IMPLEMENTASI ALGORITMA GENETIKA UNTUK MENYUSUN JADWAL MATA PELAJARAN

SKRIPSI

KONSENTRASI REKAYASA KOMPUTER

Diajukan untuk memenuhi sebagian persyaratan
Memperoleh gelar Sarjana Teknik

Disusun Oleh:

ARIF AGUNG SANTOSA

NIM. 0510630021 – 63

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN
UNIVERSITAS BRAWIJAYA
FAKULTAS TEKNIK
MALANG
2012

LEMBAR PERSETUJUAN

IMPLEMENTASI ALGORITMA GENETIKA UNTUK MENYUSUN JADWAL MATA PELAJARAN

SKRIPSI

KONSENTRASI REKAYASA KOMPUTER

Diajukan untuk memenuhi persyaratan

memperoleh gelar Sarjana Teknik

Disusun Oleh:

ARIF AGUNG SANTOSA

0510630021-63

Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	**Dosen Pembimbing II**
M. Aziz Muslim, ST., MT., Ph.D	**Adharul Muttaqin, ST., MT.**
NIP. 19741203 200012 1 001	**NIP. 19760121 200501 1 001**

LEMBAR PENGESAHAN

IMPLEMENTASI ALGORITMA GENETIKA UNTUK MENYUSUN JADWAL MATA PELAJARAN

SKRIPSI

KONSENTRASI REKAYASA KOMPUTER

Diajukan Untuk Memenuhi Persyaratan

Memperoleh Gelar Sarjana Teknik

Disusun oleh:

ARIF AGUNG SANTOSA

NIM. 0510630021 - 63

Skripsi ini telah diuji dan dinyatakan lulus pada

Tanggal 27 Juli 2012

DOSEN PENGUJI

R. Arief Setyawan, ST., MT.	**Ir. Muhammad Aswin, MT**
NIP.19750819 199903 1 001	**NIP. 19640626 199002 1 001**

Mochammad Rif'an, ST., MT
NIP. 19710301 200012 1 001

Mengetahui,
Ketua Jurusan Teknik Elektro

Dr. Ir. Sholeh Hadi Pramono, MS
NIP. 19710615 199802 1 003

ABSTRAK

Arif Agung Santosa, Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya, Juli 2012, *Implementasi Algoritma Genetika untuk Menyusun Jadwal Mata Pelajaran,,* Dosen Pembimbing: M. Aziz Muslim, ST., MT., Ph.D dan Adharul Muttaqin, ST., MT

Dalam melaksanakan kegiatan belajar mengajar, penjadwalan mata pelajaran merupakan salah satu hal penting yang harus diperhatikan oleh pihak sekolah. Penjadwalan mata pelajaran yang baik akan mendukung kelancaran proses kegiatan belajar mengajar. Penyusunan jadwal mata pelajaran pada SMAN 1 Srengat harus sesuai dengan kriteria-kriteria yang telah di tentukan, antara lain tidak boleh ada bentrok mengajar guru, jam mengajar mata pelajaran olahraga harus pagi, guru diharapkan tidak mengajar lebih dari 6 jam dalam sehari, serta mata pelajaran eksak diharapkan terletak pada jam pagi.

Untuk mengatasi permasalahan penjadwalan pada SMA Negeri 1 Srengat dapat digunakan Algoritma Genetika. Algoritma Genetika merupakan suatu pendekatan untuk mendapatkan solusi yang optimal berdasarkan teori evolusi. Pada penjadwalan mata pelajaran, algoritma genetika digunakan untuk meminimalkan kesalahan-kesalahan yang terjadi berdasarkan kriteria-kriteria yang telah ditentukan.

Pengujian dengan jumlah kelas sebanyak 23 kelas dan jumlah generasi sebanyak 10 generasi, terdapat kenaikan nilai fitness pada tiap generasi meskipun pada generasi terakhir masih terdapat sejumlah pelanggaran, sedangkan pengujian dengan jumlah kelas sebanyak 4 kelas dan jumlah generasi sebanyak 30 generasi, terjadi kenaikan nilai fitness pada tiap generasi dengan jumlah bentrok 1, jumlah jam mata pelajaran olahraga diatas jam keempat sebanyak 1, jumlah jam mata pelajaran matematika diatas jam keempat sebanyak 4, jumlah jam mata pelajaran fisika diatasjam keempat sebanyak 2, dan jumlah jam mengajar guru lebih dari 6 jam sehari sebanyak 2.

Kata Kunci: Penjadwalan, Algoritma Genetika, Mata Pelajaran

KATA PENGANTAR

Puji syukur kehadirat Allah SWT, Sang Maha Pencipta yang telah memberikan limpahan rahmat dan hidayah sehingga penulis dapat menyelesaikan skripsi dengan judul "Implementasi Algoritma Genetika Untuk Menyusun Jadwal Mata Pelajaran" sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik di Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya Malang.

Tidak banyak yang bisa penulis sampaikan kecuali ungkapan terima kasih kepada berbagai pihak yang telah dengan tulus ikhlas memberikan bimbingan, arahan, dan dukungan hingga penulisan skripsi ini dapat terselesaikan. Pada kesempatan kali ini, penulis mengucapkan banyak terima kasih kepada:

1. Bapak Dr. Ir. Sholeh Hadi Pramono, MS selaku Ketua Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya.
2. Bapak M. Azis Muslim, S.T., M.T., Ph.D selaku Sekretaris Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya.
3. Bapak Moch. Rif'an. ST., MT. selaku Ketua Program Studi Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya.
4. Bapak M. Azis Muslim, S.T., M.T., Ph.D dan Bapak Adharul Muttaqin ST., MT. selaku dosen pembimbing yang telah banyak memberikan bimbingan, masukan dan arahan dalam penyusunan skripsi ini.
5. Bapak Ir. Soemarwanto, MT selaku dosen pendamping akademik.
6. Bapak dan Ibu Dosen serta karyawan jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya.
7. Ayahanda Budi Santoso, ibunda Sri Suhartutik, istriku tercinta Salnan Ratih A dan orang-orang terdekat yang telah memberikan dukungan serta do'a dan semangat untuk terus maju hingga terselesaikan skripsi ini.
8. Bagus Royan M, Yusuf Faris, Alfian Yuda, M. Nur Lanta adena, Eka Prakarsa, dan teman-teman terdekat yang telah memberikan saran, arahan serta dukungan hingga terselesaikan skripsi ini.
9. Teman-teman Streamline 2005, teman-teman RisTIE-HME kepengurusan 09/10 dan juga teman paket E angkatan 2005 yang selalu setia dengan kebersamaan dan kepedulian.

10. Serta semua pihak yang tidak dapat penulis sebutkan satu per satu yang terlibat baik secara langsung maupun tidak langsung demi terselesaikannya skripsi ini.

Penulis menyadari sepenuhnya bahwa Skripsi ini masih banyak kekurangan. Segala kritik dan saran yang membangun akan penulis terima demi kesempurnaan skripsi ini. Harapan penulis semoga skripsi ini bermanfaat bagi pembaca dan khususnya mahasiswa jurusan teknik elektro dimasa yang akan datang.

Malang, Juli 2012

Penyusun

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dalam melaksanakan kegiatan belajar mengajar, penjadwalan mata pelajaran merupakan salah satu hal penting yang harus diperhatikan oleh pihak sekolah. Penjadwalan mata pelajaran yang baik akan mendukung kelancaran proses kegiatan belajar mengajar. Sedangkan penjadwalan yang buruk dapat menimbulkan masalah dalam proses belajar mengajar, antara lain dapat menimbulkan bentrok antar mata pelajaran, penempatan mata pelajaran yang tidak sesuai, dan lain-lain.

Algoritma genetika (AG) adalah algoritma pencarian yang didasarkan pada mekanisme seleksi alamiah dan genetika alamiah. Pada awalnya, AG digunakan sebagai algoritma pencarian parameter-parameter optimal. Tetapi, dalam perkembangannya, AG bisa diaplikasikan untuk berbagai masalah lain, seperti learning, peramalan, pemrograman otomatis, dan sebagainya. Pada bidang *soft computing*, AG banyak digunakan untuk mendapatkan nilai-nilai parameter yang optimal pada JST maupun sistem *fuzzy*.

Pada penulisan skripsi ini, data-data yang digunakan adalah data pada SMA Negeri 1 Srengat Kabupaten Blitar. Selama ini, penjadwalan di SMA Negeri 1 Srengat masih dilakukan secara manual. Selama proses penyusunan penjadwalan mata pelajaran banyak kendala yang dihadapi oleh SMA Negeri 1 Srengat, antara lain penempatan mata pelajaran yang tidak sesuai dengan waktunya, adanya guru yang memiliki beban mengajar yang tinggi dalam satu hari, beban mengajar guru yang tidak merata, serta membutuhkan waktu yang cukup lama untuk menyusun jadwal mata pelajaran yang tidak terdapat bentrok di dalamnya.

Untuk mengatasi permasalahan penjadwalan pada SMA Negeri 1 Srengat dapat menggunakan Algoritma Genetika. Algoritma Genetika merupakan suatu pendekatan untuk mendapatkan solusi yang optimal berdasarkan teori evolusi. Pada penjadwalan mata pelajaran, algoritma genetika digunakan untuk meminimalkan kesalahan-kesalahan yang terjadi (misalnya, bentrok yang terjadi, beban mengajar guru yang berlebihan, dsb) sehingga didapatkan jadwal mata pelajaran yang optimal.

Prosedur algoritma genetika dimulai dengan menetapkan suatu set solusi di awal yang disebut populasi dan melakukan perubahan dengan beberapa iterasi untuk mencapai solusi terbaik. Satu solusi disebut kromosom yang terdiri dari sekumpulan gen. Kromosom ini berupa susunan angka yang di-*generate* secara random. Kemudian, kromosom-kromosom tersebut akan berevolusi dalam beberapa tahap iterasi yang disebut dengan generasi. Generasi baru di-*generate* dengan teknik kawin silang (*crossover*) dan mutasi (*mutation*).

Kromosom-kromosom ini selanjutnya berevolusi dengan suatu kriteria kesesuaian (*fitness*) yang ditetapkan. Kromosom dengan nilai *fitness* terbaik akan dipilih sementara yang lainnya diabaikan. Selanjutnya, proses dilakukan berulang-ulang sampai dengan suatu kromosom yang mempunyai nilai *fitness* terbaik akan diambil sebagai solusi terbaik dari permasalahan.

1.2 Rumusan Masalah

Berdasarkan pada permasalahan yang telah dijelaskan pada bagian latar belakang, maka rumusan masalah dapat disusun sebagai berikut :

1. Bagaimana merancang aplikasi untuk menyusun jadwal mata pelajaran menggunakan algoritma genetika?
2. Bagaimana mengimplementasikan dan menguji aplikasi untuk menyusun jadwal mata pelajaran menggunakan algoritma genetika?

1.3 Batasan masalah

Mengingat luasnya kemungkinan pengembangan yang dapat dilakukan berkenaan dengan salah pengertian dalam penulisan tugas akhir ini dan lebih memfokuskan terhadap permasalahan, penulis menetapkan batasan-batasan permasalahan yang dikaji sebagai berikut:

1. Mata pelajaran yang digunakan adalah mata pelajaran yang terdapat pada SMA Negeri 1 Srengat Kabupaten Blitar.
2. Algoritma yang digunakan dalam aplikasi untuk menyusun jadwal mata pelajaran adalah algoritma genetika.

1.4 Tujuan

Tujuan penulisan ini adalah untuk mengimplementasikan algoritma genetika dalam menyusun jadwal mata pelajaran pada SMA Negeri 1 Srengat.

1. 5 Sistematika Penulisan

Sistematika penulisan dalam tugas akhir ini adalah sebagai berikut:

BAB I Pendahuluan

Bab ini berisi uraian mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, dan sistematika penulisan laporan tugas akhir.

BAB II Tinjauan Pustaka

Bab ini berisi uraian mengenai teori-teori yang digunakan dalam penyelesaian pembuatan aplikasi untuk menyusun jadwal mata pelajaran dengan mengimplementasikan algoritma genetika

BAB III Metode Penelitian

Bab ini berisi tentang metode penelitian yang digunakan dalam implementasi algoritma genetika dalam aplikasi untuk menyusun jadwal mata pelajaran.

BAB IV Perancangan

Bab ini berisi tentang perencanaan, perancangan, dan implementasi sistem yang meliputi pembuatan *interface* serta menerapkan teknologi yang akan digunakan.

BAB V Pengujian

Memuat proses dan hasil pengujian terhadap sistem algoritma genetika untuk penyusunan jadwal mata pelajaran yang telah dibangun.

BAB VI Kesimpulan dan Saran

Memuat kesimpulan dan saran-saran untuk pengembangan lebih lanjut dari sistem yang telah dibuat

BAB II
TINJAUAN PUSTAKA

2.1 SMA Negeri 1 Srengat

SMA Negeri 1 Srengat berdiri sejak bulan Juli tahun 1981. SMA Negeri1 Srengat berlokasi di Jl Raya Bagelenan Srengat Kelurahan Bagelenan Kecamatan Srengat Kabupaten Blitar. SMA Negeri 1 Srengat Memiliki 8 kelas untuk kelas X, 8 kelas untuk kelas XI dan 8 kelas untuk kelas XII. Jumlah total kelas yang dimiliki SMA Negeri 1 Srengat adalah 23 kelas.

Jumlah mata pelajaran pada tiap-tiap tingkat kelas pada SMA Negeri 1 Srengat berbeda-beda antara lain, kelas X terdiri dari 8 kelas dan memiliki jumlah mata pelajaran sebanyak 19 mata pelajaran, kelas XI terdiri dari 3 kelas jurusan ilmu alam dan 5 kelas jurusan ilmu sosial dan memiliki jumlah pelajaran sebanyak 15 mata pelajaran, kelas XII terdiri dari 3 kelas jurusan ilmu alam dan 5 kelas jurusan ilmu sosial dan memiliki jumlah pelajaran sebanyak 14 mata pelajaran. Sedangkan jumlah guru yang terdapat pada SMA Negeri 1 Srengat adalah 61 orang.

2.2 Penjadwalan

Kata jadwal berdasarkan Kamus Besar Bahasa Indonesia berarti pembagian waktu berdasarkan rencana pengaturan urutan kerja; daftar atau tabel kegiatan atau rencana kegiatan dengan pembagian waktu pelaksanaan yang terperinci. Sedangkan penjadwalan berarti proses, cara, perbuatan menjadwalkan atau memasukkan dalam jadwal.

