langkah-langkah dari proses mutasi:

- Ditentukan probabilitas mutasi (pm) = 0.2
- Jumlah anak = 0.2 x 5 = 1 anak (*offspring*)
- *Generate parent* random. Misalkan didapat individu 2 sebagai *parent*.
- Generate titik awal proses mutasi. Misalkan didapat titik = 2 (mulai dari gen 1)
- Cari gen pertama setelah gen-1 yang memiliki sub-gen waktu sama dengan gen-1. Didapat gen ke-5 = E052A01A03
- Tukar sub-gen ke-3 dan sub-gen ke-4 dari kedua gen

No.	Gen
1.	E015A01A05
2.	E022A01A05
3.	E034A02A04
4.	E041A02A03
5.	E052A01A03

Tabel 3.29 Hasil Mutasi *Offspring* 3

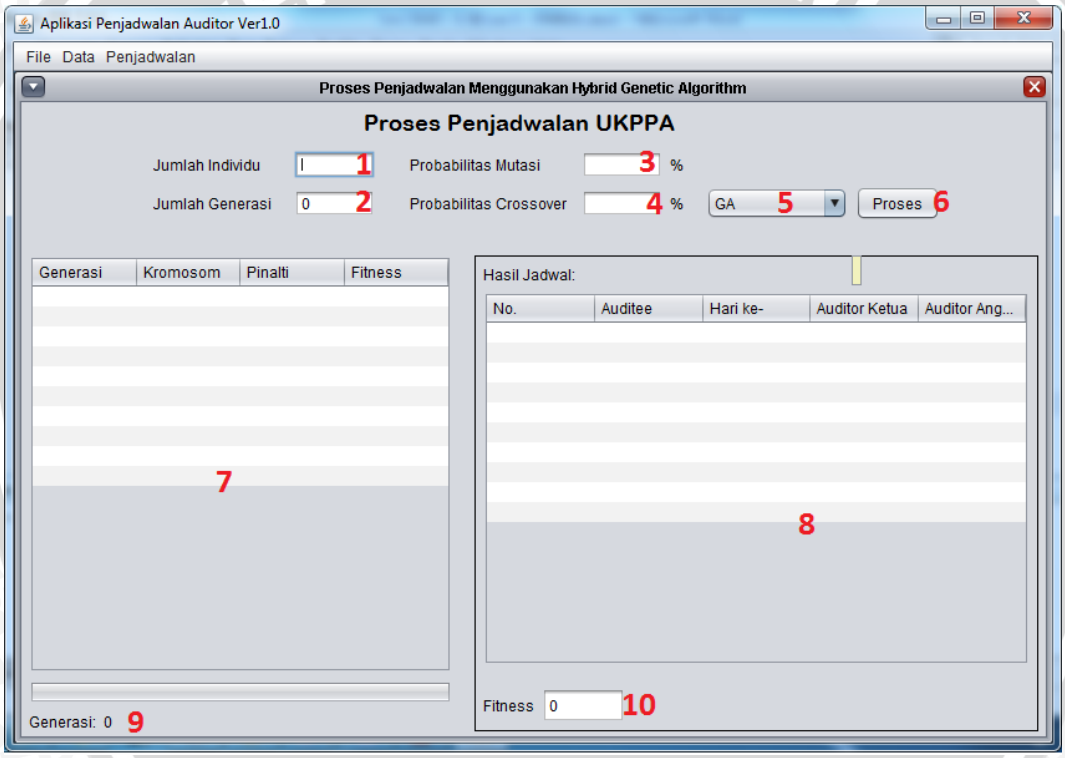
No.	Gen
1.	E015A01A05
2.	E022A01A03

3.	E034A02A04
4.	E041A02A03
5.	E052A01A05

Proses perulangan akan terus dilanjutkan hingga didapat nilai fitness 1, yakni tidak ada *constraint* yang terlanggar.

3.2.4 Perancangan Antar Muka

Rancangan antar muka dari aplikasi penjadwalan auditor dapat dilihat pada Gambar 3.10.



Gambar 3.10 Tampilan Proses Penjadwalan Auditor

Keterangan:

1. Inputan parameter jumlah individu
2. Inputan parameter jumlah generasi
3. Inputan parameter nilai probabilitas mutasi (pm)
4. Inputan parameter nilai probabilitas *crossover* (pc)

5. Pemilihan jenis algoritma yang akan digunakan, Algoritma Genetika atau Algoritma Genetika dengan Tabu Search
6. Tombol mulai proses penjadwalan
7. Tabel nilai *fitness* dari semua generasi hasil proses algoritma genetika
8. Tabel hasil jadwal terbaik dari proses penjadwalan menggunakan Algoritma Genetika atau Algoritma Genetika dengan Tabu Search
9. Jumlah generasi yang dibutuhkan sampai kondisi berhenti terpenuhi
10. Nilai *fitness* terbaik dari semua generasi
11. Menu "File" digunakan untuk membuka sub-menu "Exit" atau keluar dari aplikasi
12. Menu "Data" digunakan untuk membuka sub-menu:
    - "Data Auditor" untuk melihat, menambah, mengubah atau menghapus data auditor
    - "Data Auditee" untuk melihat, menambah, mengubah atau menghapus data auditee
    - "Data Kesediaan Auditor" untuk melihat, menambah, mengubah atau menghapus data kesediaan auditor
    - "Data Kesediaan Auditee" untuk melihat, menambah, mengubah atau menghapus data kesediaan *auditee*

3.2.5 Skenario Pengujian

Penelitian ini memiliki tujuan untuk membandingkan kinerja dari Algoritma Genetika murni dengan Hybrid Algoritma Genetika (dalam penelitian ini menggunakan penggabungan Algoritma Genetika dan *Tabu Search*). Uji coba pada penelitian ini menggunakan 3 skenario, dimana masing-masing skenario memiliki tujuan untuk mengukur kinerja masing-masing algoritma berdasarkan parameter yang digunakan. Pengujian kinerja didasarkan pada jumlah generasi yang dibutuhkan algoritma untuk mencapai nilai *fitness* 1. Pada masing-masing skenario dilakukan 20 kali percobaan dengan harapan dicapainya hasil yang konvergen pada 20 percobaan tersebut. Berikut penjelasan masing-masing skenario yang digunakan:

1 Uji kinerja algoritma dengan jumlah individu (jumlah populasi) sebagai pembanding. Jumlah individu yang digunakan adalah 25, 50, 75, dan 100 dengan masing-masing 20 kali uji coba. Pada pengujian ini menggunakan parameter probabilitas *crossover* 0.5, dan probabilitas mutasi 0.5. Perancangan pengujian pada skenario 1 dapat dilihat pada Tabel 3.30.

Tabel 3.30 Skenario Uji Coba Pengaruh Jumlah Individu Pada Proses Penjadwalan UKPPA

Percobaan ke-	**Jumlah Individu**							
	25		**50**		**75**		**100**	
	GA	**GA + Tabu**	**GA**	**GA + Tabu**	**GA**	**GA + Tabu**	**GA**	**GA + Tabu**
1								
2								
3								
4								
5								
6								
7								
8								
…								
20								

2 Uji kinerja algoritma dengan pembanding probabilitas *crossover*. Kombinasi pengujian yang digunakan adalah menggunakan probabilitas mutasi 0.5 dengan nilai probabilitas *crossover* pada 0.3, 0.5, 0.7, dan 0.9 dengan masing-masing 20 kali uji coba. Pada pengujian ini, jumlah individu yang digunakan adalah 50. Perancangan pengujian pada skenario 2 dapat dilihat pada Tabel 3.31.

Tabel 3.31 Skenario Uji Coba Pengaruh Probabilitas *Crossover* Pada Proses Penjadwalan UKPPA

Percobaan ke-	**Probabilitas *Crossover***							
	0.3		**0.5**		**0.7**		**0.9**	
	GA	**GA + Tabu**	**GA**	**GA + Tabu**	**GA**	**GA + Tabu**	**GA**	**GA + Tabu**
1								

2								
3								
4								
5								
6								
7								
8								
…								
20								

3 Uji kinerja algoritma dengan pembanding probabilitas mutasi. Kombinasi pengujian yang digunakan adalah menggunakan probabilitas *crossover* 0.5 dengan nilai probabilitas mutasi pada 0.3, 0.5, 0.7, dan 0.9 dengan masing-masing 20 kali uji coba. Pada pengujian ini, jumlah individu yang digunakan adalah 50. Perancangan pengujian pada skenario 3 dapat dilihat pada Tabel 3.32.

Tabel 3.32 Skenario Uji Coba Pengaruh Probabilitas Mutasi Pada Proses Penjadwalan UKPPA

Percobaan ke-	**Probabilitas Mutasi**							
	0.3		**0.5**		**0.7**		**0.9**	
	GA	**GA + Tabu**	**GA**	**GA + Tabu**	**GA**	**GA + Tabu**	**GA**	**GA + Tabu**
1								
2								
3								
4								
5								
6								
7								
8								
…								
20								

BAB IV
IMPLEMENTASI

4.1 Lingkungan Implementasi

Lingkungan implementasi yang dijelaskan dalam sub bab ini merupakan lingkungan implementasi perangkat lunak dan perangkat keras yang digunakan dalam proses implementasi algoritma yang telah dibuat dalam penelitian ini.

4.2.1 Lingkungan Perangkat Keras

Spesifikasi dari perangkat keras yang digunakan dalam pengembangan sistem penjadwalan auditor menggunakan *hybrid* algoritma genetika adalah sebagai berikut:

1. Prosesor Intel® Atom™ CPU N2800 @ 1.86GHz (2 CPUs)
2. Memori 2GB
3. Harddisk dengan kapasitas 320GB
4. Monitor 10”
5. Keyboard

4.2.2 Lingkungan Perangkat Lunak

Spesifikasi dari perangkat lunak yang digunakan dalam pengembangan sistem penjadwalan auditor menggunakan *hybrid* algoritma genetika adalah sebagai berikut:

1. Sistem Operasi Windows 7 Ultimate 32 bit
2. NetBeans IDE 7.3.1
3. XAMPP v3.2.1
4. Notepad++

4.2 Implementasi Program

Berdasarkan analisa dan perancangan sistem yang telah diuraikan pada bab 3, maka pada subbab ini akan dijelaskan mengenai implementasi sistem tersebut menggunakan bahasa pemrograman Java. Sistem yang dibangun terdiri dari dua

bagian, yaitu pengolahan data dan pengolahan jadwal. Pada bagian pengolahan data terdapat 4 *class* penting dimana masing-masing *class* digunakan untuk mengolah data-data yang digunakan dalam proses penjadwalan. Sedangkan pada bagian pengolahan jadwal terdapat *class* yang terdiri dari *class* pengolahan jadwal untuk AIM UKPPA. Penjelasan dari masing-masing *class* dapat dilihat pada Tabel 4.1.