2.2.1 Penjadwalan pada SMAN 1 Srengat

SMAN 1 Srengat memiliki jumlah total kelas 23 kelas yang terdiri dari 3 tingkat kelas yaitu kelas X, kelas XI, dan kelas XII dengan rincian sebagai berikut:

a. kelas X terdiri dari 8 kelas dan masing-masing kelas memiliki jumlah mata pelajaran sebanyak 19 mata pelajaran
b. kelas XI terdiri dari 4 kelas jurusan ilmu alam dan 4 kelas jurusan ilmu sosial dengan masing-masing kelas memiliki jumlah pelajaran sebanyak 15 mata pelajaran

c. kelas XII terdiri dari 3 kelas jurusan ilmu alam dan 4 kelas jurusan ilmu sosial dengan masing-masing kelas memiliki jumlah pelajaran sebanyak 14 mata pelajaran

Beban belajar kegiatan tatap muka per jam pembelajaran di SMA Negeri 1 Srengat berlangsung selama 45 menit dengan jumlah jam tatap muka perhari adalah 8 jam kecuali hari jumat yaitu 6 jam. Sedangkan beban belajar dalam satu minggu untuk tiap-tiap kelas maksimal 45 jam.

Permasalahan-permasalahan dalam penyusunan mata pelajaran pada SMAN 1 Srengat:

1. Mata Pelajaran:
   a. Jumlah jam dalam tiap-tiap mata pelajaran dapat berubah tiap semester sesuai hasil evaluasi, sedangkan jumlah total jam mata pelajaran adalah tetap.
   b. Mata pelajaran yang memiliki jumlah jam lebih dari 2 jam harus dipecah.
   c. Mata pelajaran olahraga harus diletakkan pada jam pertama hingga jam keempat
   d. Mata pelajaran eksak diutamakan pagi sedangkan mata pelajaran kesenian diletakkan pada akhir jam sekolah.

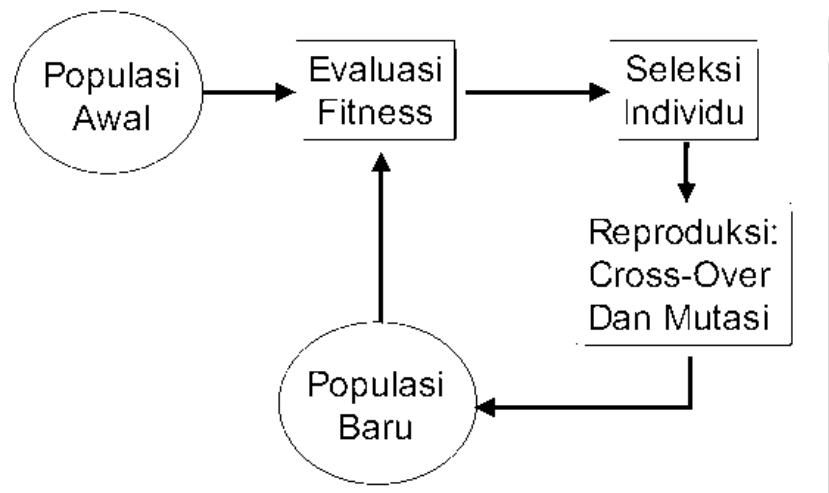
2. Guru:
   a. Lama mengajar guru dalam 1 hari tidak lebih dari 6 jam
   b. Lama mengajar guru dalam 1 minggu adalah 24 jam – 40 jam
   c. Distribusi mengajar harus merata untuk semua guru
   d. Perubahan jumlah jam dalam mata pelajaran mempengaruhi jumlah jam mengajar guru yang bersangkutan.

2.3 Algoritma Genetika

Algoritma genetika (AG) adalah algoritma pencarian yang didasarkan pada mekanisme seleksi alamiah dan genetika alamiah. Pada awalnya, AG memang digunakan sebagai algoritma pencarian parameter-parameter optimal. Tetapi, dalam perkembangannya, AG bisa diaplikasikan untuk berbagai masalah lain, seperti learning, peramalan, pemrograman otomatis, dan sebagainya. Pada bidang soft computing, AG banyak digunakan untuk mendapatkan nilai-nilai parameter yang optimal pada JST maupun sistem fuzzy.

Terdapat dua variasi algoritma genetika, yaitu: steady state dan generational replacement. Pada AG yang berjenis steady state, proses penggantian dilakukan setiap kali dihasilkan dua offspring hasil crossover. Offspring menggantikan kromosom yang nilai fitness-nya paling kecil. Dengan demikian, populasi baru yang dihasilkan selalu memiliki individu-individu yang lebih baik dibandingkan populasi yang lama.

Sedangkan pada AG berjenis generational replacement, proses penggantian dilakukan sekaligus ketika dihasilkan satu populasi baru. Untuk mempertahankan individu terbaik pada suatu generasi, diperlukan satu komponen yang di sebut elitisme, yakni pengkopian individu terbaik untuk dimasukkan sebagai anggota populasi pada generasi berikutnya. Dengan adanya elitisme, maka populasi baru yang dihasilkan selalu memiliki satu individu terbaik yang kualitasnya sama baiknya atau bahkan lebih baik dibandingkan populasi lama.



Gambar 2.1 Skema algoritma genetika

2.3.1 Komponen-Komponen Algoritma Genetika

Algoritma genetika terdiri dari delapan komponen, yaitu: skema pengkodean, nilai fitness, pindah silang (crossover), mutasi, elitisme (untuk AG berjenis generational replacement), penggantian populasi, dan kriteria penghentian.

2.3.1.1 Skema Pengkodean

Untuk dapat diproses menggunakan AG, suatu permasalahan harus dikonversi dulu kedalam bentuk individu yang diwakili oleh satu atau lebih kromosom dengan kode tertentu. Berbeda dengan teori genetika di dunia nyata yang mempresentasikan gen sebagai deretan berbasis A, C, T dan G. AG mempresentasikan gen (buatan), secara umum, sebagai bilangan real, desimal, atau biner, yaitu:

a. real-number encoding, pada skema ini, nilai gen berada dalam interval [0,R], dimana R adalah bilangan real positif

b. discrete decimal encoding, pada skema ini, setiap gen bisa berupa deretan bilangan bulat dalam interval [0,9]

c. binary encoding, setiap gen bisa berupa deretan nilai 0 atau 1

2.3.1.2 Nilai Fitness

Pada evolusi di dunia nyata, individu bernilai fitness tinggi akan bertahan hidup. Sedangkan individu bernilai fitness rendah akan mati. Pada AG, suatu individu dievalusai berdasarkan suatu fungsi tertentu sebagai ukuran nilai fitness-nya. Pada masalah optimasi, jika solusi yang dicari adalah memaksimalkan sebuah fungsi h, maka nilai fitness yang digunakan adalah nilai dari fungsi h tersebut, yakni

$$f = h \tag{2-1}$$

dengan:

f = nilai fitness.

h = fungsi fitness.

Sedangkan jika masalahnya adalah meminimalkan fungsi h, maka fungsi h tidak bisa digunakan secara langsung karena akan menghasilkan nilai fitness yang rendah. Oleh karena itu, nilai fitness untuk meminimalkan fungsi adalah

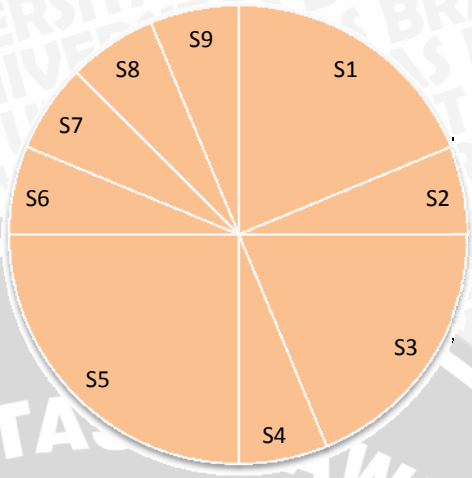
$$f = \frac{\quad}{(\quad)} \tag{2-2}$$

dengan nilai a tidak sama dengan 0 agar nilai fitness tidak bernilai tak berhingga

2.3.1.3 Seleksi Orang Tua

Proses pemilihan dua individu sebagai orang tua biasanya dilakukan secara proporsional berdasarkan nilai-nilai fitness-nya. Salah satu metode seleksi yang umum digunakan adalah Roulette-wheel. Sesuai dengan namanya, metode ini menirukan permainan roulette-wheel yang mana masing-masing individu menempati potongan lingkaran pada roulette-wheel secara proporsional sesuai dengan nilai fitness-nya.

String	Nilai Fitness
S1	0.6
S2	0.2
S3	0.6
S4	0.2
S5	0.8
S6	0.2
S7	0.2
S8	0.2
S9	0.2
Jumlah	3.2



Gambar 2.2 Contoh seleksi orang tua menggunakan roulette wheel. Individu S5 yang bernilai paling besar menempati seperempat lingkaran roda roulette. Dengan demikian, S5 memiliki peluang sebesar 0,25 untuk terpilih sebagai orang tua.

2.3.1.4 Pindah Silang

Pada proses pindah silang terjadi kombinasi pewarisan gen-gen dari induknya, gen-gen dari kedua induk dapat bercampur sehingga dihasilkan susunan kromosom yang baru. Dari proses tersebut akan dihasilkan variasi genetik.

Dua individu dipilih sebgai orang tua dengan suatu skema tertentu. Setelah didapatkan dua individu orang tua, selanjutnya ditentukan titik pindah silang secara acak. Jika diasumsikan L adalah panjang kromosom, maka titik pindah silang berada antara 1 hingga L-1. Kemudian beberapa bagian dari dua kromosom ditukar pada titik pindah silang yang dipilih. Titik pindah silang adalah titik terjadinya pertukaran gen antar dua individu orang tua. Pertukaran tersebut akan menghasilkan dua individu anak. Peluang keberhasilan operasi pindah silang dinyatakan dengan probabilitas pindah silang atau p_c. Terdapat tiga skema pindah silang yang umum digunakan, yaitu:

a. pindah silang satu titik (single-point crossover)

Pindah silang ini merupakan skema pindah silang yang paling sederhana. Titik pindah silang hanya satu dengan posisi yang dibangkitkan secara acak.

b pindah silang banyak titik (multi-point crossover)

Pada suatu masalah tertentu yang mana suatu individu terdiri dari sangat banyak gen (misalkan 1000 gen), mungkin diperlukan lebih dari satu titik pindah silang. Banyaknya titik pindah silang ini akan mempengaruhi pola pertukaran gen-gen antar individu orangtua.

c. pindah silang pola seragam (uniform crossover)

Dengan operasi pindah silang pola seragam maka komposisi gen-gen tertentu pada suatu individu dapat dipertahankan. Hal ini akan memudahkan proses pencarian solusi.

2.3.1.5 Mutasi

Mutasi diperlukan untuk mengembalikan informasi bit yang hilang akibat crossover. Mutasi diterapkan dengan probabilitas yang sangat kecil. Jika mutasi dilakukan terlalu sering, maka akan menghasilkan individu yang lemah karena konfigurasi gen yang unggul akan dirusak. Berdasarkan bagian yang bermutasi, proses mutasi data diubah menjadi 3 bagian:

a. mutasi pada tingkat kromosom: semua gen dalam kromosom berubah
b. mutasi pada tingkat gen: semua bit dalam 1 gen akan berubah. Misal gen 2 yang mengalamai mutasi.
c. mutasi pada tingkat bit: hanya satu bit yang berubah

2.3.1.6 Elitisme

Karena seleksi dilakukan secara acak, maka tidak ada jaminan bahwa suatu individu bernilai fitness tertinggi akan selalu terpilih. Kalaupun individu bernilai fitness tertingi terpilih, mungkin saja mungkin saja individu tersebut akan rusak karena proses pindah silang. Untuk menjaga agar individu bernilai fitness tertinggi tersebut tidak hilang selama evolusi, perlu dibuat satu atau dua kopinya. Prosedur ini dikenal sebagai elitism. Prosedur ini hanya digunakan pada AG berjenis generational replacement.

2.3.1.7 Penggantian Populasi

Pada AG berjenis generational replacement, N individu pada suatu generasi digantikan sekaligus oleh N individu baru hasil pindah silang dan mutasi. Untuk mempertahankan individu terbaik, diperlukan skema elitisme seperti dijelaskan di atas.

Sedangkan untuk AG berjenis steady state, dapat digunakan beberapa prosedur untuk skema penggantian populasinya, diantaranya adalah:

a. selalu mengganti individu yang memiliki nilai fitness terkecil

b. selalu mengganti individu yang paling tua

c. membandingkan anak dengan orang tua. Apabila anak memiliki nilai fitness yang lebih baik daripada salah satu atau kedua orang tua, maka anak menggantikan orang tua yang memiliki nilai fitness terendah. Skema ini dapat menjaga keanekaragaman yang lebih baik daripada skema sebelumnya. Penggantian orang tua yang memiliki beda bit yang lebih sedikit diharapkan tidak cepat menghilangkan keragaman.

2.3.1.8 Kriteria Penghentian

Terdapat berbagai macam kriteria penghentian yang bisa digunakan, tiga diantaranya adalah:

a. diberikan batasan iterasi. Apabila batas iterasi tersebut dicapai, iterasi dihentikan dan dipilih individu dengan nilai fitness tertinggi sebagai solusi terbaik.

b. diberikan batasan waktu proses AG. Kriteria ini digunakan pada sistem-sistem waktu nyata (real time systems), yang mana solusi harus ditemukan paling lama, misalkan 3 menit. Dengan demikian, AG bisa dihentikan ketika proses sudah berlangsung selama hampir 3 menit.

c. dihitung kegagalan penggantian anggota populasi yang terjadi secara berurutan sampai jumlah tertentu. Misalkan, setelah 100 iterasi tidak ada penggantian individu dalam populasi Karena individu anak yang dihasilkan selalu memiliki nilai fitness yang lebih rendah daripada orang tuanya. Dalam kondisi seperti ini, iterasi dapat dihentikan.

BAB III
METODE PENELITIAN

Dalam penyusunan skripsi ini, dirancang suatu aplikasi untuk menyusun jadwal mata pelajaran pada SMAN 1 Srengat menggunakan algoritma genetika. Metode penelitian yang digunakan pada penyusunan skripsi ini adalah

3.1 Analisis Kebutuhan

SMAN 1 Srengat memiliki jumlah total kelas 23 kelas yang terdiri dari 3 tingkat kelas yang memiliki jumlah mata pelajaran yang berbeda-beda antara lain, kelas X terdiri dari 8 kelas dan memiliki jumlah mata pelajaran sebanyak 19 mata pelajaran, kelas XI terdiri dari 4 kelas jurusan ilmu alam dan 4 kelas jurusan ilmu social dan memiliki jumlah pelajaran sebanyak 15 mata pelajaran, kelas XII terdiri dari 3 kelas jurusan ilmu alam dan 4 kelas jurusan ilmu sosial dan memiliki jumlah pelajaran sebanyak 14 mata pelajaran. Sedangkan jumlah guru yang dimiliki adalah 61 orang.

Untuk menyusun jadwal mata pelajaran yang baik, maka dalam proses penyusunan jadwal mata pelajaran didasarkan pada kriteria-kriteria yang telah ditetapkan oleh pihak SMAN 1 Srengat. Kriteria-kriteria tersebut antara lain:

a. Tidak boleh ada mata pelajaran yang bentrok

   Kriteria ini bersifat mutlak dan harus terpenuhi. Dalam jadwal mata pelajaran yang disusun, tidak boleh ada mata pelajaran yang bentrok atau seorang guru yang mengajar dalam dua kelas yang berbeda dalam waktu yang bersamaan. Jika dalam jadwal mata pelajaran yang telah tersusun terdapat bentrok mata pelajaran atau bentrok guru maka jadwal mata pelajaran tersebut harus dibuat ulang.

b. Penempatan mata pelajaran yang sesuai.

   Dalam menyusun jadwal mata pelajaran, pihak SMAN 1 Srengat menempatkan mata pelajaran olahraga dan mata pelajaran yang bersifat eksak pada pagi hari. Hal ini ditujukan agar siswa dapat mengikuti kegiatan belajar mengajar sampai akhir jam sekolah dengan baik.

c. Beban mengajar guru harus sesuai dengan standar yang ditetapkan.

Seperti yang telah disebutkan sebelumnya bahwa jumlah jam mengajar guru dalam 1 hari adalah tidak lebih dari 6 jam. Beban mengajar lebih dari 6 jam dalam sehari akan memberatkan guru dan siswa. Selain dapat membuat guru kelelahan, beban mengajar yang berat akan mengurangi konsentrasi mengajar guru sehingga situasi kelas dapat menjadi tidak kondusif.

Selama ini penyusunan jadwal mata pelajaran pada SMAN 1 Srengat masih dikerjakan secara manual, sehingga untuk menyusun jadwal yang sesuai dengan kriteria di atas membutuhkan waktu yang cukup lama. Untuk memudahkan proses penyusunan jadwal mata pelajaran, pihak sekolah memerlukan suatu perangkat lunak untuk menyusun jadwal mata pelajaran.

Dari perangkat lunak yang dibangun, pihak sekolah mengharapkan agar aplikasi dapat memenuhi kriteria-kriteria di atas dan dapat memenuhi kebutuhan seperti berikut:

a. Aplikasi penyusunan jadwal mata pelajaran dapat bekerja dengan memberikan masukan berupa data mata pelajaran, data guru dan data kelas. Hasil keluaran yang ditampilkan berupa jadwal mata pelajaran yang telah tersusun untuk tiap-tiap kelas

b. Aplikasi penyusunan jadwal mata pelajaran mudah dioperasikan

c. Aplikasi penyusunan jadwal mata pelajaran dapat dijalankan dan ditampilkan secara online.

d. *User interface* memliki tampilan berupa form-form pengisian nilai dan menggunakan bahasa indonesia.

3.2 Pengumpulan Data

Pengambilan data-data dan informasi yang diperlukan untuk membuat aplikasi sistem, dilakukan dengan wawancara dan survei langsung ke SMA Negeri 1 Srengat.

Data-data yang diperlukan untuk membuat aplikasi penyusunan jadwal mata pelajaran antara lain data mata pelajaran, data guru, dan data kelas.

3.3 Perancangan

Langkah-langkah yang dilakukan dalam tahap perancangan sistem ini antara lain:

a. Perancangan *Database*

Perancangan *database* diawali dengan merancang *Relation Database* dan membuat tabel-tabel beserta atribut masing-masing tabel di dalam database.

b. Perancangan Algoritma Genetika untuk Menyusun Jadwal Mata Pelajaran

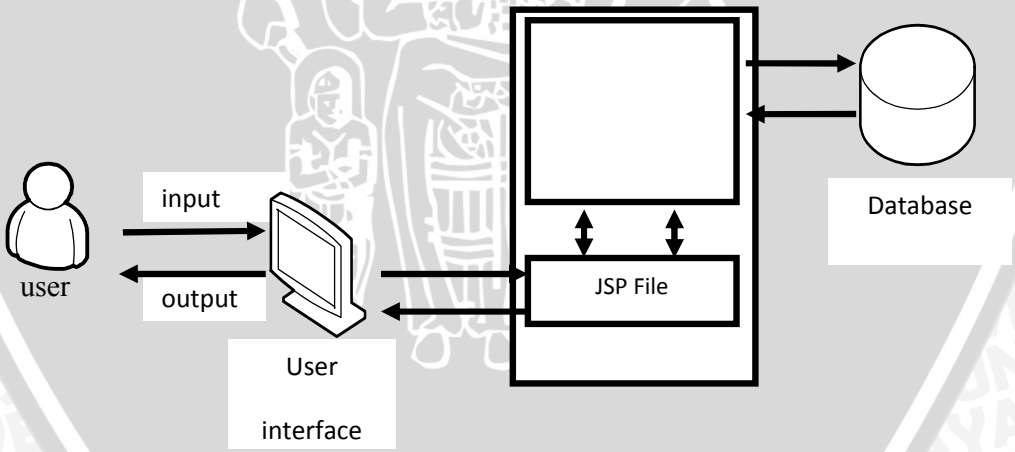
Perancangan Algoritma Genetika diawali dengan perancangan fungsi *fitness*, inisialisasi populasi, seleksi orang tua, skema pindah silang (*crossover*) dan mutasi.

c. Perancangan *Interface*

Perancangan *interface* aplikasi dibuat sesuai dengan sistem dan kebutuhan sistem serta perancangan desain *interface* sistem.

3.4 Blok Diagram Sistem

Langkah-langkah pemrosesan jadwal mata pelajaran dapat dilihat pada bagan di bawah ini.



Gambar 3.1 Blok Diagram Sistem

Keterangan:

a. *User*

Orang yang menjalankan aplikasi.

b. *User Interface*

Komponen antarmuka yang berfungsi untuk menerima data *input* dari *user* dan menampilkan hasil *output* kepada *user*. Data *input* terdiri dari data guru, data mata pelajaran, dan data kelas. Hasil *output* berupa jadwal mata pelajaran untuk tiap-tiap kelas.

c. *Web Server*

Web Server berfungsi untuk menerima permintaan data atau proses dari *user* kemudian mengerjakannya dan mengembalikan hasilnya kepada *user*. Saat memenuhi permintaan data atau proses tertentu, *Web Server* mengakses file *JSP* yang berisi *HTML,* skrip *JSP* dan komponen *java beans*. Dalam menangani permintaan data, komponen *java beans* mungkin mengambil data dari *database* dan data yang diambil biasanya diproses terlebih dahulu oleh kode program *java beans*. Hasil akhirnya akan dikirim kembali ke *web server*. *Web server* mengirim kembali hasil tersebut ke *user* dan data ditampilkan sesuai dengan *format* desain *html* yang hasilnya dapat dilihat pengguna *browser*.

d. *Database Server*

Komponen ini berfungsi untuk menyimpan data-data yang dibutuhkan untuk aplikasi penyusunan jadwal mata pelajaran dan data masukan dari user.

3.5 Parameter Algoritma Genetika

Sebelum program dijalankan, ada beberapa parameter Algoritma Genetika yang harus ditetapkan terlebih dahulu, di antaranya ukuran populasi, jumlah generasi, probabilitas *crossover*, dan probabilitas mutasi

3.5.1 Ukuran Populasi

Populasi adalah sekumpulan himpunan solusi yang akan dimasukkan ke dalam persamaan evaluasi untuk diketahui nilai *fitness*nya. Besarnya populasi pada setiap generasi akan mempengaruhi kecepatan konvergensi. Semakin besar jumlah populasi akan mengakibatkan konvergensi yang lambat, akan tetapi bila jumlah populasi awal semakin kecil maka dapat terjadi konvergensi prematur. Untuk itu, jumlah polulasi yang tepat, harus diperhitungkan dalam melakukan proses optimasi menggunakan AG.

3.5.2 Jumlah Generasi

Generasi dapat dikatakan sebagai jumlah iterasi yang dilakukan terhadap proses evaluasi tiap-tiap populasi. Seperti halnya ukuran populasi, besarnya generasi akan

mempengaruhi kecepatan konvergensi. Semakin besar jumlah generasi akan mengakibatkan konvergensi yang lambat, akan tetapi bila jumlah generasi awal semakin kecil maka dapat terjadi konvergensi prematur. Untuk itu, jumlah generasi yang tepat, juga harus diperhitungkan dalam melakukan proses optimasi menggunakan AG.

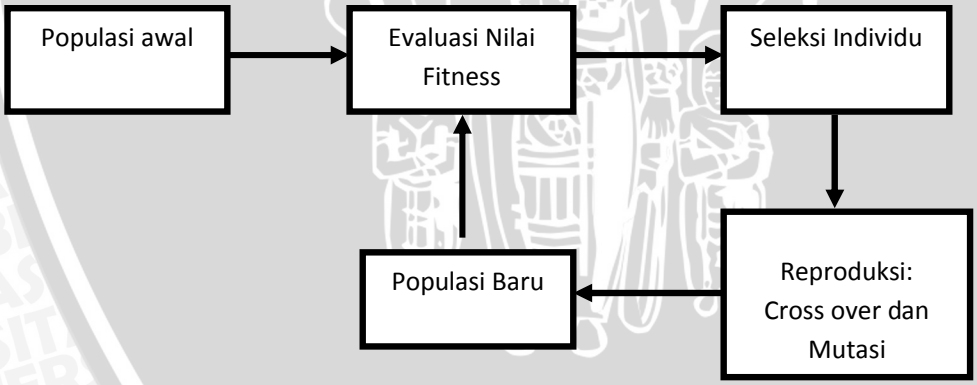
3.5.3 Probabilitas Crossover

Tidak semua kromosom akan melalui proses *crossover*. Untuk menentukan berapa persen dari total kromosom yang akan melalui *crossover*, maka ditentukanlah probabilitas *crossover* P_c. Probabilitas *crossover* diatur dengan nilai yang bervariasi, mulai dari 0,1 sampai dengan 1,0.

3.5.4 Probabilitas Mutasi

Probabilitas mutasi digunakan untuk memilih posisi gen yang akan dimutasi. Probabilitas mutasi dilambangkan dengan p_m dan nilainya sangat kecil. Probabilitas mutasi 0,01 berarti rata-rata 0,01 atau 1% dari total bit dalam suatu populasi akan mengalami mutasi

3.6 Sistem Kerja Algoritma Genetika



Gambar 3.2 Sistem Kerja Algoritma Genetika

Keterangan:

1. Pada awal pencarian solusi, algoritma genetika akan membangkitkan sekumpulan alternatif solusi yang disebut populasi. Dengan adanya populasi ini, maka algoritma genetika melakukan proses pencarian solusi dari beberapa alternatif solusi sekaligus. Agar dapat diproses oleh algoritma genetika, maka alternatif solusi tersebut dikodekan terlebih dahulu kedalam kromosom. Masing-masing kromosom berisi sejumlah gen

yang mengodekan informasi. Gen-gen yang terdapat dalam kromosom tersebut bernilai integer.

2. Alternatif solusi (kromosom) pada populasi kemudian di evaluasi berdasarkan fungsi *fitness* yang telah di tentukan. Hasil dari evaluasi tersebut berupa nilai *fitness* dari kromosom. Fungsi *fitness* dibuat berdasarkan batasan-batasan yang telah di tetapkan. Batasan-batasan tersebut antara lain:

a. Tidak boleh terjadi bentrok, baik waktu maupun ruangan untuk guru yang sama.

b. Mata pelajaran yang memiliki jumlah jam lebih dari 2 jam harus dipecah.

c. Mata pelajaran olahraga harus diletakkan pada jam pertama hingga jam keempat

d. Mata pelajaran eksak (matematika dan fisika) diutamakan pagi

e. Lama mengajar guru dalam satu hari tidak lebih dari 6 jam

Batasan-batasan terebut kemudian di kategorikan berdasarkan *hard constraint* (batasan keras) dan *soft constraint* (batasan lunak). Setelah dikategorikan, batasan-batasan tersebut diberi nilai prioritas sesuai kategori yang telah ditetapkan. Semakin batasan tersebut tidak boleh dilanggar maka nilai prioritasnya semakin tinggi.

Fungsi *fitness* pada aplikasi ini di tujukan untuk meminimalkan pelanggaran-pelanggaran terhadap batasan-batasan diatas. Sehingga fungsi *fitness* akan memiliki persamaan

$$F = \frac{1}{(\ h1+h2+h3+h4+h5+\ )} = \frac{1}{h+} \qquad (3\text{-}1)$$

dengan:

h1 = jumlah jadwal yang mengalami bentrok dikalikan nilai prioritasnya

h2 = jumlah mata pelajaran olahraga diatas jam keempat dikalikan nilai prioritasnya

h3 =jumlah jadwal mata pelajaran matematika diatas jam keempat dikalikan nilai prioritasnya

h4 = jumlah jadwal mata pelajaran fisika diatas jam keempat dikalikan nilai prioritasnya

h5 = jumlah mata guru yang mengajar lebih dari 6 jam dalam 1 hari dikalikan nilai prioritanya

h = jumlah batasan yang dilanggar dikalikan nilai prioritasnya dan

a = bilangan yang bernilai sangat kecil untuk menghindari kemungkinan terjadinya pembagian dengan nilai 0.

3. Setelah nilai *fitness* dari kromosom-kromosom yang terdapat pada populasi telah dihasilkan, maka akan dipilih dua individu yang digunakan sebagai orang tua. Seleksi orangtua dilakukan dengan metode roulette-wheel.

4. Selanjutnya ditentukan titik pindah silang secara acak dari dua individu yang terpilih sebagai orang tua. Dari proses pindah silang tersebut akan dihasilkan individu baru (*offspring*). Pada algoritma ini digunakan pindah silang banyak titik *(multi-point crossover)* yaitu proses pindah silang dengan menggunakan 1 atau lebih titik pindah silang.