Tabel 4.1 *Class* Yang Dibangun

No	Nama *Class*	Keterangan
1.	`DataAuditeeUKPPA`	Kelas yang mengolah semua data *auditee* pada AIM UKPPA. Pengolahan data yang dapat dilakukan adalah penambahan, pengubahan dan penghapusan data
2.	`DataAuditorUKPPA`	Kelas yang mengolah semua data auditor pada AIM UKPPA. Pengolahan data yang dapat dilakukan adalah penambahan, pengubahan dan penghapusan data
3.	`DataKesediaanAuditeeUKPPA`	Kelas yang mengolah semua data kesediaan waktu *auditee* pada AIM UKPPA. Pengolahan data yang dapat dilakukan adalah penambahan, pengubahan dan penghapusan data.
4.	`DataKesediaanAuditorUKPPA`	Kelas yang mengolah semua data kesediaan waktu auditor pada AIM UKPPA. Pengolahan data yang dapat dilakukan adalah penambahan, pengubahan dan penghapusan data.
5.	`ProsesPenjadwalanUKPPA`	Kelas yang mengolah proses implementasi algoritma untuk menyelesaikan permasalahan penjadwalan AIM UKPPA
6.	`Pengujian`	Kelas yang mengolah proses penjadwalan menggunakan algoritma genetika dan hybrid algoritma genetika untuk memperlihatkan perbedaan generasi yang dibutuhkan.

Dari *class-class* tersebut dapat diambil beberapa fungsi / *method* utama yang mewakili setiap proses pada masing-masing *class* tersebut. Penjelasan dari masing-masing *method* dapat dilihat pada Tabel 4.2.

Tabel 4.2 Fungsi Yang Dibangun

No	Nama *Method*	Keterangan
1.	`buka_koneksi()`	Mengkoneksikan aplikasi dengan database MySQL
2.	`ambil_data()`	Mengambil data yang tersimpan di dalam database untuk ditampilkan pada tabel
3.	`insert_data()`	Menambah data kebutuhan proses penjadwalan ke dalam database
4.	`edit_data(String kode)`	Mengubah isi data kebutuhan proses penjadwalan yang berada di dalam database
5.	`delete_data(String kode)`	Menghapus data kebutuhan proses penjadwalan ke dalam database
6.	`auditor_waktu_tertentu()`	Mengelompokkan auditor berdasarkan kesamaan kesediaan waktu audit
7.	`generate_kromosom(): kromosom`	Proses *generate* populasi baru
8.	`fungsi_fitness(String[] kromosom):hasil_fit[]`	Proses perhitungan nilai *fitness* dari masing-masing individu
9.	`crossover(int u, String[][] kro_ind, String algorithm):kro_reg[][]`	Proses *crossover* yang merupakan persilangan 2 induk terpilh untuk menghasilkan individu-individu baru
10.	`mutasi(int u, String[][] kro_ind): kro_reg[][]`	Proses mutasi yang menukarkan gen terpilih dengan gen lain untuk mendapatkan individu baru
11.	`roulette_wheel(int o, String[][] kro_reg, double[] temp_fitt):kro_ind[][]`	Proses seleksi individu menggunakan metode *roulette wheel* dan menyimpan individu dengan hasil *fitness* terbaik sebagai optimum global
12.	`ambil_jadwal()`	Proses pengambilan hasil jadwal terbaik dari hasil optimum global proses sistem penjadwalan

Struktur data yang digunakan dalam proses penjadwalan auditor pada penelitian ini dapat dilihat pada Gambar 4.1.

```
String [][]kromosom_ind;
String [][]kromosom_reg;
kromosom_ind = new String[juml_ind][juml_auditee];
best = new String[juml_auditee];
kromosom_reg = new String[juml_ind+juml_cross+juml_mut][juml_auditee];
String []kromosomm = new String[juml_auditee];
String [][]rr = new String[juml_cross][juml_auditee];
String [][]mm = new String[juml_mut][juml_auditee];
save_fit = new double[100000][juml_ind];
save_pinalti = new double[2500][juml_ind];
```

Gambar 4.1 Struktur Data Proses Penjadwalan

4.2.1 Buka Koneksi

Proses buka koneksi digunakan untuk mengkoneksikan antara Java dengan *database* MySQL karena semua data masukan untuk proses penjadwalan disimpan di dalam database MySQL. Hasil jadwal yang paling optimal nantinya juga disimpan dalam *database* MySQL agar mudah untuk diakses kembali. Proses buka koneksi dapat dilihat pada Source Code 4.1.

```
private static Connection buka_koneksi() {
        if (koneksi==null) {
            try {
                String
url="jdbc:mysql://localhost:3306/ta_ver1"; //nama database
belajar
                String user="root"; //user mysql
                String password=""; //password mysql

                DriverManager.registerDriver(new
com.mysql.jdbc.Driver());

koneksi=DriverManager.getConnection(url,user,password);
            }catch (SQLException t) {
                System.out.println("Error membuat koneksi");
            }
        }
     return koneksi;
    }
```

Source Code 4.1 Fungsi Buka Koneksi Database

4.2.2 Ambil Data

Proses pengambilan data digunakan untuk mengambil data-data yang tersimpan di dalam *database* MySQL yang kemudian disimpan dalam bentuk

array untuk digunakan kembali pada proses penjadwalan. Sebelum dilakukan pengambilan data harus dipastikan terlebih dahulu bahwa program telah terkoneksi dengan *database* MySQL. Contoh proses pengambilan data *auditee* dapat dilihat pada Source Code 4.2.

```
private void ambil_data_auditee()
    {
        DefaultTableModel tbl = new DefaultTableModel();
        tbl.addColumn("Kode");
        tbl.addColumn("Auditee");
        tbl.addColumn("Unit Kerja");
        try {
             Connection c=buka_koneksi();
             Statement s= c.createStatement();
             String sql="Select * from
`ta_ver1`.`tabel_auditee_ukppa`;";
             ResultSet r=s.executeQuery(sql);
                while (r.next()) {
                    tbl.addRow(new Object[]{
                    r.getString("kode"),
                    r.getString("nama_auditee"),
                    r.getString("asal_fakultas")});
                }
                r.close();
                s.close();
        }catch(SQLException e) {
                JOptionPane.showMessageDialog(this, "Terjadi
kesalahan select data "+e.getMessage());
        }
```

Source Code 4.2 Fungsi Ambil Data Pada *Database*

4.2.3 Insert Data

Proses memasukkan data ke dalam *database* digunakan bila terdapat data baru yang juga harus diproses dalam penjadwalan atau merupakan data penunjang proses penjadwalan. Untuk melakukan penambahan data, kode primer dari data haruslah unik dan memiliki pola yang sama untuk memudahkan proses penjadwalan oleh sistem. Sebelum dilakukan penambahan data, harus dipastikan terlebih dahulu bahwa program telah terkoneksi dengan *database* MySQL. Contoh proses memasukkan data pada data *auditee* dapat dilihat pada Source Code 4.3.

```
Connection c=buka_koneksi();
        if("".equals(this.inskode_auditee.getText()) ||
"".equals(this.insnama_auditee.getText())){
            JOptionPane.showMessageDialog(this, "Field tidak
boleh kosong");
        }
        else{
            String sqlkode="Insert into tabel_auditee VALUES
('"+this.inskode_auditee.getText()+"',"
                            +
"'"+this.insnama_auditee.getText()+"',"
                            +
"'"+this.insasal_auditee.getSelectedItem()+"');";
            try { //jalankan query tersebut
                 PreparedStatement p2=(PreparedStatement)
                 c.prepareStatement(sqlkode);
                 p2.executeUpdate();
                 p2.close();
                 JOptionPane.showMessageDialog(this, "Data
berhasil ditambahkan");
             }
             catch (SQLException ex) {
                 JOptionPane.showMessageDialog(this, "Terjadi
kesalahan insert data auditor \n "+ex.getMessage());
             }
        }
```

Source Code 4.3 Fungsi Insert Data Auditee UKPPA Pada Database

4.2.4 Edit Data

Proses pengubahan isi data pada *database* digunakan bila terdapat isi data yang tidak sesuai, perubahan status auditor, penambahan *constraint*, ataupun perubahan waktu kesediaan. Untuk melakukan perubahan data, harus diperhatikan terlebih dahulu kode dari data yang akan dirubah agar data yang dirubah benar-benar data yang kita harapkan. Sebelum dilakukan penambahan data, harus dipastikan terlebih dahulu bahwa program telah terkoneksi dengan *database* MySQL. Contoh proses perubahan isi data pada data *auditee* dapat dilihat pada Source Code 4.4.

```
        Connection c=buka_koneksi();
        String sqlkode="Update tabel_auditee_ukppa SET
`nama_auditee`='"+this.enama_auditee.getText()+"',"
                        +
"`asal_fakultas`='"+this.easal_auditee.getSelectedItem()+"',
"
                        + "Where
`kode`="+this.ekode_auditee.getSelectedItem()+";";
        try {
                PreparedStatement p2=(PreparedStatement)
c.prepareStatement(sqlkode);
```

```
                p2.executeUpdate();
                p2.close();
                JOptionPane.showMessageDialog(this, "Update
data berhasil ");
        } catch (SQLException ex) {
                JOptionPane.showMessageDialog(this, "Terjadi
kesalahan update data "+ex.getMessage());
        }
```