5. Pada individu baru yang telah dihasilkan kemudian diterapkan mutasi. Mutasi dilakukan dengan cara *swap mutation*. Dua gen dipilih secara acak untuk dimutasi nilainya, dengan cara menukarkan posisi kedua gen tersebut dalam interval nilai-nilai gen yang diizinkan.

3.7 Implementasi Sistem

Membangun aplikasi sistem dengan bahasa pemrograman *java* berdasarkan perancangan yang telah dilakukan sebelumnya. Pemrograman yang digunakan dengan pendekatan berorientasi objek.

3.8 Pengujian

Pengujian dilakukan untuk mengetahui apakah sistem yang telah dibangun masih terdapat kesalahan dan apakah hasil keluaran sistem sesuai dengan persyaratan dan kebutuhan yang diminta *user*. Pengujian dilakukan dengan cara pengujian *white box* dan pengujian *black box*. Pengujian *white box* berfokus pada struktur control program. *Test case* dilakukan untuk memastikan bahwa semua statemen pada program telah dieksekusi paling tidak satu kali selama pengujian dan bahwa semua kondisi logis telah diuji.

Pengujian *black box* adalah pengujian aspek fundamental sistem tanpa memperhatikan struktur logika internal perangkat lunak. Ujicoba *black box* digunakan untuk mendemonstrasikan fungsi perangkat lunak yang dioperasikan, apakah input dapat diterima dengan benar dan output yang dihasilkan sesuai dengan kebutuhan dan keinginan *user*.

3.9 Kesimpulan dan Saran

Setelah melakukan perancangan, pembuatan aplikasi untuk menyusun jadwal mata pelajaran, dan melakukan pengujian aplikasi untuk menyusun jadwal mata pelajaran maka dilakukan pengambilan kseimpulan dan saran untuk pengembangan sistem lebih lanjut.

BAB IV
PERANCANGAN

Di dalam bab perancangan akan dijelaskan lebih detail tentang proses pengembangan perangkat lunak yaitu sekumpulan metode dan praktik yang digunakan untuk menghasilkan produk perangkat lunak. Model yang digunakan dalam pembuatan aplikasi penjadwalan mata pelajaran adalah model *waterfall*. Model ini menyarankan pengembangkan perangkat lunak dimulai dengan tahap analisis, perancangan, pemrograman, pengujian dan pemeliharaan.

4.1 Analisis dan Perancangan

4.1.1 Identifikasi Masalah

SMA Negeri 1 Srengat memiliki jumlah kelas sebanyak 23 kelas dengan jumlah mata pelajaran pada tiap-tiap tingkat kelas yang berbeda-beda antara lain, kelas X terdiri dari 8 kelas dan memiliki jumlah mata pelajaran sebanyak 19 mata pelajaran, kelas XI terdiri dari 4 kelas jurusan ilmu alam dan 4 kelas jurusan ilmu sosial dan memiliki jumlah pelajaran sebanyak 15 mata pelajaran, kelas XII terdiri dari 3 kelas jurusan ilmu alam dan 4 kelas jurusan ilmu sosial dan memiliki jumlah pelajaran sebanyak 14 mata pelajaran. Sedangkan jumlah guru yang terdapat pada SMA Negeri 1 Srengat adalah 61 orang.

Jumlah kelas yang cukup banyak dan jumlah guru yang terbatas menyebabkan diperlukan suatu penjadwalan mata pelajaran yang efektif sehingga mempermudah proses belajar mengajar. Berdasarkan permasalahan-permasalahan yang telah disebutkan sebelumnya maka penjadwalan harus sesuai dengan standar sebagai berikut:

1. Jumlah mata pelajaran harus sesuai dengan jumlah jam mata pelajaran keseluruhan dan jumlah jam tiap-tiap mata pelajaran berdasarkan tingkat kelas
2. Mata pelajaran yang memiliki jumlah jam lebih dari 2 harus dipecah
3. Mata pelajaran olah raga harus ditempatkan pada jam pertama hingga jam keempat
4. Mata pelajaran eksak diutamakan terletak pada jam pertama hingga jam keempat

5. Lama mengajar guru dalam 1 hari tidak lebih dari 6 jam
6. Distribusi mengajar harus merata untuk semua guru pada tiap-tiap mata pelajaran

4.1.2 Perancangan Algoritma Genetika

4.1.2.1 Proses Pembentukan individu

Individu berbentuk terdiri dari slot-slot sesuai dengan banyaknya slot dalam 1 minggu dikalikan dengan jumlah kelas, yaitu 23 x 23 = 529 slot. Slot-slot tersebut mewakili tabel mata pelajaran sbb:

Jam\Hari	Senin	Selasa	Rabu	Kamis	Jumat	Sabtu
07.00-08.30	1	5	9	13	17	20
08.30-09.30	2	6	10	14	18	21
10.00-11.30	3	7	11	15	19	22
11.30-13.00	4	8	12	16		23

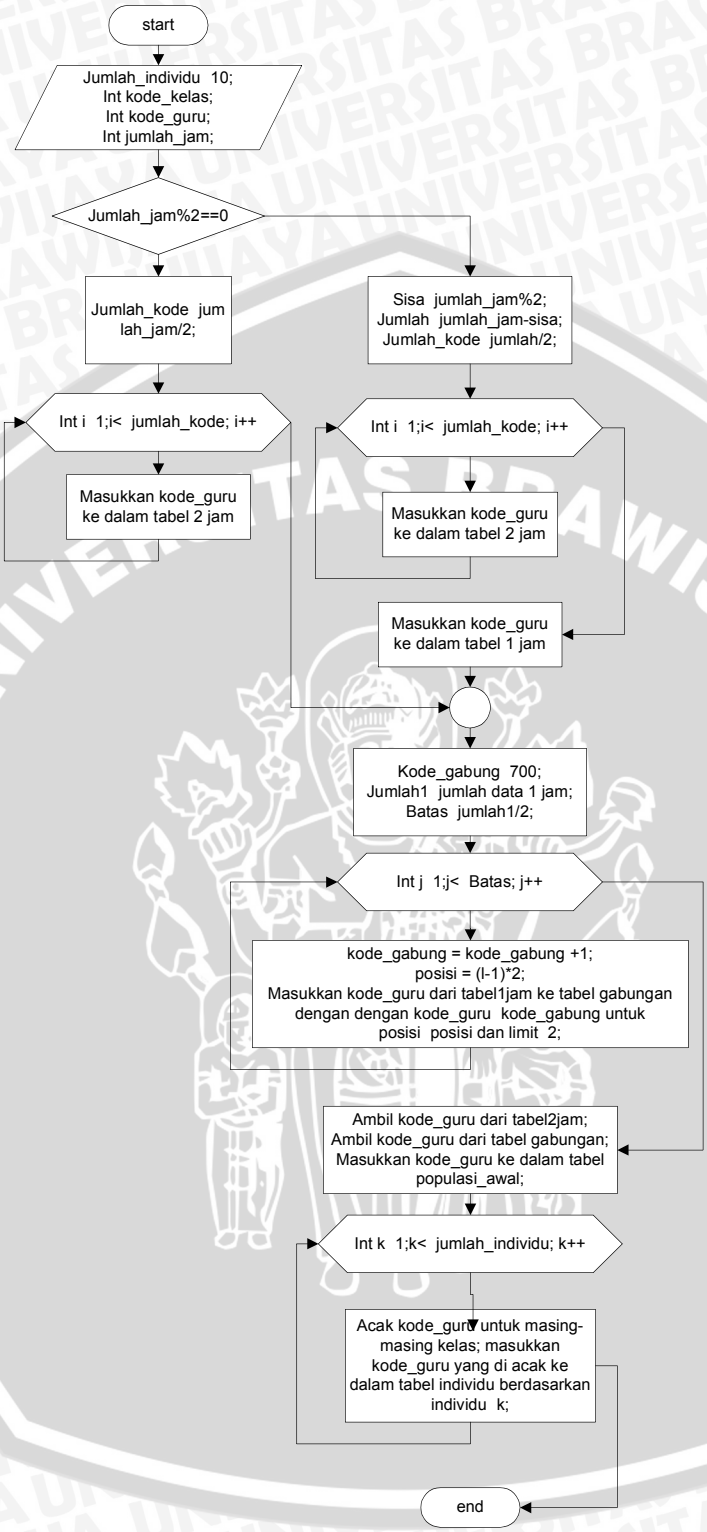
Bentuk individu:

21	**1**	**5**	**23**	**4**	**15**	**12**	**20**	**3**	**..**	**..**	**52**
gen 1	gen 2	gen 3	gen 4	gen 5	gen 6	gen 7	gen 8	gen 9	..	…	gen 529

Pada bentuk individu diatas, nomor urut gen menunjukkan nomor slot serta jam mengajar guru sedangkan nilai gen menunjukkan kode guru pada tabel guru. Pada tabel serta bentuk individu di atas, satu slot mewakili 2 jam mata pelajaran.

Langkah-langkah pembentukan individu diatas adalah sebagai berikut:

1. Ambil data pada tabel pembagian_guru pada database
2. Pecah data yang memiliki jumlah jam lebih dari 2 jam menjadi masing-masing 2 jam
3. Setelah di pecah, masukkan data yang nilainya 2 jam kedalam tabel dua_jam dan data yang nilainya 1 jam kedalam tabel satu_jam
4. Untuk data yang nilainya 1 jam, masukkan kedalam tabel gabungan untuk digabung menjadi 2 jam dengan kode_gabung lebih dari 700
5. Ambil data pada tabel dua_jam dan tabel gabungan dan masukkan ke dalam tabel populasi_awal
6. Dari tabel populasi_awal, ambil data secara acak sebanyak 23 data secara urut dan masukkan ke tabel individu.



Gambar 4.1 Flowchart pembentukan individu

4.1.2.2 Nilai Fitness

Setelah individu-individu terbentuk, individu dievaluasi berdasarkan fungsi fitness untuk mendapatkan nilai fitness-nya. Fungsi fitness pada aplikasi ini di tujukan untuk meminimalkan pelanggaran-pelanggaran terhadap batasan-batasan diatas. Sehingga fungsi fitness akan memiliki persamaan

$$F = \frac{}{((\quad)\quad)} = \frac{}{(\quad)} \qquad (4\text{-}1)$$

dengan:

h1 = jumlah jadwal yang mengalami bentrok dikalikan nilai prioritasnya

h2 = Jumlah mata pelajaran olahraga diatas jam keempat dikalikan nilai prioritasnya

h3 = jumlah mata pelajaran matematika diatas jam keempat dikalikan nilai prioritasnya

h4 = jumlah mata pelajaran fisika diatas jam keempat dikalikan nilai prioritasnya

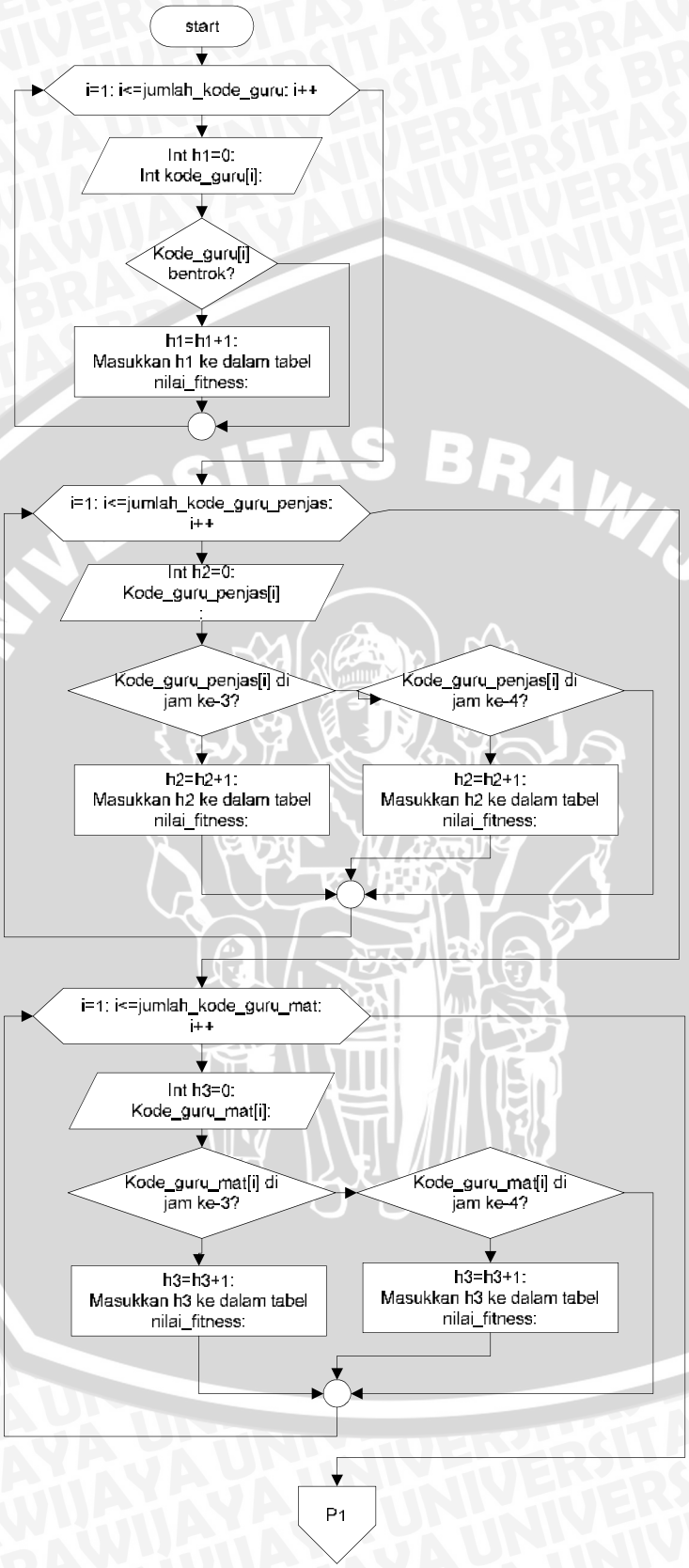
h5 = jumlah jam mengajar guru lebih dari 6 jam

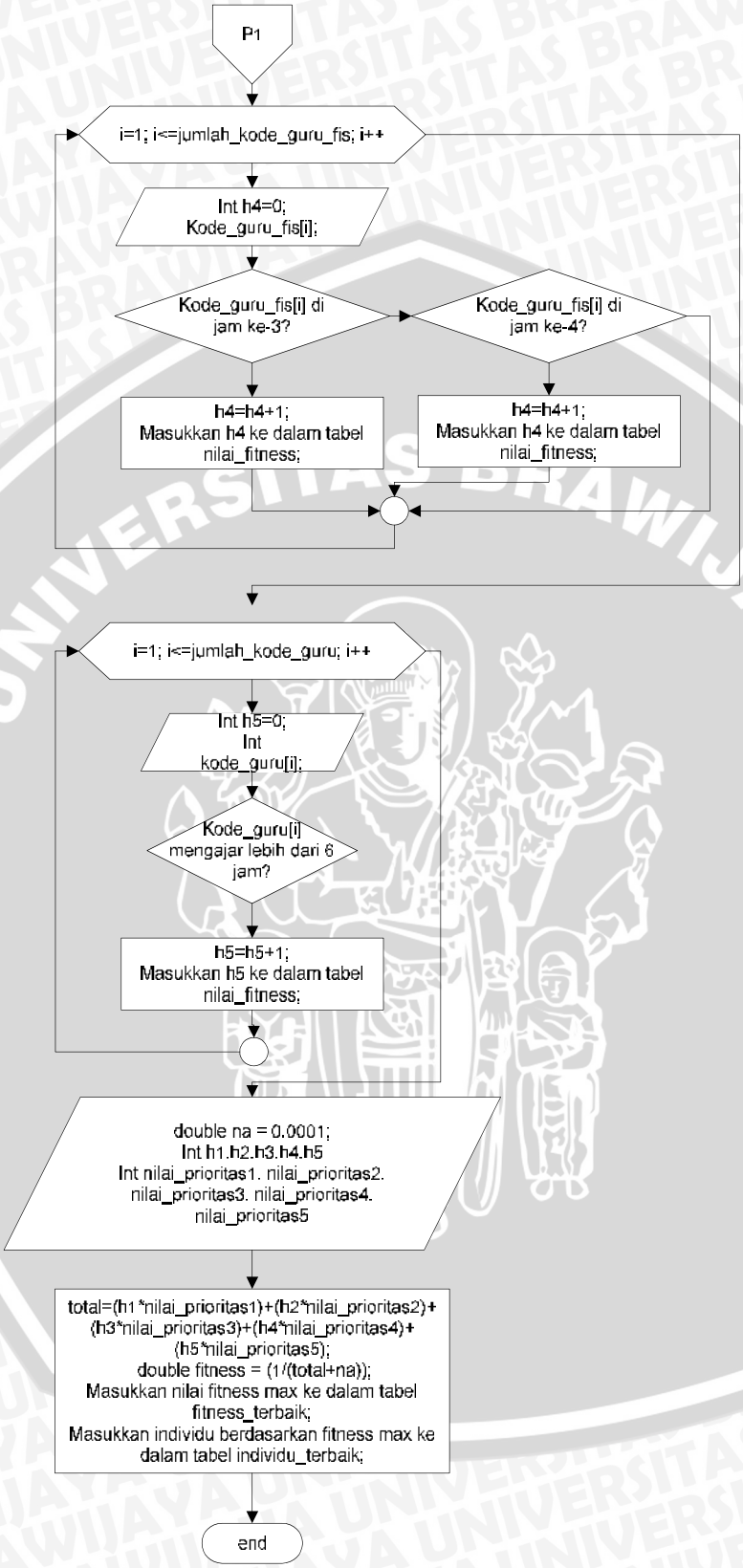
h = jumlah batasan yang dilanggar dikalikan nilai prioritasnya dan

a = bilangan yang bernilai sangat kecil untuk menghindari kemungkinan terjadinya pembagian dengan nilai 0.

Tabel 4.1 Nilai Prioritas

Batasan	Nilai Prioritas
h1	1000
h2	700
h3	400
h4	400
h5	200


start
i=1; i<=jumlah_kode_guru; i++
Int h1=0;
Int kode_guru[i];
Kode_guru[i] bentrok?
h1=h1+1;
Masukkan h1 ke dalam tabel nilai_fitness;
i=1; i<=jumlah_kode_guru_penjas; i++
Int h2=0;
Kode_guru_penjas[i]
;
Kode_guru_penjas[i] di jam ke-3?
Kode_guru_penjas[i] di jam ke-4?
h2=h2+1;
Masukkan h2 ke dalam tabel nilai_fitness;
h2=h2+1;
Masukkan h2 ke dalam tabel nilai_fitness;
i=1; i<=jumlah_kode_guru_mat; i++
Int h3=0;
Kode_guru_mat[i];
Kode_guru_mat[i] di jam ke-3?
Kode_guru_mat[i] di jam ke-4?
h3=h3+1;
Masukkan h3 ke dalam tabel nilai_fitness;
h3=h3+1;
Masukkan h3 ke dalam tabel nilai_fitness;
P1



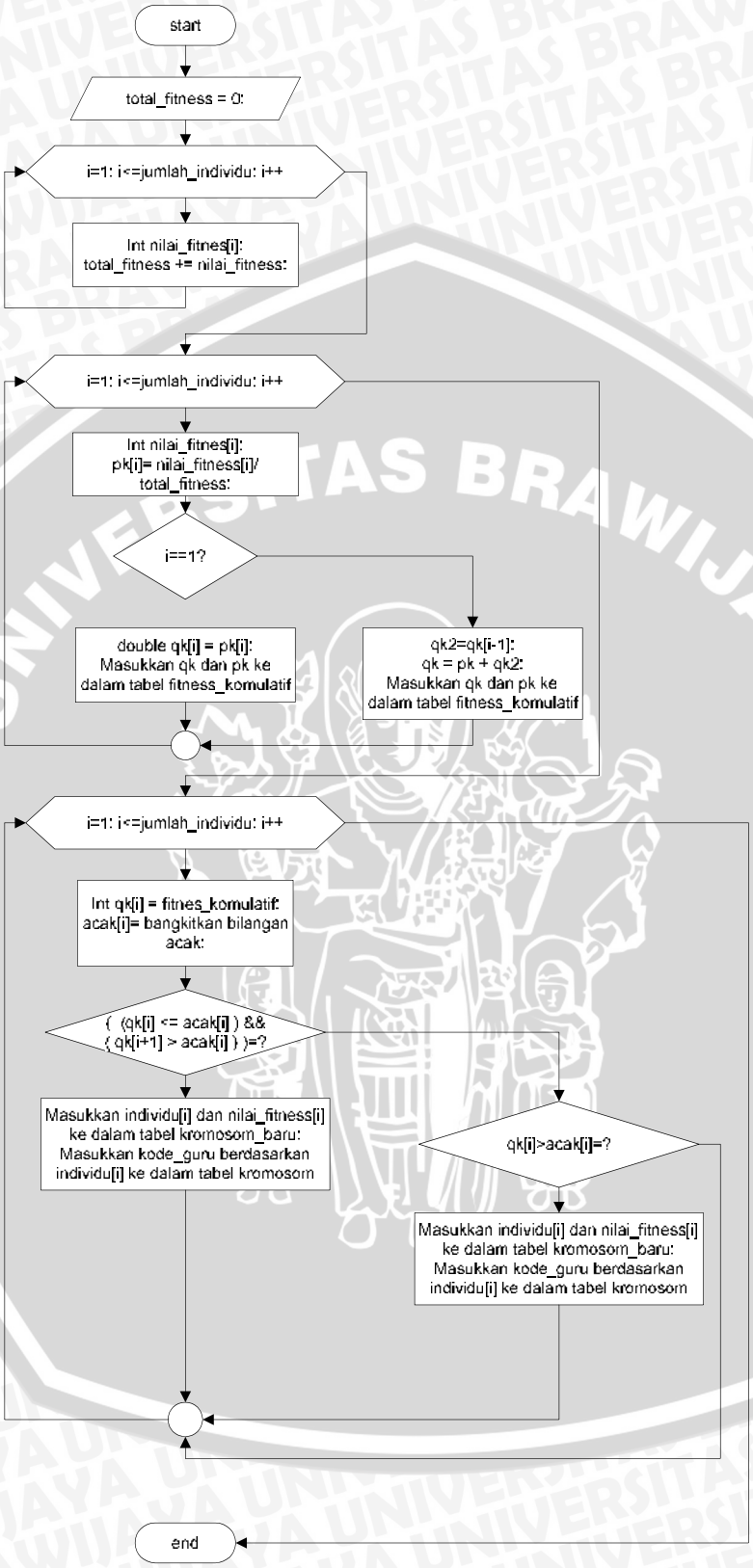
Gambar 4.2 Flowchart Penghitungan Nilai Fitness

4.1.2.3 Seleksi Orangtua

Setelah masing-masing individu dalam populasi dievaluasi dan didapatkan nilai *fitness*nya, maka proses selanjutnya adalah melakukan seleksi orangtua. Dari sekumpulan individu yang ada dalam suatu populasi, akan dipilih individu-individu sebagai orangtua yang akan disilangkan untuk proses mendapatkan generasi selanjutnya. Proses seleksi orang tua dilakukan dengan menggunakan algoritma seleksi *roulette-wheel*. Sesuai dengan namanya, algoritma ini menirukan permainan *roulette-whhel* yang mana masing-masing individu menempati potongan lingkaran pada roda *roulette* secara proporsional sesuai dengan nilai fitnessnya.

Dalam metode *roulette-wheel* nilai *fitness* tiap individu dinyatakan dalam sektor luasan lingkaran, semakin baik kualitas individu maka semakin luas sektor yang dimilikinya. Algoritma untuk seleksi induk dengan metode *roulette-wheel* adalah sebagai berikut:

1. Menghitung total fitness (F):
   TotFitness = ∑ ; k=1,2,…, popsize
2. Menghitung nilai fitness relative tiap individu:
   = / TotFitness
3. Menghitung Fitness Komulatif
   - =
   - = + ; k=2, 3. …, popsize
4. Memilih induk yang akan menjdai kandidat untuk direkombinasi dengan cara:
   - Membangkitan bilangan acak *r*
   - Jika ≤ *r* dan > *r* , maka kromosom (k+1) terpilih sebagai kandidat induk
   - Jika < , maka kromosom *k* terpilih sebagai kandidat induk

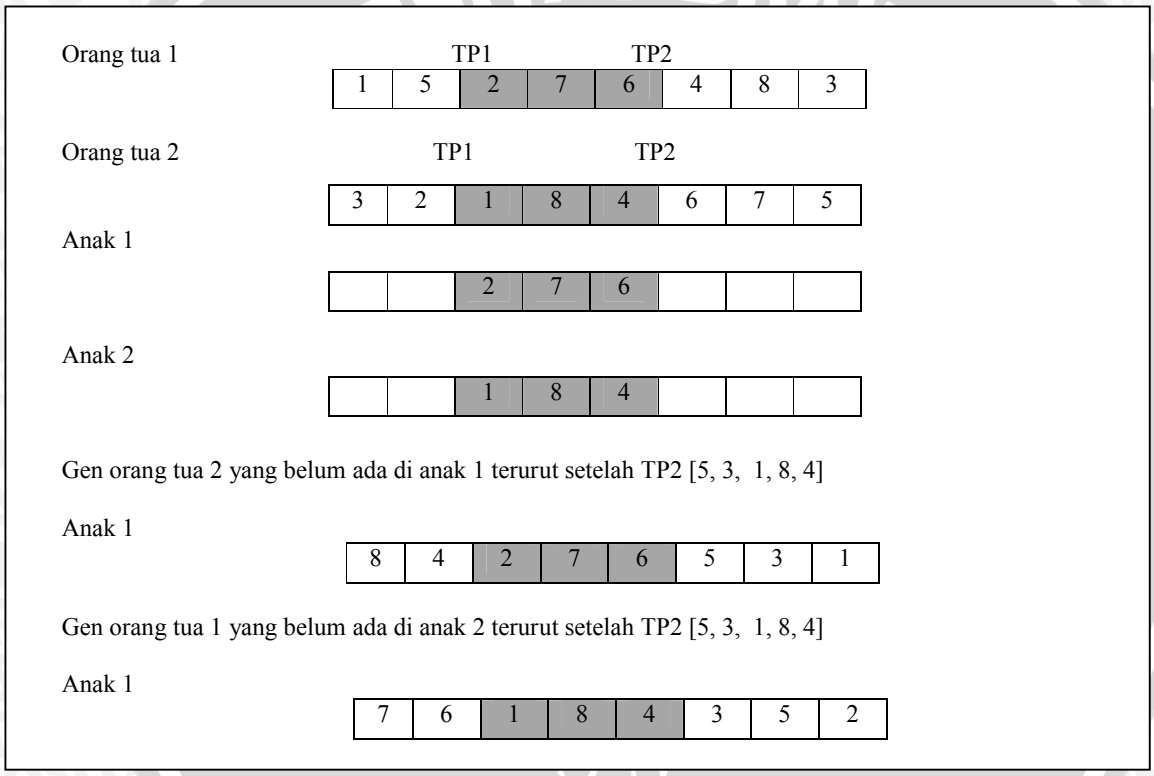


Gambar 4.3 Flowchart Seleksi Induk

4.1.2.4 Rekombinasi

Setelah mendapatkan N (ukuran populasi) individu di *mating pool* sebagai orang tua, AG menjalankan operator rekombinasi (*crossover*/pindah silang) terhadap orang tua berdasarkan probabilitas tertentu.

Pada kasus penyusunan jadwal mata pelajaran ini digunakan rekombinasi *multi-point crossover* dengan metode *order crossover*. Pada metode ini, satu bagian individu dipertukarkan tetapi urutan gen secara relatif yang bukan bagian dari individu tersebut tetap dijaga.