Source Code 4.4 Fungsi Edit Data Auditee UKPPA Pada Database

4.2.5 Delete Data

Proses penghapusan data pada *database* harus diperhatikan benar kode dari data yang akan dihapus agar data yang dihapus benar-benar data yang kita harapkan. Sebelum dilakukan penghapusan data, harus dipastikan terlebih dahulu bahwa program telah terkoneksi dengan *database* MySQL. Contoh proses penghapusan isi data pada data *auditee* dapat dilihat pada Source Code 4.5.

```
        Connection c=buka_koneksi();
        int hasil = JOptionPane.showConfirmDialog(null,
"Apakah anda akan menghapus data" +
this.ddata_auditee.getSelectedItem(), null,
JOptionPane.YES_NO_OPTION);
        if (hasil == JOptionPane.YES_OPTION) {
            String sqlkode="Delete from tabel_auditee_ukppa
WHERE `kode`='"+this.ddata_auditee.getSelectedItem()+"';";
            try {
                PreparedStatement p2=(PreparedStatement)
c.prepareStatement(sqlkode);
                p2.executeUpdate();
                p2.close();
                JOptionPane.showMessageDialog(this, "Data
auditee telah berhasil dihapus");
            } catch (SQLException ex) {
                JOptionPane.showMessageDialog(this, "Terjadi
kesalahan delete data "+ex.getMessage());
            }
        }
        else{
            System.out.print("");
        }
```

Source Code 4.5 Fungsi Delete Data *Auditee* UKPPA Pada *Database*

4.2.6 Pengelompokkan Auditor Berdasarkan Waktu Kesediaan

Proses pengelompokkan auditor berdasarkan waktu kesediaan bertujuan untuk mempermudah serta mempercepat waktu proses pembangkitan populasi awal. Data auditor yang sebelumnya telah diambil dari *database* dan disimpan dalam sebuah *array* akan dikelompokkan berdasarkan pembagian waktu audit menurut masing-masing kesediaan yang dimiliki auditor dan disimpan kembali pada *array* yang terpisah. Proses pengelompokkan data auditor berdasarkan waktu kesediaannya dapat dilihat pada Source Code 4.6.

```
int[] j= {0,0,0,0,0};
        String kes = "aaaaa";
        for(int i=0;i<juml_auditor;i++){
            kes = kes_auditor[i];
            if("1".equals(kes.substring(0,1))){ j[0]++; }
            if("1".equals(kes.substring(1,2))){ j[1]++; }
            if("1".equals(kes.substring(2,3))){ j[2]++; }
            if("1".equals(kes.substring(3,4))){ j[3]++; }
            if("1".equals(kes.substring(4,5))){ j[4]++; }
        }
        //masukkan auditor ke dalam masing-masing array
sesuai waktu
        hari = new int[juml_wkt][];
        for(int h=0;h<hari.length;h++){hari[h]=new
int[j[h]];}
        int a=0;int b=0;int c=0;int d=0;int e=0;
        for(int i=0;i<juml_auditor;i++){
            kes = kes_auditor[i];
            if("1".equals(kes.substring(0,1))){
hari[0][a]=i;a++; }
            if("1".equals(kes.substring(1,2))){
hari[1][b]=i;b++; }
            if("1".equals(kes.substring(2,3))){
hari[2][c]=i;c++; }
            if("1".equals(kes.substring(3,4))){
hari[3][d]=i;d++; }
            if("1".equals(kes.substring(4,5))){
hari[4][e]=i;e++; }
        }
```

Source Code 4.6 Pengelompokkan Auditor Berdasarkan Waktu Kesediaan

4.2.7 Generate Kromosom (Populasi Awal)

Proses pembangkitan populasi awal merupakan proses dari algoritma genetika yang pertama. Pada proses ini jumlah kromosom atau individu yang

dibangkitkan adalah sejumlah inputan yang diberikan oleh *user*. Satu kromosom atau satu individu merupakan satu solusi jadwal audit. Proses pembangkitan kromosom dilakukan secara random dengan komposisi gen-gen dan subgen-subgen yang telah dijelaskan pada bagian perancangan sebelumnya. Proses pembangkitan kromosom pada penelitian ini dapat dilihat pada Source Code 4.7.

```
        for(int i=0;i<juml_auditee;i++){
            //auditee [001] -> 3 digit
            kromosom[i] = ""+kode_auditee[i];
            String kes = kes_auditee[i];
            while("0".equals(kes.substring(rand,rand+1))){
                rand = coba.nextInt(juml_wkt);
            }
            waktu = rand;
            //waktu [0011] -> 4 digit
            kromosom[i] += rand;
           //ketua dan anggota
            int h1=0; int h2=0; String kodek=""; String
           kodea="";
            while(h1<1){
               int rand1 = coba.nextInt(juml_auditor);
//ketua
               for(int j=0;j<hari[waktu].length;j++){
                  if(rand1==hari[waktu][j] &&
"Ketua".equals(status_auditor[rand1])){
                     h1++; kodek += kode_auditor[rand1];
                  }
               }
            }
            while(h2<1){
               int rand2 = coba.nextInt(juml_auditor);
//anggota
               for(int j=0;j<hari[waktu].length;j++){
                  if(rand2==hari[waktu][j] &&
"Anggota".equals(status_auditor[rand2])){
                     h2++; kodea += kode_auditor[rand2];
                  }
               }
            }
            //ketua [0011017] -> 7 digit
            kromosom[i] += kodek;
            //anggota [0011018] -> 10 digit
            kromosom[i] += kodea;
        }
    }
```

Source Code 4.7 Fungsi Generate Kromosom Awal

4.2.8 Perhitungan Nilai *Fitness*

Proses perhitungan nilai *fitness* diperoleh dengan menghitung total pinalti dari setiap gen terhadap pelanggaran yang dilakukan kemudian total pinalti tersebut dihitung menggunakan rumus *fitness* yang telah dijelaskan pada bagian perancangan bab 3. Proses perhitungan nilai *fitness* untuk masing-masing individu dapat dilihat pada Source Code 4.8.

```
        fit = 0;
        int kak; int kaa; int ke; String day;
        for(int i=0;i<juml_auditee;i++){
            //akses hari
            day = krof[i].substring(3,4);
            //akses index array auditee
            ke = Integer.parseInt(krof[i].substring(0,3))-1;
            //akses index array ketua
            kak = Integer.parseInt(krof[i].substring(4,7))-1;
            //akses index array anggota
            kaa = Integer.parseInt(krof[i].substring(7))-1;
            //constraint 1
            if(unit_kerja_auditor[kak].equals(unit_kerja_audite
            e[ke])||unit_kerja_auditor[kaa].equals(unit_kerja_a
            uditee[ke])){
                fit += 1;
            }
            //constraint 2
            if(unit_kerja_auditor[kak].equals(unit_kerja_audito
            r[kaa])){
                fit += 1;
            }
            //constraint 3
            String c31 =
""+krof[i].substring(3,4)+krof[i].substring(4,7); //ketua
            String c32 =
""+krof[i].substring(3,4)+krof[i].substring(7);   //anggota
            for(int j=i+1;j<juml_auditee;j++){
                if(krof[j].substring(3,7).equals(c31)){
                    fit += 1;
                    break;
                }
                String
t=krof[j].substring(3,4)+krof[j].substring(7);
                if(t.equals(c32)){
                    fit += 1;
                    break;
                }
            }
        }
        //constraint 4
        for(int i=0;i<juml_auditor;i++){
            int x=0;
            search:
            for(int k=0;k<juml auditee;k++){
```

```
if(krof[k].substring(4,7).equals(kode_auditor[i]) ||
krof[k].substring(7).equals(kode_auditor[i])){
                    x +=1;
                }
                if(x=3){break search;}
            }
            if(x>3){fit += 1;}
        }
```

Source Code 4.8 Fungsi Perhitungan Nilai *Fitness*

Pada baris ke 53 (*code* yang berwarna biru) didapat melalui proses *trial and error*, dimana *code* tersebut digunakan untuk menentukan agar jumlah sebaran auditor merata.

4.2.9 *Crossover* Menggunakan Metode *One-Cut-Point* dan *Tabu Search*

Proses *crossover* pada penelitian ini dikombinasikan dengan metode *tabu search* dengan harapan individu-individu baru yang terbentuk akan lebih variatif, sehingga dapat menemukan solusi dengan waktu yang lebih cepat pula. Jika setelah dilakukan proses random untuk pemilihan *parent* dan kromosom *parent* sudah pernah dilakukan proses *crossover* pada generasi yang sama sebelumnya, maka akan kembali dilakukan pemilihan *parent* secara random sampai ditemukan *parent* yang benar-benar belum mengalami *crossover* pada generasi yang sama. Proses *crossover* untuk masing-masing generasi dapat dilihat pada Source Code 4.9.

```
String [][]kro_reg = new String[u][juml_auditee];
        ArrayList<Integer> tabu_cross = new ArrayList();
        int parent1; int parent2;
        int a=0;
        //buat offspring (anak) sebanyak u
        for(int i=0;i<u/2;i++){
            parent1 = coba.nextInt(juml_ind);
            parent2 = coba.nextInt(juml_ind);
            if(algorithm=="tabu"){
                //tabu search
                if(a<1){
                    tabu_cross.add(parent1);
                    while(parent2==parent1){
                       parent2 = coba.nextInt(juml_ind);
                    }
                    tabu_cross.add(parent2);
                }
                else{
                    while(tabu_cross.contains(parent1)){
                        parent1 = coba.nextInt(juml_ind);
                    }
```

```
                    tabu_cross.add(parent1);
                    while(tabu_cross.contains(parent2)){
                        parent2 = coba.nextInt(juml_ind);
                    }
                    tabu_cross.add(parent2);
                }
                //akhir tabu search
            }
            double pc = coba.nextInt(juml_auditee);
            //pembentukan offsprings
            for(int k=0;k<juml_auditee;k++){
                if(k<pc){
                    kro_reg[a][k] = kro_ind[parent1][k];
                    kro_reg[a+1][k] = kro_ind[parent2][k];
                }
                else{
                    kro_reg[a][k] = kro_ind[parent2][k];
                    kro_reg[a+1][k] = kro_ind[parent1][k];
                }
            }
            a += 2;
        }
```

Source Code 4.9 Proses *Crossover* Menggunakan Metode *One-Cut-Point* dan *Tabu Search*

4.2.10 Mutasi Menggunakan Metode *Exchange-Point*

Metode mutasi yang digunakan pada penelitian ini adalah *exchange-point*. Penentuan individu yang akan dimutasi dilakukan secara acak, begitu pula untuk penentuan titik awal mutasi. Setelah didapat titik awal mutasi berada pada gen keberapa, cari gen setelah gen titik awal mutasi tersebut yang memiliki waktu audit sama dengan gen pada titik awal mutasi. Hal ini bertujuan untuk menghindari terbentuknya individu baru dengan nilai *fitness* yang lebih buruk. Setelah didapat titik kedua, lakukan pertukaran isi subgen dari titik awal dan titik kedua tersebut. Proses mutasi pada penelitian ini dapat dilihat pada Source Code 4.10.

```
String [][]kro_reg = new String[u][juml_auditee];
        //buat offsprings (anak) sebanyak u
        int satu=0; int dua=0;
        for(int i=0;i<u;i++){
            int parent = coba.nextInt(juml_ind);
            //tentukan waktu yang akan di mutasi
            int pm = coba.nextInt(juml_wkt);
            int mulai = coba.nextInt(40);
            kro_reg[i] = kro_ind[parent];
```

```
            for(int k=mulai;k<juml_auditee;k++){

if(kro_ind[parent][k].substring(3,4).equals(""+pm)){
                    if(satu==0){satu = k;}
                    dua = k;
                    if(dua!=satu){break;}
                }
            }

            //proses mutasi
            String tempp_1 =
kro_ind[parent][satu].substring(0,4)+kro_ind[parent][dua].su
bstring(4);
            String tempp_2 =
kro_ind[parent][dua].substring(0,4)+kro_ind[parent][satu].su
bstring(4);
            kro_reg[i][satu] = tempp_1;
            kro_reg[i][dua] = tempp_2;
        }
        return kro_reg;
```

Source Code 4.10 Fungsi Mutasi Menggunakan Metode *Exchange Point*

4.2.11 Seleksi Menggunakan Metode Elitis

Langkah pada proses seleksi dengan menggunakan metode elitis adalah melakukan perhitungan nilai *fitness* dari semua individu serta *offspring* baru hasil proses genetika. Setelah didapat nilai *fitness*-nya, urutkan individu serta *offspring* tersebut berdasarkan nilai *fitness* tertinggi. Barulah didapat populasi baru yang dihasilkan dari pengambilan individu sebanyak jumlah inisialisasi individu yang digunakan. Hasil dari pembentukan populasi baru ini merupakan individu-individu terbaik selama proses genetika berlangsung.

Setelah didapat populasi baru, cari kromosom yang memiliki *fitness* terbaik dan dibandingkan dengan hasil pada generasi sebelumnya untuk menentukan kromosom terbaik sebagai global optimum. Proses seleksi pada penelitian ini ditunjukkan pada Source Code 4.11.

```
String[][] kro_ind = new String[juml_ind][juml_auditee];
        double temporary_fit;
        String []temporary_kro = new String[juml_auditee];
        //mencari kromosom terbaik untuk dijadikan best
        for(int i=0;i<o;i++){
            tampil += "Pinalti-"+(i+1)+" =
"+temp_fitt[i]+"\n";
```

```
if(temp_m<=temp_fitt[i]){temp_m=temp_fitt[i];best =
kro_reg[i];}
        }
        //proses elitism
        for(int i=0;i<(o-1);i++){
            for(int j=(i+1);j<o;j++){

                if(temp_fitt[i]<temp_fitt[j]){
                    temporary_fit = temp_fitt[i];
                    temporary_kro = kro_reg[i];
                    System.arraycopy(temp_fitt, j,
temp_fitt, i, 1);
                    System.arraycopy(kro_reg, j, kro_reg, i,
1);
                    temp_fitt[j] = temporary_fit;
                    kro_reg[j] = temporary_kro;
                }
            }
        }
        System.arraycopy(kro_reg, 0, kro_ind, 0, juml_ind);
        return kro_ind;
```

Source Code 4.11 Fungsi Seleksi Menggunakan Elitis

4.2.12 Ambil Jadwal

Proses pengambilan jadwal terbaik yang telah disimpan ke dalam *database* dilakukan setelah proses penjadwalan selesai. Gen-gen hasil penjadwalan kemudian di transformasikan ke dalam data penjadwalan dan ditampilkan dalam tabel sehingga jadwal yang terbentuk dapat terbaca dengan jelas. Proses pengambilan jadwal dapat dilihat pada Source Code 4.12.

```
        Connection c=buka_koneksi();
        DefaultTableModel tbl = new DefaultTableModel();
        tbl.addColumn("No");
        tbl.addColumn("Hari ke-");
        tbl.addColumn("Auditee");
        tbl.addColumn("Auditor Ketua");
        tbl.addColumn("Auditor Anggota");
        for(int g=0;g<j;g++){
            try {
             Statement s= c.createStatement();
             String sql="Select nama_auditee as a from
`ta_ver1`.`tabel_auditee` where kode='"+auditee[g]+"';";
             ResultSet r=s.executeQuery(sql);
                while (r.next()) {
                    auditee[g] = r.getString("a");
                }
                r.close();
                s.close();
```

```
            }catch(SQLException e) {
                JOptionPane.showMessageDialog(this, "Terjadi
kesalahan select data "+e.getMessage());
            }
            tbl.addRow(new Object[]{
                    no[g],
                    hari[g],
                    auditee[g],
                    auditor1[g],
                    auditor2[g]});
        }
        hasil_jadwal.setModel(tbl);
        DefaultTableCellRenderer centerRenderer = new
DefaultTableCellRenderer();
        centerRenderer.setHorizontalAlignment( JLabel.CENTER
);
        for(int i=0;i<5;i++){

hasil_jadwal.getColumnModel().getColumn(i).setCellRenderer(
centerRenderer );
        }
```

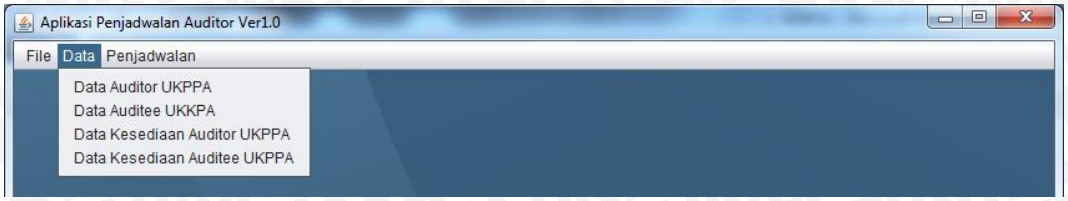
Source Code 4.12 Fungsi Ambil Jadwal Hasil Proses Penjadwalan

4.3 Implementasi Antar Muka

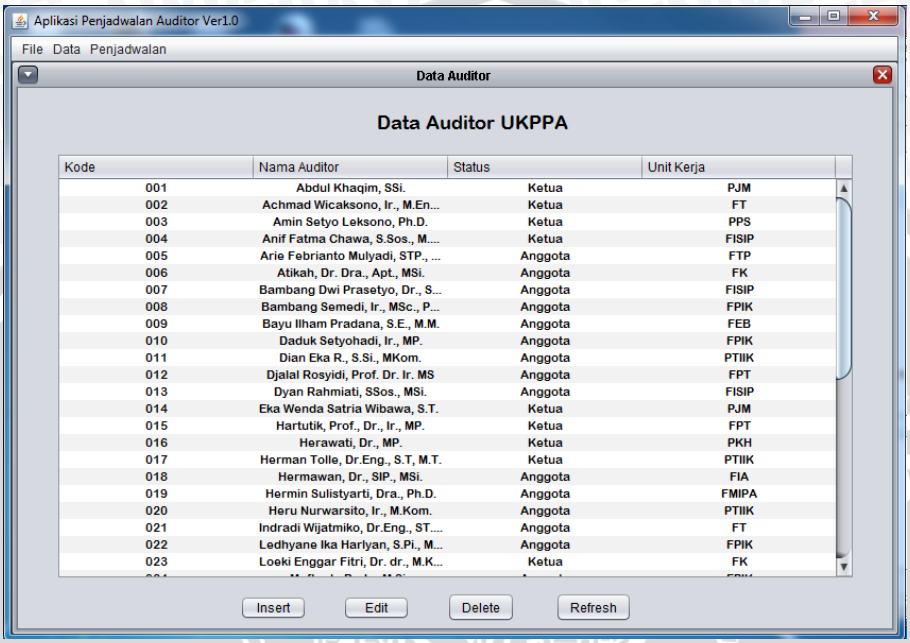
Berdasarkan rancangan antar muka yang telah dideskripsikan pada bagian perancangan antar muka bab 3, terdapat dua menu pengolahan. Menu pertama adalah pengolahan data, dan menu kedua adalah pengolahan penjadwalan.

4.3.1 Menu Pengolahan Data

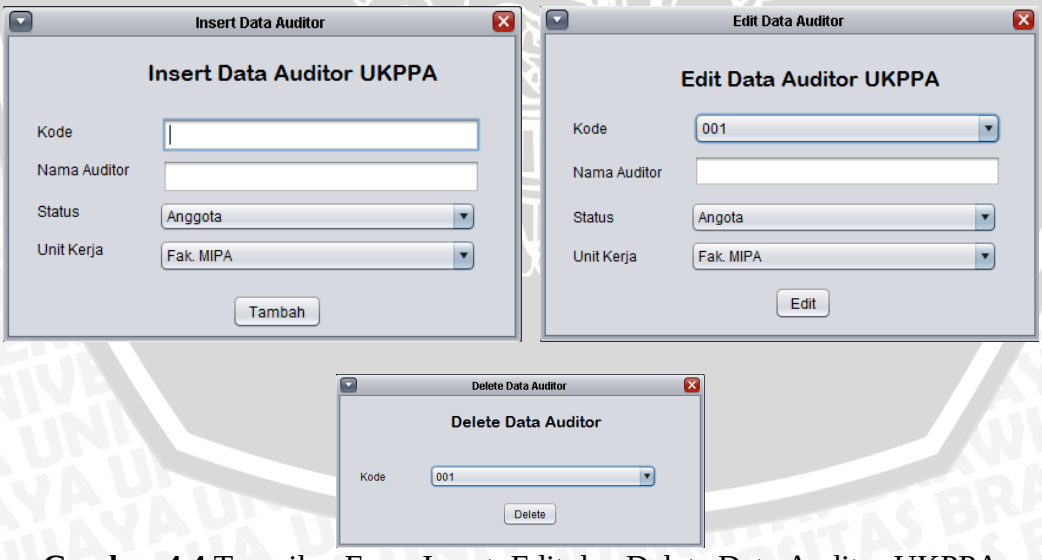
Menu “Data” yang merupakan bagian menu pengolahan data memiliki 8 submenu yang masing-masing memiliki fungsi untuk mengolah data tertentu. Submenu yang terdapat pada menu “Data” ditunjukkan pada Gambar 4.2. Antar muka dan fungsi-fungsi pengolahan data pada data audit UKPPA Tampilan antar muka fungsi pengolahan data dapat dilihat pada Gambar 4.3 sampai dengan Gambar 4.4.