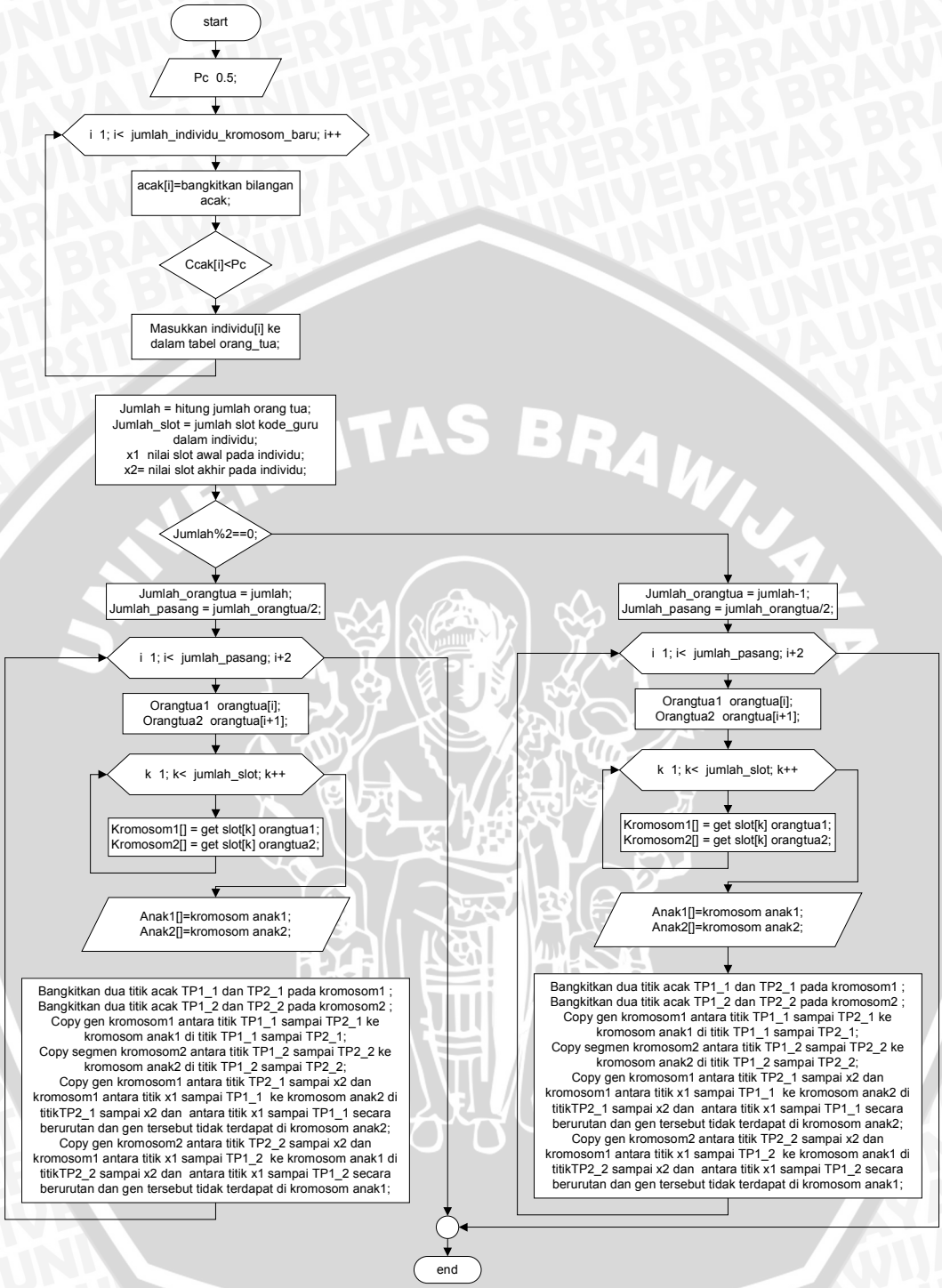
Gambar 4.4 Rekombinasi menggunakan order crossover. Kedua individu anak yang dihasilkan bernilai valid, tidak ada dua gen yang bernilai sama.

Proses rekombinasi diatas adalah mula-mula dua buah titik TP1 dan TP2 dibangkitkan secara acak untuk mengambil satu segmen individu orang tua. Misalkan, pembangkitan dua titik menghasilkan TP1=2 dan TP2=5 sehingga dihasilkan dua segmen individu (gen 3, gen 4, gen 5) dari orang tua 1 dan orang tua 2. Selanjutnya anak 1 mendapatkan gen-gen dari segmen individu orang tua 1 berupa (2, 7, 6) dan anak 2 mendapatkan gen-gen dari segmen individu orang tua 2 berupa (1, 8, 4). Selanjutnya semua gen pada orang tua 2 yang belum ada di anak 1 dikumpulkan secara berurutan

setelah TP2, sehingga dihasilkan (5, 3, 1, 8, 4). Kemudian, kumpulan gen dimasukkan secara berurutan ke dalam anak 1 dimulai dari posisi setelah TP2. Sehingga dihasilkan anak 1 yang memiliki gen (8, 4, 2, 7, 6, 5, 3, 1). Hal sama juga terjadi untuk menghasilkan anak 2.

Akan tetapi dikarenakan pada individu penjadwalan nomor urut slot juga menunjukkan nama kelas dan tingkatan kelas, maka apabila proses di atas dilakukan maka ada kemungkinan nilai individu anak yang dihasilkan tidak valid. Hal tersebut dapat terjadi apabila ada nilai slot untuk mata pelajaran kelas X-A tertukar ke posisi slot untuk mata pelajaran kelas X-D atau nilai slot untuk mata pelajaran kelas X-A tertukar ke posisi slot untuk mata pelajaran kelas XI-IA-1. Sehingga untuk menjaga agar individu anak yang dihasilkan tetap valid, maka TP1 dan TP2 dibangkitkan pada range slot tiap-tiap kelas dan gen-gen yang dipertukarkan adalah gen-gen yang ada pada tiap-tiap range slot kelas tersebut.

Di dunia nyata, tidak semua pasangan bisa menghasilkan anak. Hal ini diadopsi AG dengan menggunakan probabilitas rekombinasi yang dilambangkan dengan Pc. Biasanya AG menggunakan Pc dalam interval 0,5-1. Jika menggunakan Pc sbesar 0,7 berarti hanya 70% dari jumlah pasangan orang tua yang direkombinasi dan menghasikan anak. Anak yang dihasilkan kemudian dikopikan ke generasi berikutnya. Jika pasangan orang tua tidak direkombinasi (tidak menghasilkan anak) maka kedua kromosom tersebut dikopikan ke generasi berikutnya.

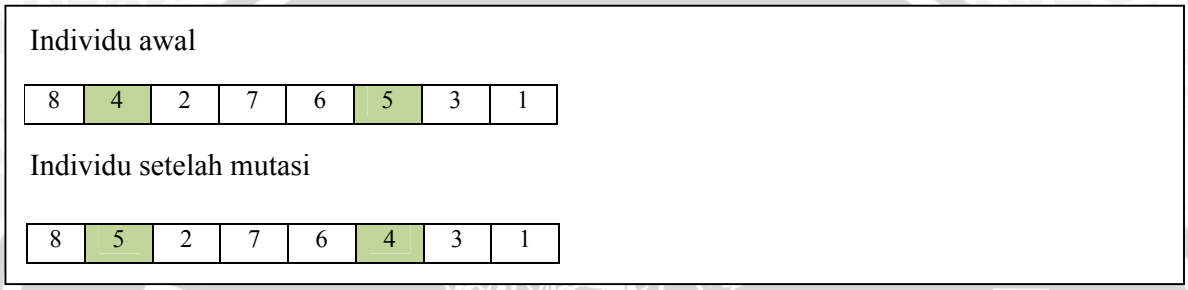


Gambar 4.5 Flowchart Proses Rekombinasi (Pindah Silang)

4.1.2.5 Mutasi

Setelah tahap rekombinasi terhadap semua pasangan orang tua pada *mating pool* yang menghasilkan N (ukuran populasi) anak (individu baru), maka AG menjalankan operator mutasi terhadap setiap individu baru tersebut.

Pada kasus penjadwalan, metode mutasi yang digunakan adalah mutasi pertukaran (*swap mutation)*. Mutasi ini berjalan dengan cara menukarkan posisi dua gen yang telah dipilih secara acak.



Gambar 4.6 Mutasi pertukaran (swap mutation). Pemilihan dua posisi gen secara acak menghasilkan posisi 4 dan 5. Sehingga gen bernilai 4 ditukarkan dengan gen bernilai 5

Mutasi pertukaran dapat menghasilkan anak yang valid. Akan tetapi sama seperti pada proses rekombinasi, sebelum pembangkitan dua posisi gen secara acak, maka terlebih dahulu ditentukan terlebih dahulu jumlah individu yang akan mengalami mutasi yang ditentukan oleh probabilitas individu termutasi (Ps). Setelah individu yang mengalami mutasi terpilih, kemudian menentukan secara acak range slot dalam kelas yang akan mengalami mutasi. Di dalam range tersebut dibangkitkan dua titik secara acak, kemudian nilai gen pada titik tersebut dipertukarkan. Cara ini digunakan agar tidak terjadi pertukaran nilai gen antar kelas atau antar tingkatan kelas yang menyebabkan nilai gen menjadi tidak valid.

Selain proses mutasi, hal penting yang perlu diperhatikan adalah jumlah gen yang mengalami mutasi. Jumlah gen yang mengalami mutasi ditentukan oleh probabilitas mutasi Pm. Probabilitas mutasi 0,001 maka berarti 0,001 atau 0,1% dari total gen dalam satu populasi akan mengalami mutasi.

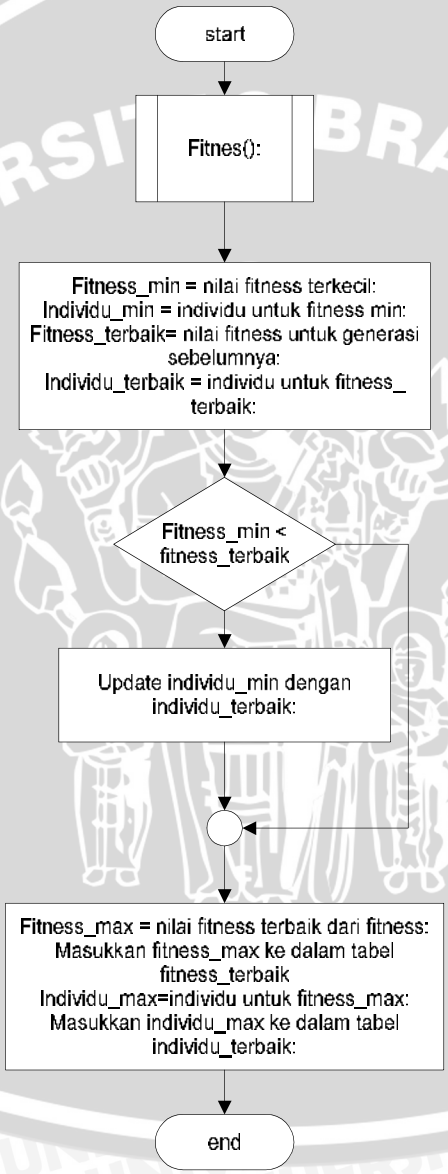
Misalkan pada kasus ini, jumlah gen dalam satu individu adalah 529 dan ukuran populasi adalah 10. Maka total gen dalam satu populasi = 529 x 10 = 5290 gen. Sehingga apabila Pm = 0,001 maka diharapkan ada sekitar 5,29 (5) mutasi dalam setiap generasinya.

Gambar 4.7 Flowchart Proses Mutasi

4.1.2.6 Elitisme

Elitisme merupakan metode untuk mempertahankan individu yang memiliki nilai fitness tertinggi. Metode ini bertujuan agar solusi terbaik (kromosom ber-fitness tertinggi) yang pernah di capai tidak hilang.

Proses elitisme pada penulisan ini adalah ganti individu dengan nilai fitness terendah pada generasi saat ini dengan individu dengan nilai fitness tertinggi pada generasi sebelumnya.

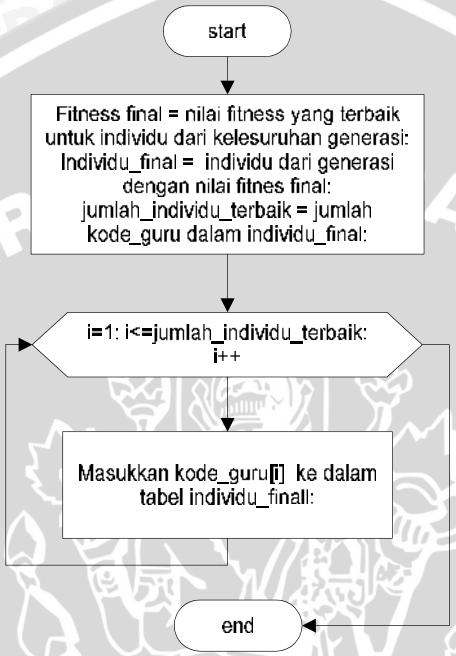


Gambar 4.8 Flowchart Proses Elitisme

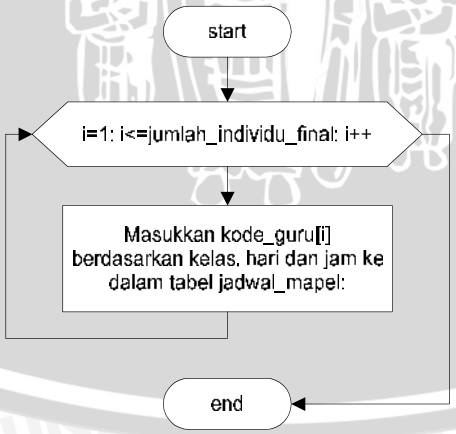
4.1.2.7 Individu Final

Individu final merupakan individu yang memiliki nilai fitness tertinggi dari total generasi yang terjadi. Individu inilah yang nantinya akan membentuk jadwal mata pelajaran.