Gambar 4.2 Submenu Pada Menu Data



Gambar 4.3 Tampilan Form Data Auditor UKPPA

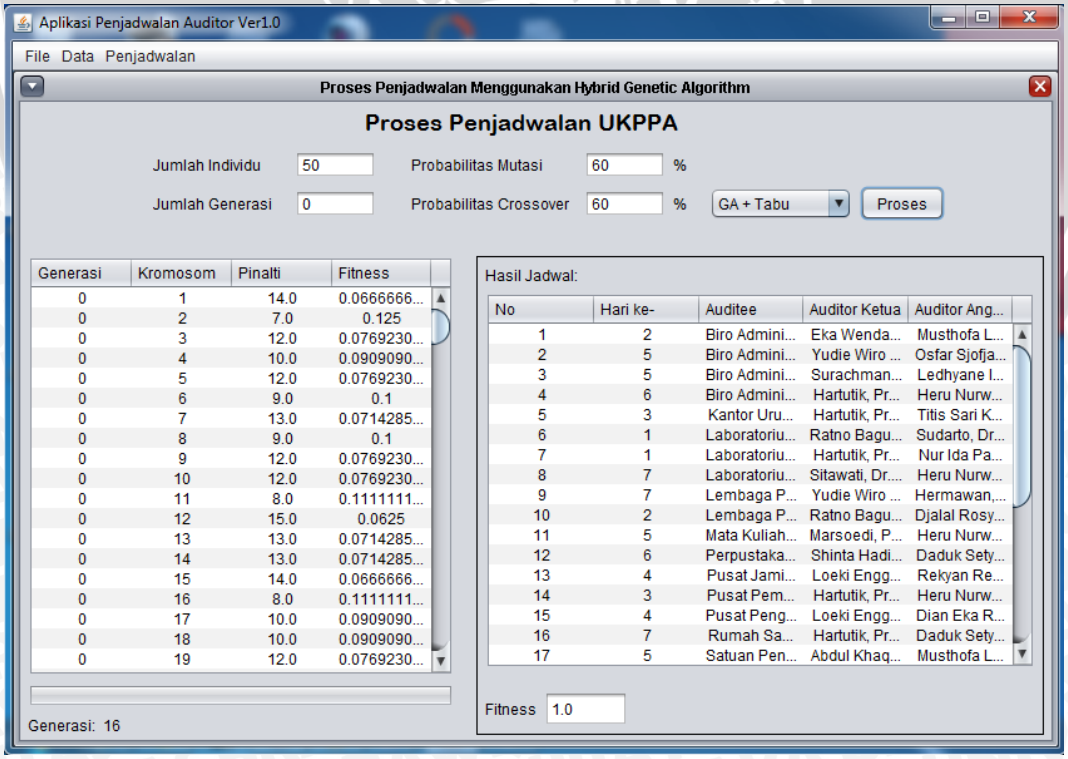


Gambar 4.4 Tampilan Form Insert, Edit dan Delete Data Auditor UKPPA

4.3.2 Menu Pengolahan Penjadwalan

Menu "Penjadwalan" yang merupakan bagian menu pengolahan penjadwalan memiliki 2 submenu. Submenu pertama digunakan untuk melakukan pengolahan penjadwalan audit UKPPA, dan submenu kedua untuk melakukan pengujian terhadap algoritma genetika dan *hybrid* algoritma genetika. Submenu yang terdapat pada menu "Penjadwalan" ditunjukkan pada Gambar 4.5, sedangkan tampilan antar muka hasil implementasi pada penjadwalan UKPPA ditunjukkan pada Gambar 4.6.

Gambar 4.5 Submenu Pada Menu Penjadwalan



Gambar 4.6 Tampilan Hasil Proses Penjadwalan UKPPA

BAB V
PENGUJIAN DAN ANALISIS

Pada bab ini akan dibahas dan dianalisis hasil dari uji coba algoritma *hybrid* genetika pada permasalahan penjadwalan audit internal mutu.

5.1 Analisis Hasil Uji Coba

Fungsi tujuan dari implementasi algoritma *hybrid* genetika pada permasalahan penjadwalan audit internal mutu ini adalah untuk mendapatkan hasil jadwal yang optimal (tidak ada *constraint* yang terlanggar) serta mengurangi waktu pencarian solusi. Berikut merupakan tabel dan grafik jumlah generasi hasil uji coba dari masing-masing skenario percobaan yang telah dirancang sebelumnya.

5.1.1 Pengujian dengan Jumlah Individu Sebagai Pembanding

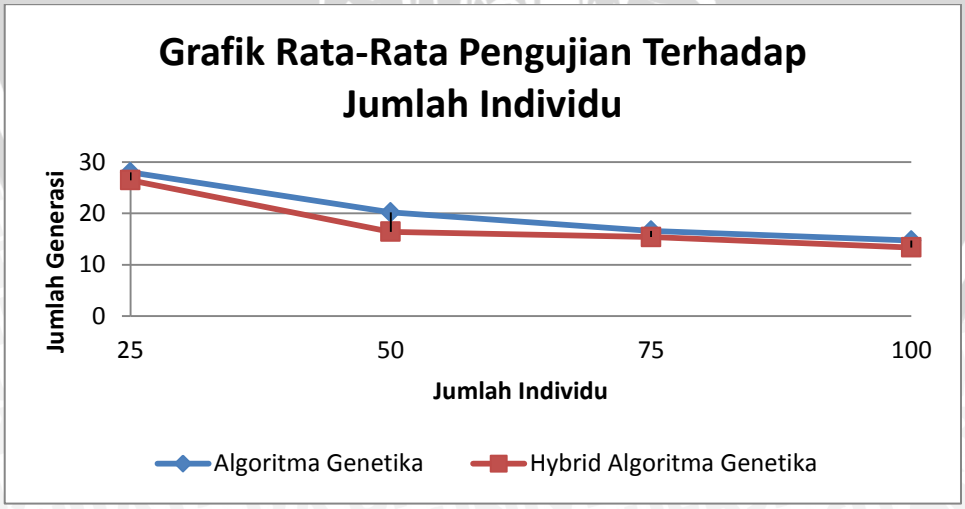
Pada skenario pengujian yang pertama, jumlah individu yang digunakan sebagai pembanding adalah 25, 50, 75, dan 100 individu dengan masing-masing 20 kali uji coba. Pada pengujian ini nilai probabilitas *crossover* yang digunakan sebesar 0.5, dan probabilitas mutasi 0.5. Pengujian dilakukan untuk menghitung jumlah generasi dari masing-masing algoritma terhadap jumlah individu (jumlah populasi) yang digunakan untuk menghasilkan solusi jadwal yang optimal (tidak ada *constraint* yang terlanggar).

Tabel 5.1 Hasil Pengujian Jumlah Generasi Terhadap Jumlah Individu

Percobaan ke-	Jumlah Individu							
	25		50		75		100	
	GA	GA + Tabu	GA	GA + Tabu	GA	GA + Tabu	GA	GA + Tabu
1.	24	45	18	19	16	15	15	10
2.	21	23	22	17	23	15	22	18
3.	32	22	16	14	18	18	14	14
4.	25	10	12	12	22	14	18	15
5.	47	41	22	15	12	15	17	15
6.	22	19	22	14	15	20	12	16

7.	33	19	14	9	19	13	7	14
8.	23	24	34	22	20	13	18	11
9.	41	14	19	15	18	12	14	10
10.	68	47	18	14	15	13	11	13
11.	17	22	19	13	17	9	17	21
12.	28	23	20	12	18	20	11	13
13.	40	22	24	16	13	18	18	18
14.	23	31	26	30	15	19	14	11
15.	13	15	12	20	13	16	11	8
16.	16	23	27	20	17	15	18	9
17.	25	19	16	16	15	12	18	16
18.	19	54	24	13	11	10	16	10
19.	20	28	14	16	21	24	10	14
20.	23	28	25	21	14	17	13	11
Jumlah	560	529	404	328	332	308	294	267
Rata-rata	28	26.45	20.2	16.4	16.6	15.4	14.7	13.35

Berdasarkan hasil pengujian pada Tabel 5.1 dapat dilihat bahwa pengguanan metode *Hybrid* Algoritma Genetika menghasilkan rata-rata jumlah generasi lebih sedikit dari pada metode Algoritma Genetika biasa. Rata-rata selisih jumlah generasi antara kedua algoritma pada penggunaan parameter jumlah individu adalah sebanyak 2 generasi dengan perkiraan perbedaan waktu komputasi sebesar 1 sampai 2 detik.



Gambar 5.1 Grafik Pengujian Terhadap Jumlah Individu

Dari hasil uji coba terhadap jumlah individu di atas dapat dibuat grafik rata-rata jumlah generasi seperti pada Gambar 5.1. Dari grafik tersebut terlihat

bahwa jumlah individu yang digunakan berpengaruh terhadap jumlah generasi untuk mendapatkan solusi penjadwalan yang optimal. Pada jumlah individu sebanyak 25, Algoritma Genetika dan *Hybrid* Algoritma Genetika menghasilkan solusi jadwal dengan rata-rata jumlah generasi paling banyak dibandingkan dengan 3 penggunaan jumlah individu lainnya. Sehingga dapat disimpulkan bahwa semakin banyak jumlah individu yang digunakan dapat memberikan solusi jadwal dengan jumlah generasi lebih sedikit.

5.1.2 Pengujian dengan Probabilitas *Crossover* Sebagai Pembanding

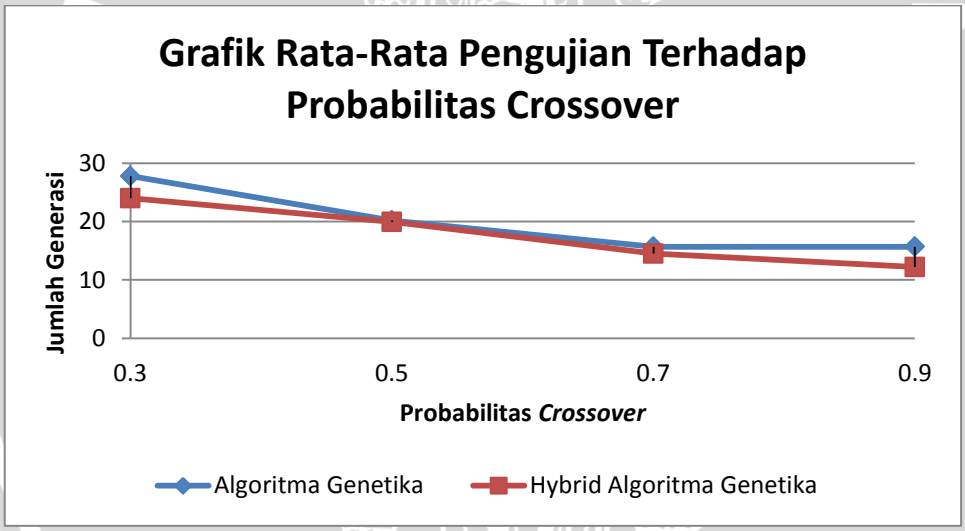
Pada skenario pengujian yang kedua, jumlah probabilitas *crossover* yang digunakan sebagai pembanding adalah 0.3, 0.5, 0.7, dan 0.9 dengan masing-masing 20 kali uji coba. Pada pengujian ini nilai probabilitas mutasi yang digunakan sebesar 0.5, dan jumlah individu 50. Pengujian dilakukan untuk menghitung jumlah generasi dari masing-masing algoritma terhadap nilai probabilitas *crossover* yang digunakan untuk menghasilkan solusi jadwal yang optimal (tidak ada *constraint* yang terlanggar).

Tabel 5.2 Hasil Pengujian Jumlah Generasi Terhadap Probabilitas *Crossover*

Percobaan ke-	Probabilitas *Crossover*							
	0.3		0.5		0.6		0.9	
	GA	GA + Tabu	GA	GA + Tabu	GA	GA + Tabu	GA	GA + Tabu
1.	24	18	13	19	18	9	31	14
2.	60	21	20	18	17	14	12	11
3.	24	25	16	11	19	20	25	12
4.	25	22	23	16	8	10	12	7
5.	21	16	22	15	16	11	12	10
6.	32	30	19	15	14	8	13	15
7.	26	24	18	33	12	14	17	11
8.	35	24	22	22	19	15	13	6
9.	29	27	24	13	15	21	11	14
10.	30	10	15	15	12	14	14	14
11.	26	30	16	19	18	10	21	18
12.	30	13	22	22	16	16	14	16
13.	34	26	23	19	11	13	21	10
14.	28	21	17	24	13	18	11	14
15.	25	32	36	20	21	15	12	14

16.	20	27	18	23	13	13	15	14
17.	24	48	24	28	19	11	5	9
18.	22	15	21	21	25	20	19	11
19.	21	33	18	24	12	20	17	13
20.	20	18	17	22	16	18	19	11
Jumlah	556	480	404	399	314	290	314	244
Rata-rata	27.8	24	20.2	19.95	15.7	14.5	15.7	12.2

Berdasarkan hasil pengujian pada Tabel 5.2 dapat dilihat bahwa pengguanan metode *Hybrid* Algoritma Genetika masih menghasilkan rata-rata jumlah generasi lebih sedikit dari pada metode Algoritma Genetika biasa. Rata-rata selisih jumlah generasi antara kedua algoritma pada penggunaan parameter nilai probabilitas *crossover* adalah sebanyak 2 generasi dengan perkiraan perbedaan waktu komputasi sebesar 1 sampai dengan 2 detik.



Gambar 5.2 Grafik Pengujian Terhadap Probabilitas *Crossover*

Dari hasil uji coba terhadap jumlah probabilitas *crossover* yang telah dilakukan dapat dibuat grafik rata-rata jumlah generasi seperti pada Gambar 5.2. Dari grafik tersebut, dapat dilihat bahwa nilai probabilitas *crossover* yang digunakan juga berpengaruh terhadap jumlah generasi untuk mendapatkan solusi penjadwalan yang optimal. Pada penggunaan nilai probabilitas *crossover* 0.3, Algoritma Genetika dan *Hybrid* Algoritma Genetika menghasilkan solusi jadwal dengan rata-rata jumlah generasi paling banyak dibandingkan dengan 3 penggunaan nilai probabilitas *crossover* lainnya. Sehingga dapat disimpulkan

bahwa semakin tinggi nilai probabilitas *crossover* yang digunakan akan dapat memberikan solusi jadwal dengan jumlah generasi lebih sedikit.

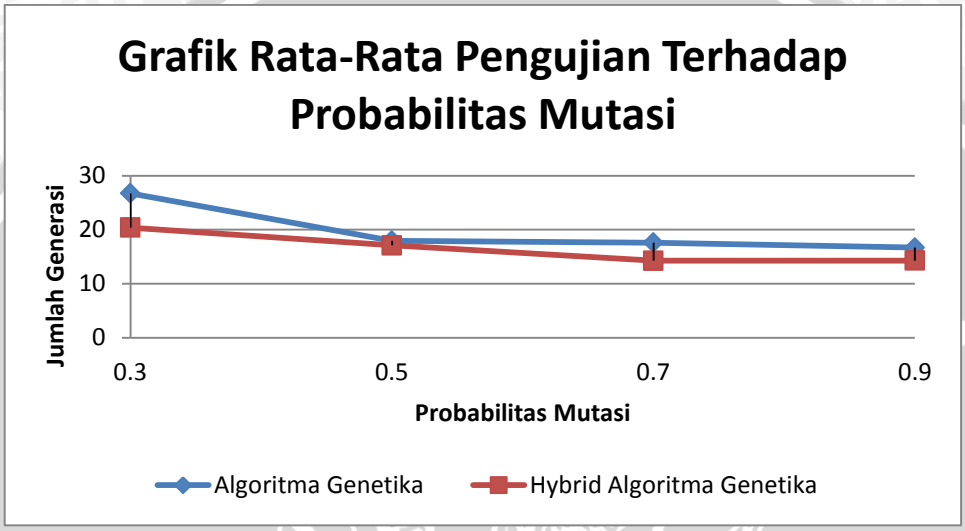
5.1.3 Pengujian dengan Probabilitas Mutasi Sebagai Pembanding

Pada skenario pengujian yang ketiga, jumlah probabilitas mutasi yang digunakan sebagai pembanding adalah 0.3, 0.5, 0.7, dan 0.9 dengan masing-masing 20 kali uji coba. Pada pengujian ini nilai probabilitas *crossover* yang digunakan sebesar 0.5, dan jumlah individu 50. Pengujian dilakukan untuk menghitung jumlah generasi dari masing-masing algoritma terhadap nilai probabilitas mutasi yang digunakan untuk menghasilkan solusi jadwal yang optimal (tidak ada *constraint* yang terlanggar).

Tabel 5.3 Hasil Pengujian Jumlah Generasi Terhadap Probabilitas Mutasi

Percobaan ke-	Probabilitas Mutasi							
	0.3		0.5		0.6		0.9	
	GA	GA + Tabu	GA	GA + Tabu	GA	GA + Tabu	GA	GA + Tabu
1.	25	15	21	11	19	12	18	14
2.	22	18	9	16	21	17	20	14
3.	20	27	16	13	18	14	27	13
4.	29	18	20	19	27	14	20	15
5.	14	18	23	25	18	12	16	12
6.	29	21	28	13	13	13	13	7
7.	21	12	11	28	19	16	12	14
8.	16	9	21	13	17	19	15	9
9.	18	15	20	17	17	13	19	24
10.	34	24	21	14	25	14	17	14
11.	56	21	17	16	14	17	12	19
12.	26	16	13	23	22	16	27	15
13.	32	27	14	20	18	14	17	16
14.	23	34	14	16	15	12	10	17
15.	32	17	12	12	18	11	13	10
16.	25	24	32	26	10	16	19	11
17.	25	17	23	20	19	7	12	16
18.	22	25	14	9	11	16	10	18
19.	45	25	10	13	17	17	14	15
20.	21	25	20	18	14	15	23	13
Jumlah	535	408	359	342	352	285	334	286
Rata-rata	26.75	20.4	17.95	17.1	17.6	14.25	16.7	14.3

Dari hasil pengujian pada Tabel 5.3 terlihat bahwa penggunaan metode *Hybrid* Algoritma Genetika juga menghasilkan rata-rata jumlah generasi lebih sedikit dari pada metode Algoritma Genetika biasa pada pengujian dengan parameter nilai probabilitas mutasi. Rata-rata selisih jumlah generasi antara kedua algoritma pada parameter ini adalah sebanyak 3 generasi dengan perkiraan perbedaan waktu komputasi sebesar 1 sampai 3 detik.



Gambar 5.3 Grafik Pengujian Terhadap Probabilitas Mutasi

Dari hasil uji coba terhadap jumlah probabilitas mutasi yang telah dilakukan dapat dibuat grafik rata-rata jumlah generasi seperti pada Gambar 5.3. Dari grafik tersebut, dapat dilihat bahwa pengguanan metode *Hybrid* Algoritma Genetika juga masih menghasilkan rata-rata jumlah generasi lebih sedikit dari pada metode Algoritma Genetika. Nilai probabilitas mutasi yang digunakan juga berpengaruh terhadap jumlah generasi untuk mendapatkan solusi penjadwalan yang optimal. Sama halnya dengan probabilitas *crossover*, pada penggunaan nilai probabilitas mutasi 0.3 Algoritma Genetika dan *Hybrid* Algoritma Genetika menghasilkan solusi jadwal dengan rata-rata jumlah generasi paling banyak dibandingkan dengan 3 penggunaan nilai probabilitas mutasi lainnya. Sehingga dapat disimpulkan bahwa semakin tinggi nilai probabilitas mutasi yang digunakan akan dapat memberikan solusi jadwal dengan jumlah generasi lebih sedikit.