Setelah individu final terpilih maka proses selanjutnya adalah mengambil nilai-nilai slot dari individu tersebut untuk membentuk jadwal mata pelajaran



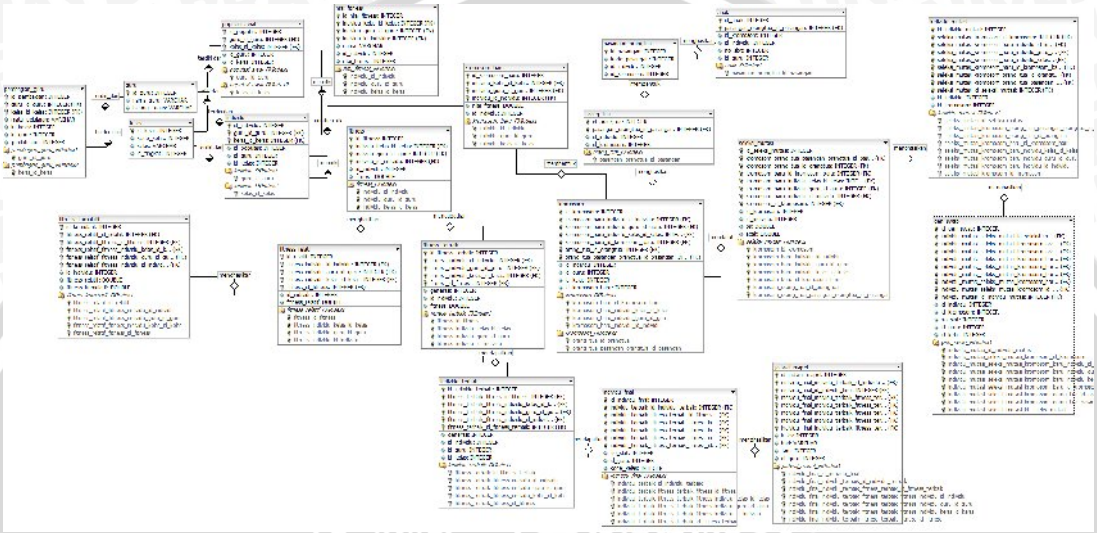
Gambar 4.9 Flowchart Pengambilan Individu Final



Gambar 4.10 Flowchart Pembentukan Jadwal Mata Pelajaran dari Individu Final

4.1.3 Perancangan Database

Metodologi perancangan basis data adalah kumpulan terknik terorganisasi untuk pembuatan rancangan basisdata. Teknik terorganisasi ini merupakan kumpulan tahapan-tahapan yang memiliki aturan-aturan terurut. Teknik yang digunakan pada perancangan basisdata terdiri dari pembuatan relasional antar tabel.



Gambar 4.11 Relasional Antar Tabel

4.2 Implementasi

Implementasi merupakan tahap dimana aplikasi sistem siap dioperasikan pada tahap yang sebenarnya yang merupakan hasil dari perancangan, sehingga akan diketahui apakah aplikasi sistem yang telah dibuat benar-benar sesuai dengan yang direncanakan. Pada implementasi perangkat lunak ini akan dijelaskan bagaimana program aplikasi ini akan bekerja. Berikut ini implementasi sistem untuk aplikasi penjadwalan mata pelajaran menggunakan metode algoritma genetika.

4.2.1 Implementasi GA ke Penjadwalan

Jumlah generasi yang digunakan dalam proses GA adalah 2 dan data yang digunakan dalam proses pembentukan jadwal mata pelajaran adalah data kelas dan kode guru yang sekaligus mewakili mata pelajaran. Agar dapat diproses maka data tersebut disatukan menjadi data pembagian guru, dengan kelas berjumlah 4 dan guru berjumlah 20 orang berikut data input berupa pembagian guru:

Tabel 4.2 Pembagian Guru

mata_pelajaran	kode_kelas	kode_guru	jumlah_jam
Bimbingan dan Konseling	101	56	1
Matematika	101	25	4
Matematika	101	25	1
Kimia	101	38	3
Agama	101	2	3
Ekonomi	101	41	4
Ekonomi	101	41	1
Seni rupa	101	53	2
Biologi	101	33	3
Kewarganegaraan	101	5	2
Fisika	101	30	2
Sejarah	101	15	2
B. Indonesia	101	8	4
B. Inggris	101	16	2
Olahraga	101	21	2
Geografi	101	49	2
B. Jerman	101	52	2
Sosiologi	101	46	2
TIK	101	60	2
Bimbingan dan konseling	102	56	1
Matematika	102	25	4
Matematika	102	25	1
Kimia	102	38	3
Agama	102	2	3
Ekonomi	102	41	4
Ekonomi	102	41	1
Seni rupa	102	53	2
Biologi	102	33	3
Kewarganegaraan	102	5	2
Fisika	102	30	2
Sejarah	102	15	2
B. Indonesia	102	8	4
B. Inggris	102	16	2
Olahraga	102	21	2
Geografi	102	49	2
B. Jerman	102	52	2
Sosiologi	102	46	2

Tabel 4.2 Pembagian Guru

mata_pelajaran	kode_kelas	kode_guru	jumlah_jam
TIK	102	60	2
Bimbingan dan konseling	103	56	1
Matematika	103	26	4
Matematika	103	25	1
Kimia	103	38	3
Agama	103	2	3
Ekonomi	103	41	4
Ekonomi	103	41	1
Seni rupa	103	53	2
Biologi	103	33	3
Kewarganegaraan	103	5	2
Fisika	103	30	2
Sejarah	103	15	2
B. Indonesia	103	8	4
B. Inggris	103	16	2
Olahraga	103	21	2
Geografi	103	49	2
B. Jerman	103	52	2
Sosiologi	103	46	2
TIK	103	60	2
Bimbingan dan konseling	104	56	1
Matematika	104	27	4
Matematika	104	28	1
Kimia	104	38	3
Agama	104	2	3
Ekonomi	104	41	4
Ekonomi	104	41	1
Seni rupa	104	53	2
Biologi	104	33	3
Kewarganegaraan	104	5	2
Fisika	104	30	2
Sejarah	104	15	2
B. Indonesia	104	8	4
B. Inggris	104	16	2
Olahraga	104	21	2
Geografi	104	49	2
B. Jerman	104	52	2
Sosiologi	104	46	2
TIK	104	60	2

1. Pembentukan Individu Awal

Untuk membentuk individu awal, digunakan data dari pembagian guru. Berdasarkan data pembagian guru untuk kode_guru dengan jumlah_jam yang lebih dari 2 jam jumlah_jamnya akan dibagi-bagi menjadi 2 jam dan untuk kode_guru yang memiliki jumlah_jam hanya 1 jam akan digabung dengan kode_guru lain yang memiliki jumlah_jam 1 jam untuk membentuk jumlah_jam 2 jam. Kode_guru yang jumlah_jamnya sudah terbagi ke dalam 2 jam kan dijadikan satu dan menghasilkan individu awal. Berikut adalah tabel individu awal yang terbentuk:

Tabel 4.3 Pembentukan Individu Awal

Id_guru	Id_kelas	Id_guru	Id_kelas	Id_guru	Id_kelas	Id_guru	Id_kelas
25	101	25	102	26	103	27	104
25	101	25	102	26	103	27	104
38	101	38	102	38	103	38	104
2	101	2	102	2	103	2	104
41	101	41	102	41	103	41	104
41	101	41	102	41	103	41	104
53	101	53	102	53	103	53	104
33	101	33	102	33	103	33	104
5	101	5	102	5	103	5	104
30	101	30	102	30	103	30	104
15	101	15	102	15	103	15	104
8	101	8	102	8	103	8	104
8	101	8	102	8	103	8	104
16	101	16	102	16	103	16	104
16	101	16	102	16	103	16	104
21	101	21	102	21	103	21	104
49	101	49	102	49	103	49	104
52	101	52	102	52	103	52	104
46	101	46	102	46	103	46	104
60	101	60	102	60	103	60	104
701	101	704	102	707	103	710	104
702	101	705	102	708	103	711	104
703	101	706	102	709	103	712	104

2. Pembentukan Populasi Awal

Jumlah populasi yang digunakan adalah 10 individu. Untuk mendapatkan individu baru sejumlah 10 individu maka populasi awal yang terbentuk di acak 10 kali. Berikut salah satu hasil dari 10 individu yang terbentuk:

Tabel 4.4 Contoh Pembentukan Individu

Id_ individu	Isi_slot	Id_kelas	Id_ individu	Isi_slot	Id_kelas
1	53	101	1	53	103
1	41	101	1	708	103
1	701	101	1	5	103
1	8	101	1	33	103
1	38	101	1	16	103
1	49	101	1	21	103
1	41	101	1	60	103
1	702	101	1	26	103
1	25	101	1	2	103
1	2	101	1	49	103
1	8	101	1	38	103
1	25	101	1	15	103
1	16	101	1	16	103
1	52	101	1	26	103
1	60	101	1	41	103
1	33	101	1	46	103
1	16	101	1	8	103
1	703	101	1	41	103
1	46	101	1	52	103
1	30	101	1	8	103
1	5	101	1	707	103
1	15	101	1	709	103
1	21	101	1	30	103
1	41	102	1	41	104
1	15	102	1	710	104
1	49	102	1	16	104
1	8	102	1	46	104
1	8	102	1	41	104
1	705	102	1	8	104
1	5	102	1	711	104
1	706	102	1	27	104
1	25	102	1	60	104
1	16	102	1	38	104
1	16	102	1	27	104
1	53	102	1	53	104
1	704	102	1	52	104
1	33	102	1	5	104
1	46	102	1	8	104
1	41	102	1	15	104
1	60	102	1	712	104
1	25	102	1	21	104
1	2	102	1	49	104
1	30	102	1	30	104
1	21	102	1	16	104
1	38	102	1	33	104
1	52	102	1	2	104

3. Perhitungan Nilai Fitness

Individu/kromosom dievaluasi berdasarkan fungsi fitness untuk mendapatkan nilai fitness-nya. Fungsi fitness pada aplikasi ini di tujukan untuk meminimalkan pelanggaran-pelanggaran yang telah ditentukan. Berikut adalah fungsi fitness yang akan digunakan :

$$F = \frac{1}{(\ h1+h2+h3+h4+h5\ +\ a)} = \frac{1}{h+a} \qquad (4\text{-}2)$$

dengan:

h1 = jumlah jadwal yang mengalami bentrok dikalikan nilai prioritasnya

h2 = Jumlah mata pelajaran olahraga diatas jam keempat dikalikan nilai prioritasnya

h3 = Jumlah mata pelajaran matematika diatas jam keempat dikalikan nilai prioritasnya

h4 = Jumlah mata pelajaran fisika diatas jam keempat dikalikan nilai prioritasnya

h5 = Jumlah jam guru mengajar tidak boleh lebih dari 6 jam dalam satu hari

h = jumlah batasan yang dilanggar dikalikan nilai prioritasnya dan

a = bilangan yang bernilai sangat kecil untuk menghindari kemungkinan terjadinya pembagian dengan nilai 0.0001.

Berikut adalah tabel nilai prioritas untuk masing-masing pelanggaran :

Tabel 4.5 Nilai Prioritas Untuk Masing-Masing Pelanggaran

Pelanggaran	Nilai_prioritas
h1	1000
h2	700
h3	400
h4	400
h5	200

Kemudian dihitung jumlah masing-masing pelanggaran untuk masing-masing individu.

Berikut adalah tabel hasil perhitungan pelanggaran untuk masing-masing individu:

Tabel 4.6 Hasil Perhitungan Pelanggaran untuk Masing-Masing Individu

Nama	Id_individu	Nilai_fitness	Nama	Id_individu	Nilai_fitness
h1	1	11	h1	6	18
h2	1	1	h2	6	2
h3	1	5	h3	6	2
h4	1	1	h4	6	3
h5	1	2	h5	6	1
h1	2	9	h1	7	8
h2	2	2	h2	7	0
h3	2	6	h3	7	6
h4	2	0	h4	7	1
h5	2	3	h5	7	1
h1	3	12	h1	8	26
h2	3	1	h2	8	2
h3	3	7	h3	8	5
h4	3	3	h4	8	0
h5	3	3	h5	8	3
h1	4	10	h1	9	14
h2	4	2	h2	9	3
h3	4	7	h3	9	3
h4	4	3	h4	9	3
h5	4	3	h5	9	0
h1	5	11	h1	10	9
h2	5	2	h2	10	3
h3	5	6	h3	10	3
h4	5	2	h4	10	3
h5	5	3	h5	10	1

Dari hasil perhitungan jumlah pelanggaran dapat dihitung nilai fitnessnya misalnya untuk indvidu satu :

h =(h1x1000)+(h2x700)+(h3x400)+(h4x400)+(h5x200)

=(11x1000)+(1x700)+(5x400)+(1x400)+(2x200)

=14500

Fitness yang diperoleh

$$F = \frac{1}{h+a}$$

$$= \frac{\quad}{\quad}$$

= 6,896552 x10

Diperoleh nilai fitness untuk masing-masing individu sebagai berikut :

Tabel 4.7 Hasil Penghitungan Nilai Fitness

Id_fitness	Id_individu	Fitness
1	1	6.896552 x10
2	2	7.462686 x10
3	3	5.780347 x10
4	4	6.249999 x10
5	5	6.172839 x10
6	6	4.629630 x10
7	7	9.090909 x10
8	8	8.620690 x10
9	9	5.405405 x10
10	10	7.299270 x10

Masukkan individu beserta gen untuk fitness terbaik ke dalam tabel fitness terbaik

Tabel 4.8 Fitness Terbaik

Generasi	Id_individu	Fitness
0	7	8.620689 x10
1	7	9.090909 x10

4. Seleksi Orang Tua

Dari sekumpulan individu yang ada, akan dipilih individu-individu sebagai orangtua yang akan disilangkan untuk proses mendapatkan generasi selanjutnya. Proses seleksi orang tua dilakukan dengan menggunakan algoritma seleksi *roulette-wheel*. Total fitness = 6,768033 x10 sehingga fitness relatif (pk) tiap-tiap kromosom dapat dicari sebagai berikut:

p1=F1/Total fitness= 6.896552 x10 /6,768033 x10 = 0.1018989

p2=F2/Total fitness= 7.462686 x10 /6,768033 x10 = 0.1102637

p3=F3/Total fitness= 5.780347 x10 /6,768033 x10 = 0.0854066

Dan seterusnya. Sedangkan fitness kumulatif dapat dicari sebagai berikut:

q1 = p1=0.1018989

q2 = q1 + p2 =0.2121626

q3 = q2 + p3 =0.75692121

dan seterusnya. Berikut adalah hasil perhitungan fitness relatif dan fitness kumulatif untuk masing-masing individu:

Tabel 4.9 hasil Perhitungan Fitness Relatif dan Fitness Kumulatif

Id_populasi	Fitness_relatif	Fitness_kumulatif
1	0.102007428	0.102007428
2	0.110381172	0.212388601
3	0.085497556	0.297886157
4	0.092444232	0.390330389
5	0.091302945	0.481633334
6	0.068477209	0.550110543
7	0.134464337	0.68457488
8	0.127509285	0.812084165
9	0.079951768	0.892035934
10	0.107964066	1

Membangkitkan bilangan acak untuk masing-masing individu dan fitness kumulatif. Berikut adalah hasil bilangan acak yang diperoleh:

Tabel 4.10 Hasil Pembangkitan Bilangan Acak

Id_populasi	Acak	Fitness_kumulatif
1	0.591039456	0.102007428
2	0.900293779	0.212388601
3	0.033025508	0.297886157
4	0.603416523	0.390330389
5	0.984859815	0.481633334
6	0.181631712	0.550110543
7	0.541400695	0.68457488
8	0.439197933	0.812084165
9	0.912921789	0.892035934
10	0.990372434	1

Terdapat 2 syarat agar yaitu:

1. ≤ & >
2. >

Jika individu memenuhi syarat pertama maka individu k+1 tersebut akan dimasukkan ke dalam kromosom baru dan jika memenuhi syarat kedua maka individu k dimasukkan ke dalam kromosom baru. Berikut adalah individu hasil seleksi induk:

Tabel 4.11 Kromosom Baru Hasil Seleksi Induk

Id	Nilai_fitness	Individu
1	9.090909 x10	7
2	7.29927 x10	10
3	6.896552 x10	1
4	9.090909 x10	7
5	7.29927 x10	10
6	7.462687 x10	2
7	4.62963 x10	6
8	6.172839 x10	5
9	7.29927 x10	10
10	7.29927 x10	10

5. Rekombinasi

Menentukan nilai Pc = 0.7 Membangkitkan bilangan acak untuk masing-masing individu kromosom baru.