Kecilnya perbedaan waktu komputasi pada ketiga pengujian tersebut dipengaruhi oleh jumlah data yang digunakan. Semakin banyak data auditor dan

data *auditee* yang digunakan, maka akan semakin terlihat perbedaan jumlah generasi dan waktu komputasi yang digunakan dari kedua algoritma tersebut.

BAB VI
PENUTUP

6.1 Kesimpulan

Berdasarkan hasil implementasian serta hasil pengujian sistem penjadwalan audit, maka dapat disimpulkan bahwa:

1. Metode *Hybrid* Algoritma Genetika dapat diimplementasikan dalam proses pembuatan jadwal Audit Internal Mutu UKPPA Universitas Brawijaya dengan baik. Penggunaan metode seleksi elitis serta penggunaan metode *tabu search* pada proses *crossover* memberikan hasil yang optimal (tidak lagi terdapat *constraint* yang terlanggar) yakni dengan nilai *fitness* 1 dan dapat mengurangi waktu pencarian solusi seperti yang diharapkan.
2. Berdasarkan hasil pengujian terhadap 31 data *auditee* dan 46 data auditor, metode *Hybrid* Algoritma Genetika mampu memberikan solusi dengan rata-rata jumlah generasi lebih sedikit dibandingkan dengan metode Algoritma Genetika pada semua parameter pengujian yang digunakan dengan selisih rata-rata 1 sampai dengan 6 generasi. Sehingga dapat disimpulkan penggunaan *Hybrid* Algoritma Genetika dapat mengurangi waktu pencarian solusi jadwal dengan selisih waktu rata-rata antara 1 sampai 3 detik dibandingkan dengan pengguaan algoritma genetika biasa.
3. Penggunaan jumlah individu, besar nilai probabilitas *crossover* dan nilai probabilitas mutasi dapat mempengaruhi banyaknya generasi yang dibutuhkan untuk mendapatkan hasil jadwal yang optimum. Semakin banyak jumlah individu yang digunakan maka akan mengurangi jumlah generasi yang dibutuhkan untuk mendapatkan solusi optimum. Begitu pula pada besar nilai probabilitas *crossover* dan probabilitas mutasi, semakin besar probabilitas yang digunakan juga akan mengurangi jumlah generasi yang dibutuhkan.

6.2 Saran

Saran yang dapat diberikan setelah menyelesaikan penelitian skripsi ini adalah:

1. Aplikasi ini dapat diimplementasikan pada proses penjadwalan Audit Internal Mutu UKPA dengan menambahkan beberapa *constraint* dalam perhitungan *fitness*, sehingga dapat diketahui perbandingan hasil uji coba penggunaan *Hybrid* Algoritma Genetika dengan Algoritma Genetika pada jumlah data yang banyak.
2. Dapat ditambahkan variabel prioritas kepada auditor yang berstatus sebagai ketua, agar auditor ketua dapat dipasangkan menjadi satu tim audit dengan auditor lain yang juga berstatus sebagai ketua.
3. Mekanisme pembentukan jadwal audit dapat dibuat lebih lebih fleksibel sesuai dengan jumlah ketersediaan auditor dan *auditee* serta mempertimbangkan keseimbangan jumlah sebaran auditor.
4. Dapat ditambahkan penanganan *error* dan pemberian solusi ketika jumlah ketersediaan auditor tidak mencukupi untuk dilakukannya audit secara keseluruhan.

DAFTAR PUSTAKA

[ABD-08] Abdullah, Ihsan S. 2008. *Analisis dan Implementasi Algoritma Genetika pada Penjadwalan Mata Kuliah*. Jurusan Teknik Informatika Fakultas Teknik dan Ilmu Komputer Universitas Komputer Indonesia. Bandung.

[ADM-11] Admin. 2011. *Audit Internal*. http://spi.ub.ac.id/jaminan-mutu/audit/audit-internal/ [10 Mar 2014].

[BAK-74] Baker, Kenneth R. 1974. *Introduction to Sequencing and Scheduling*. John Wiley & Sons, Inc. New York.

[BAS-03] Basuki, Achmad. 2003. *Strategi Menggunakan Algoritma Genetika*. Politeknik Elektronika Negeri Surabaya PENS-ITS. Surabaya.

[DAR-11] Darmawan, Irfan. 2011. *Hibridisasi Genetic-Tabu Search Algoritmh untuk Penjadwalan Job Terhadap Beberap Resource di dalam Komputasi Grid*. Seminar Nasional Aplikasi Teknologi Informasi 2011 (SNATI 2011), Yogyakarta.

[GIN-09] Ginting, Rosnaini. 2009. *Penjadwalan Mesin*. Graha Ilmu. Yogyakarta.

[GUN-99] Gunes, Evrim D. 1999. *Workforce Scheduling*. Departmen of Industrial Engineering, Bilkent University.

[HUS-09] Husen, Abrar. 2009. *Manajemen Proyek*. Penerbit Andi. Yogyakarta.

[JUN-03] Juniawati. 2003. *Implementasi Algoritma Genetika untuk MencariVolume Terbesar Bangun Kotak Tanpa Tutup Dari Suatu Bidang Datar Segi Empat*. Jurnal Ilmiah LPPM Universitas Surabaya. Surabaya.

[KAR-11] Karas, Ismail R., & Umit, Atila. 2011. *A Genetic Approach For Finding The Shortest Driving Time On Mobile Device*. Scientific Research and Essays Vol 6(2), pp 394-405.

[KUS-03] Kusumadewi, Sri. 2003. *Artificial Intelligence (Teknik dan Aplikasinya)*. Graha Ilmu. Yogyakarta.

[KUS-05] Kusumadewi, S. & Purnomo, H. 2005. *Penyelesaian Masalah Optimasi Menggunakan Teknik-teknik Heuristik*. Graha Ilmu. Yogyakarta.

[MIT-99] Mitchell, Melanie. 1999. *An Introduction to Genetic Algorithms.* Massachusetts Institute of Technology. A Bradford Book The MIT Press. England.

[RIF-11] Rifai, Ulisna A. 2011. *Pengembangan Aplikasi Penjadwalan Kegiatan dengan Menggunakan Algoritma Genetika (Studi Kasus: Humas Kementrian Agama RI)*. Fakultas Sains dan Teknologi Universitas Islam Negeri Syarif Hidayatullah. Jakarta.

[SET-09] Setemen, K. & Purnomo, M. H. 2009. *Kombinasi Algoritma Genetika dan Tabu Search dalam Pembuatan Tabel Jadwal Mata Kuliah*. Seminar on Intelligence Technology and Its Application.

[SIA-10] Sianipar, Yanti N. 2010. *Optimasi Penjadwalan Mata Kuliah di Perguruan Tinggi dengan Menggunakan Algoritma Memetika*. Universitas Indonesia, Departemen Teknik Industri.

[SIA-12] Sianturi, Andrew L. 2012. *Optimasi Penjadwalan Karyawan Pengawas Pembangunan Kapal dengan Menggunakan Algoritma Genetika*. Fakultas Teknik Program Studi Teknik Industri Universitas Indonesia. Depok.

[SUK-03] Sukmawan, B. 2003. *Sekilas Tentang Algoritma Genetika dan Aplikasinya pada Optimasi Jaringan Pipa Air Bersih*. http://bimacipta.com/algoritma-genetika.html [28 Feb 2014].

[SUY-11] Suyanto. 2011. *Artificial Intelligence*. Informatika. Bandung.

[SYA-08] Syarif, A. & Ardiansyah. 2008. *Aplikasi Hybrid Genetic Algorithm dengan Fuzzy Logic Controller: pada Traveling Salesman Problem (Studi Kasus: Rute Truk Sampah di Kota Bandar Lampung)*. Prosiding Seminar Nasional Sains dan Teknologi-II 2008. Universitas Lampung.

[WRE-96] Wren, A. 1996. Scheduling, Timetabling, and Rostering – A Special Relationship?. *Practice and Theory of Automated Timetabling.*

Lecture Notes in Computer Science. 1153, 46-75, DOI:10.1007/3-540-61794-9_51.

[ZAI-13] Zaini, Muhammad Fariz. 2013. *Optimasi Penjadwalan Mata Pelajaran Menggunakan Genetic Algorithm (Studi Kasus: Madrasah Tsanawiyah Negeri Bangil)*. Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya. Malang.

Lampiran 1 : Data *Auditee* AIM UKPPA Siklus 7 Universitas Brawijaya

No.	Nama *Auditee*
1	Biro Administrasi Akademik dan Kerjasama (BAAK)
2	Biro Administrasi Kemahasiswaan (BAK)
3	Biro Administrasi Keuangan dan Perencanaan (BAKP)
4	Biro Administrasi Umum dan Kepegawaian (BAUK)
5	Kantor Urusan Internasional (International Office)
6	Laboratorium Biosains (Biosains)
7	Laboratorium Sentral Ilmu Hayati (LSIH)
8	Laboratorium Sentral Sains dan Rekayasa (L2SR)
9	Lembaga Penelitian & Pengabdian kepada Masyarakat (LPPM)
10	Lembaga Pengkajian & Pengembangan Pendidikan (LP3)
11	Mata Kuliah Umum (MKU)
12	Perpustakaan
13	Pusat Jaminan Mutu (PJM)
14	Pusat Pembinaan Agama (PPA)
15	Pusat Pengelola Informasi, Dokumentasi dan Keluhan (PIDK)
16	Rumah Sakit Akademik dan Poliklinik (RSAP)
17	Satuan Pengawasan Internal (SPI)
18	UB Press
19	Unit Bisnis (Corporate)
20	Unit Bisnis (UB Hotel)
21	Unit Bisnis (Griya Brawijaya)
22	Unit Bisnis (Yubi Travel)
23	Unit Bisnis (UB Media)
24	Pusat Inkubator Bisnis dan Layanan Masyarakat (PIBLAM)
25	Unit Job Placement Center (JPC)
26	Unit Layanan Pengadaan (ULP)
27	Unit Teknologi Informasi & Komunikasi (TIK)
28	Unit Peningkatan Publikasi Internasional dan Karya Ilmiah Dosen (PPIKID)
29	Unit Pengelola Program Double Degree
30	Unit Pemeringkatan Internasional (UPI)
31	Tim Koordinasi Program Hibah Kompetisi (TKPHK)

Berikut daftar kesediaan dari masing-masing *Auditee*.

No.	Nama *Auditee*	Waktu Audit							
		Senin, 12 Mei 2014	Selasa, 13 Mei 2014	Rabu, 13 Mei 2014	Senin, 19 Mei 2014	Selasa, 20 Mei 2014	Rabu, 21 Mei 2014	Kamis, 22 Mei 20134	Jumat, 23 Mei 2014
1	Biro Administrasi Akademik dan Kerjasama (BAAK)			V	V				
2	Biro Administrasi Kemahasiswaan (BAK)						V	V	
3	Biro Administrasi Keuangan dan Perencanaan (BAKP)						V		
4	Biro Administrasi Umum dan Kepegawaian (BAUK)							V	
5	Kantor Urusan Internasional (International Office)				V				
6	Laboratorium Biosains (Biosains)		V						
7	Laboratorium Sentral Ilmu Hayati (LSIH)		V						
8	Laboratorium Sentral Sains dan Rekayasa (L2SR)							V	V
9	Lembaga Penelitian & Pengabdian kepada Masyarakat (LPPM)								V
10	Lembaga Pengkajian & Pengembangan Pendidikan (LP3)			V					
11	Mata Kuliah Umum (MKU)						V		

12	Perpustakaan							V	
13	Pusat Jaminan Mutu (PJM)					V			
14	Pusat Pembinaan Agama (PPA)				V				V
15	Pusat Pengelola Informasi, Dokumentasi dan Keluhan (PIDK)				V	V			
16	Rumah Sakit Akademik dan Poliklinik (RSAP)								V
17	Satuan Pengawasan Internal (SPI)			V			V		
18	UB Press				V				
19	Unit Bisnis (Corporate)								
20	Unit Bisnis (UB Hotel)	V		V					
21	Unit Bisnis (Griya Brawijaya)		V	V					
22	Unit Bisnis (Yubi Travel)								V
23	Unit Bisnis (UB Media)				V	V			
24	Pusat Inkubator Bisnis dan Layanan Masyarakat (PIBLAM)								V
25	Unit Job Placement Center (JPC)						V	V	
26	Unit Layanan Pengadaan (ULP)							V	V
27	Unit Teknologi Informasi & Komunikasi (TIK)				V				
28	Unit Peningkatan Publikasi Internasional dan Karya Ilmiah Dosen (PPIKID)							V	V
29	Unit Pengelola Program Double								V

	Degree								
30	Unit Pemeringkatan Internasional (UPI)							V	V
31	Tim Koordinasi Program Hibah Kompetisi (TKPHK)				V			V	

Lampiran 2 : Data Auditor AIM UKPPA Siklus 7 Universitas Brawijaya

No.	Nama Auditor	Unit Kerja
1	Abdul Khaqim, SSi.	PJM
2	Achmad Wicaksono, Ir., M.Eng., Ph.D.	FT
3	Amin Setyo Leksono, Ph.D.	PPS
4	Anif Fatma Chawa, S.Sos., M.Si., Ph.D.	FISIP
5	Arie Febrianto Mulyadi, STP., MP.	FTP
6	Atikah, Dr. Dra., Apt., MSi.	FK
7	Bambang Dwi Prasetyo, Dr., S.Sos., M.Si.	FISIP
8	Bambang Semedi, Ir., MSc., PhD.	FPIK
9	Bayu Ilham Pradana, S.E., M.M.	FEB
10	Daduk Setyohadi, Ir., MP.	FPIK
11	Dian Eka R., S.Si., MKom.	PTIIK
12	Djalal Rosyidi, Prof. Dr. Ir. MS	FPT
13	Dyan Rahmiati, SSos., MSi.	FISIP
14	Eka Wenda Satria Wibawa, S.T.	PJM
15	Hartutik, Prof., Dr., Ir., MP.	FPT
16	Herawati, Dr., MP.	PKH
17	Herman Tolle, Dr.Eng., S.T, M.T.	PTIIK
18	Hermawan, Dr., SIP., MSi.	FIA
19	Hermin Sulistyarti, Dra., Ph.D.	FMIPA
20	Heru Nurwarsito, Ir., M.Kom.	PTIIK
21	Indradi Wijatmiko, Dr.Eng., ST., MEng. (Prac).	FT
22	Ledhyane Ika Harlyan, S.Pi., M.Sc.	FPIK
23	Loeki Enggar Fitri, Dr. dr., M.Kes., Sp.ParK.	FK
24	Maftuch, Dr. Ir., M.Si.	FPIK
25	Marsoedi, Prof. Ir., Ph.D.	FPIK
26	Mas'ud Effendi, STP., MP.	FTP
27	Musthofa Lutfi, Ir., MP.	FTP
28	Nila Firdausi Nuzula, S.Sos., M.Si., Ph.D.	FIA
29	Nur Ida Panca N., STP., MP.	FTP
30	Osfar Sjofjan, Dr. Ir. M.Sc.	FPT
31	Ratno Bagus Edy Wibowo, S.Si., M.Si., Ph.D.	FMIPA
32	Rekyan Regasari, ST., MT.	PTIIK
33	Restu Karlina Rahayu, SIP., MSi.	FISIP
34	Rurini Retnowati, Dr.	FMIPA
35	Shinta Hadiyantina, Dr., SH., MH.	FH
36	Sigit Kusmaryanto, Ir., MEng.	FT

37	Sitawati, Dr. Ir., M.S.	FP
38	Siti Aisjah, S.E., M.S., Dr.	FEB
39	Sri Karindah, Dr., Ir., MS.	FP
40	Sri Palupi Prabandari, S.E., M.M.	FEB
41	Sudarto, Dr. Ir., MS.	FP
42	Surachman, Prof. Dr., MSi.E.	FEB
43	Titis Sari Kusuma, S.Gz.	FK
44	Tri Novi Kurnia Wardhani, S.ST., M.Kes.	FK
45	Yoyon Supriyono, SPsi., MPsi.	FISIP
46	Yudie Wiro Eko Setyawan, S.Si.	PJM

Berikut daftar kesediaan dari masing-masing Auditor.

No.	Nama Auditor	Waktu Audit							
		Senin, 12 Mei 2014	Selasa, 13 Mei 2014	Rabu, 13 Mei 2014	Senin, 19 Mei 2014	Selasa, 20 Mei 2014	Rabu, 21 Mei 2014	Kamis, 22 Mei 20134	Jumat, 23 Mei 2014
1	Abdul Khaqim, SSi.	v	v	v	v	v	v	v	
2	Achmad Wicaksono, Ir., M.Eng., Ph.D.		v	v		v	v		v
3	Amin Setyo Leksono, Ph.D.	v	v	v	v				v
4	Anif Fatma Chawa, S.Sos., M.Si., Ph.D.	v			v			v	
5	Arie Febrianto Mulyadi, STP., MP.	v			v			v	
6	Atikah, Dr. Dra., Apt., MSi.		v			v		v	v
7	Bambang Dwi Prasetyo, Dr., S.Sos., M.Si.	v	v					v	
8	Bambang Semedi, Ir., MSc., PhD.	v	v	v					
9	Bayu Ilham Pradana, S.E., M.M.			v			v	v	
10	Daduk Setyohadi, Ir., MP.					v	v	v	v
11	Dian Eka R., S.Si., MKom.		v					v	
12	Djalal Rosyidi, Prof. Dr. Ir. MS		v	v		v	v		v
13	Dyan Rahmiati, SSos., MSi.					v	v		v
14	Eka Wenda Satria Wibawa, S.T.		v	v		v	v	v	
15	Hartutik, Prof., Dr., Ir., MP.	v	v	v	v	v	v		v
16	Herawati, Dr., MP.	v	v						
17	Herman Tolle, Dr.Eng., S.T, M.T.	V			v				v

18	Hermawan, Dr., SIP., MSi.	v	v						v
19	Hermin Sulistyarti, Dra., Ph.D.	v			v		v	v	
20	Heru Nurwarsito, Ir., M.Kom.	v		v	v		v		v
21	Indradi Wijatmiko, Dr.Eng., ST., MEng. (Prac).					v		v	v
22	Ledhyane Ika Harlyan, S.Pi., M.Sc.			v			v		
23	Loeki Enggar Fitri, Dr. dr., M.Kes., Sp.ParK.				v	v	v	v	v
24	Maftuch, Dr. Ir., M.Si.	v	v	v					
25	Marsoedi, Prof. Ir., Ph.D.	v	v						
26	Mas'ud Effendi, STP., MP.	v	v	v					
27	Musthofa Lutfi, Ir., MP.		v	v		v	v	v	
28	Nila Firdausi Nuzula, S.Sos., M.Si., Ph.D.		v			v			v
29	Nur Ida Panca N., STP., MP.	v	v	v	v	v			
30	Osfar Sjofjan, Dr. Ir. M.Sc.				v		v		
31	Ratno Bagus Edy Wibowo, S.Si., M.Si., Ph.D.	v	v	v					
32	Rekyan Regasari, ST., MT.		v	v	v				1
33	Restu Karlina Rahayu, SIP., MSi.		v	v		v			
34	Rurini Retnowati, Dr.			v			v	v	
35	Shinta Hadiyantina, Dr., SH., MH.			v				v	v
36	Sigit Kusmaryanto, Ir., MEng.	v			v		v		v
37	Sitawati, Dr. Ir., M.S.	v			v		v		

38	Siti Aisjah, S.E., M.S., Dr.	v			v			v	
39	Sri Karindah, Dr., Ir., MS.	v			v				v
40	Sri Palupi Prabandari, S.E., M.M.					v		v	
41	Sudarto, Dr. Ir., MS.	v	v		v	v		v	
42	Surachman, Prof. Dr., MSi.E.			v			v	v	
43	Titis Sari Kusuma, S.Gz.			v	v		v	v	
44	Tri Novi Kurnia Wardhani, S.ST., M.Kes.				v		v		v
45	Yoyon Supriyono, SPsi., MPsi.	v		v	v		v		
46	Yudie Wiro Eko Setyawan, S.Si.		v	v	v	v	v	v	v