Tabel 4.12 Pembangkitan Bilangan Acak untuk Rekombinasi

Individu	Pc	Acak	Id_kromosom
7	0.7	0.169455	1
10	0.7	0.676687	2
1	0.7	0.12173	3
7	0.7	0.54214	4
10	0.7	0.113003	5
2	0.7	0.909166	6
6	0.7	0.464447	7
5	0.7	0.379305	8
10	0.7	0.718994	9
10	0.7	0.000345	10

Jika < maka individu k dijadikan calon induk untuk proses rekombinasi. Dari tabel kromosom baru terlihat bahwa kromosom 1,2,3,4,5,7,8,10 memenuhi syarat. Maka individu tersebut masuk ke orang tua. Berikut daftar orang tua yang terbentuk :

Tabel 4.13 Kumpulan Induk yang Terpilih untuk Proses Rekombinasi

Individu	Id_kromosom
7	1
10	2
1	3
7	4
10	5
6	7
5	8
10	10

Sehingga terdapat 4 pasang orang tua yang nantinya akan direkombinasikan untuk menghasilkan anak yang baru yang nantinya anak-anak tersebut menggantikan induknya pada kromosom baru.

Tabel 4.14 Tabel Pasangan Induk yang Mengalami Rekombinasi

Kode_pasangan	Individu	Id_kromosom
1	7	1
1	10	2
2	1	3
2	7	4
3	10	5
3	6	7
4	5	8
4	10	10

6. Mutasi

Metode mutasi yang digunakan adalah mutasi pertukaran (*swap mutation)*. Mutasi ini berjalan dengan cara menukarkan posisi dua gen yang telah dipilih secara acak. Proses acak bergantung pada nilai Ps (Probablitias seleksi mutasi), dengan nilai Ps= 0.1 dan bilangan acak dibangkitkan untuk masing-masing individu kromosom baru.

Tabel 4.15 Pembangkitan Bilangan Acak untuk menentukan Individu yang Termutasi

Id_kromosom	Individu	Ps	Acak
3	1	0.1	0.969658757
6	2	0.1	0.299195477
8	5	0.1	0.76353723
7	6	0.1	0.429781251
1	7	0.1	0.03528946
4	7	0.1	0.815180056
2	10	0.1	0.632121354
5	10	0.1	0.415343935
9	10	0.1	0.416324353
10	10	0.1	0.007697055

Jika < maka kromosom k dimasukkan ke dalam individu yang termutasi. Dari tabel di atas kromosom 3 (individu 1) memenuhi syarat. Sehingga dimasukkan ke dalam individu termutasi.

Berikut daftar individu termutasi :

Tabel 4.16 Individu yang Terpilih untuk Mutasi

Individu	Id_kromosom
7	1
10	10

Dilihat dari tabel tersebut terdapat 7 individu yang memungkinkan untuk dimutasi. Menentukan jumlah bit yang akan dimutasi dengan langkah-langkah sebagai berikut:

a. Jumlah slot 1 individu (L) = jumlah kelas x jumlah slot 1 individu
= 4 x 23
= 92

b. Total bit = L x jumalah individu (popsize)
= 92 x 10
= 920

c. Jumlah bit = Pm x total bit (dengan Pm = 0.01)
= 0.01 x 920
= 9.2
= 9

d. Pasang bit = 9-1
= 8/2
= 4 pasang

Pasang bit yang dibutuhkan untuk dimutasi sebanyak 4 pasang (8 bit) hal ini berarti terdapat 4 individu yang di mutasi. Diketahui bahwa terdapat 2 individu yang memungkinkan untuk dimutasi maka masing-masing individu memiliki 2 pasang bit yang ditukarkan. Setelah di mutasi didapatkan kromosom dan individu baru.

7. Elitisme

Dari individu yang terbentuk setelah proses mutasi, dihitung kembali nilai fitnessnya, diambil nilai fitness yang terbaik untuk disimpan dan jika nilai fitness terburuk < nilai fitness terbaik generasi sebelumnya, nilai fitness terburuk digantikan dengan nilai fitness terbaik generasi sebelumnya. Kemudian mengulang kembali proses dari tahap 3 sampai 7 sampai jumlah generasi memenuhi syarat.

8. Pembentukan Individu Final

   Berikut adalah daftar nilai fitness terbaik selama dua generasi

   Tabel 4.17 Nilai Fitness Terbaik untuk Tiap Generasi

Generasi	Id_individu	Fitness
0	7	8.6206896 x10
1	7	9.090909 x10
2	5	9.090909 x10

   Dari tabel tersebut diambil individu yang terbaik yaitu individu dari generasi yang memiliki nilai fitness paling besar yaitu individu 5 dari generasi 2 untuk diambil gennya.

9. Pembentukan Jadwal Mata Pelajaran

   Pembentukan jadwal mapel terbentuk dari individu terbaik yaitu individu 5 generasi 2.

Kelas	101							
Hari/Jam	1	2	3	4	5	6	7	8
Senin	9	9	16	16	41	41	50	50
Selasa	32	32	48	48	38	38	21	21
Rabu	24	24	24	24	6	6	61	61
Kamis	3	3	38	3	9	9	15	15
Jumat	16	16	44	37	41	41		
Sabtu	60	25	37	37	53	53	52	52

Kelas	103							
Hari/Jam	1	2	3	4	5	6	7	8
Senin	19	19	38	38	11	11	24	24
Selasa	43	43	52	52	41	33	53	53
Rabu	21	21	50	50	61	61	24	24
Kamis	59	25	5	5	19	19	3	3
Jumat	31	31	43	43	48	48		
Sabtu	11	11	38	3	33	33	14	14

Kelas	102							
Hari/Jam	1	2	3	4	5	6	7	8
Senin	11	11	16	16	48	48	43	43
Selasa	5	5	43	43	38	3	15	15
Rabu	11	11	24	24	3	3	31	31
Kamis	52	52	50	50	41	33	33	33
Jumat	48	48	21	21	38	38		
Sabtu	60	25	24	24	54	54	16	16

Kelas	104							
Hari/Jam	1	2	3	4	5	6	7	8
Senin	29	29	53	53	43	43	14	14
Selasa	48	48	19	19	3	3	11	11
Rabu	43	43	1	25	11	11	31	31
Kamis	52	52	52	52	38	38	19	19
Jumat	21	21	29	29	7	7		
Sabtu	61	61	41	34	38	3	34	34

Gambar 4.12 Jadwal Mata Pelajaran yang Terbentuk dari Individu Terbaik

4.2.2 Implementasi Basis Data MySQL

Implementasi basis data MySQL dibuat berdasarkan *Database Relational*. Implementasi perancangan basis data aplikasi penjadawalan menggunakan *Data Definition Language* (DDL). DDL yang digunakan dalam membuat basis data aplikasi penjadwalan adalah sebagai berikut :

```
CREATE DATABASE  penjadwalan;
```

DDL yang digunakan untuk membentuk tabel kelas adalah sebagai berikut:

```
CREATE TABLE `kelas` (
  `id_kelas` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `kode_kelas` int(5) unsigned NOT NULL,
  `kelas` varchar(25) NOT NULL,
  `id_tingkat` int(5) NOT NULL,
  PRIMARY KEY (`id_kelas`)
)
```

DDL yang digunakan untuk membentuk tabel guru adalah sebagai berikut:

```
CREATE TABLE `guru` (
  `id_guru` int(11) NOT NULL AUTO_INCREMENT,
  `nama_guru` varchar(100) NOT NULL,
 `bidang_tugas` varchar(10) NOT NULL,
  PRIMARY KEY (`id_guru`)
);
```

DDL yang digunakan untuk membentuk tabel pembagian_guru adalah sebagai berikut:

```
CREATE TABLE `pembagian_guru` (
  `id_pembagian` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `mata_pelajaran` varchar(50) NOT NULL,
  `kode_kelas` int(10) NOT NULL,
  `kode_guru` varchar(10) NOT NULL,
  `jumlah_jam` int(10) NOT NULL,
PRIMARY KEY (`id_pembagian`)
)
```

DDL yang digunakan untuk membentuk tabel populasi_awal adalah sebagai berikut:

```
CREATE TABLE `populasi_awal` (
  `id` int(10) NOT NULL AUTO_INCREMENT,
  `id_guru` int(10) DEFAULT NULL,
  `id_kelas` int(10) DEFAULT NULL,
  PRIMARY KEY (`id`)
)
```

DDL yang digunakan untuk membentuk tabel individu adalah sebagai berikut:

```
CREATE TABLE `individu` (
  `id` int(100) NOT NULL AUTO_INCREMENT,
  `id_populasi` int(100) NOT NULL,
  `isi_slot` int(100) NOT NULL,
  `id_kelas` int(100) DEFAULT NULL,
  PRIMARY KEY (`id`)
)
```

DDL yang digunakan untuk membentuk tabel nilai_fitness adalah sebagai berikut:

```
CREATE TABLE `nilai_fitness` (
  `id_fitness` int(10) NOT NULL AUTO_INCREMENT,
  `nama` varchar(100) NOT NULL,
  `id_populasi` int(100) NOT NULL,
  `nilai_fitness` int(50) NOT NULL,
  PRIMARY KEY (`id_fitness`)
)
```

DDL yang digunakan untuk membentuk tabel fitness adalah sebagai berikut:

```
CREATE TABLE `fitness` (
  `id_fitness` int(11) NOT NULL AUTO_INCREMENT,
  `id_populasi` int(11) NOT NULL,
  `fitness` double NOT NULL,
  PRIMARY KEY (`id_fitness`)
)
```

DDL yang digunakan untuk membentuk tabel fitness_relatif adalah sebagai berikut:

```
CREATE TABLE `fitness_relatif` (
  `id_relatif` int(200) NOT NULL AUTO_INCREMENT,
  `id_populasi` int(200) NOT NULL,
  `fitness_relatif` double NOT NULL,
  PRIMARY KEY (`id_relatif`)
)
```

DDL yang digunakan untuk membentuk tabel kromosom_baru adalah sebagai berikut:

```
CREATE TABLE `kromosom_baru` (
  `id` int(100) NOT NULL AUTO_INCREMENT,
  `nilai_fitness` double NOT NULL,
  `individu` int(100) NOT NULL,
  PRIMARY KEY (`id`)
)
```

DDL yang digunakan untuk membentuk tabel kromosom adalah sebagai berikut:

```
CREATE TABLE `kromosom` (
  `id` int(100) NOT NULL AUTO_INCREMENT,
  `id_populasi` int(100) NOT NULL,
  `isi_slot` int(100) NOT NULL,
  `id_kelas` int(100) DEFAULT NULL,
  `id_kromosom` int(100) NOT NULL,
  PRIMARY KEY (`id`)
)
```

DDL yang digunakan untuk membentuk tabel orang_tua adalah sebagai berikut:

```
CREATE TABLE `orang_tua` (
  `id` int(100) NOT NULL AUTO_INCREMENT,
  `individu` int(100) NOT NULL,
  `id_kromosom` int(100) NOT NULL,
  PRIMARY KEY (`id`)
)
```

DDL yang digunakan untuk membentuk tabel pasangan_orangtua adalah sebagai berikut:

```
CREATE TABLE `pasangan_orangtua` (
  `id` int(100) NOT NULL AUTO_INCREMENT,
  `kode_pasangan` int(100) NOT NULL,
  `individu` int(100) NOT NULL,
  `id_kromosom` int(100) NOT NULL,
  PRIMARY KEY (`id`)
)
```

DDL yang digunakan untuk membentuk tabel anak adalah sebagai berikut:

```
CREATE TABLE `anak` (
  `id` int(100) NOT NULL AUTO_INCREMENT,
  `id_kromosom` int(100) NOT NULL,
  `individu` int(100) NOT NULL,
  `no_slot` int(100) NOT NULL,
  `isi_slot` int(100) NOT NULL,
  PRIMARY KEY (`id`)
)
```

DDL yang digunakan untuk membentuk tabel seleksi_mutasi adalah sebagai berikut:

```
CREATE TABLE `seleksi_mutasi` (
  `id` int(100) NOT NULL AUTO_INCREMENT,
  `id_kromosom` int(100) NOT NULL,
  `individu` int(100) NOT NULL,
  `ps` double NOT NULL,
  `acak` double NOT NULL,
  PRIMARY KEY (`id`)
)
```

DDL yang digunakan untuk membentuk tabel fitness_terbaik adalah sebagai berikut:

```
CREATE TABLE `fitness_terbaik` (
  `id` int(100) NOT NULL AUTO_INCREMENT,
  `generasi` int(100) NOT NULL,
  `id_populasi` int(100) NOT NULL,
  `fitness` double NOT NULL,
  PRIMARY KEY (`id`)
)
```

DDL yang digunakan untuk membentuk tabel individu_terbaik adalah sebagai berikut:

```
CREATE TABLE `individu_terbaik` (
  `id` int(100) NOT NULL AUTO_INCREMENT,
  `generasi` int(100) NOT NULL,
  `individu` int(100) NOT NULL,
  `isi_slot` int(100) NOT NULL,
  `id_kelas` int(100) NOT NULL,
  PRIMARY KEY (`id`)
)
```

DDL yang digunakan untuk membentuk tabel individu_final adalah sebagai berikut:

```
CREATE TABLE `individu_final` (
  `id` int(100) NOT NULL AUTO_INCREMENT,
  `slot` int(100) NOT NULL,
  `isi_slot` int(100) NOT NULL,
  `kode_kelas` int(100) NOT NULL,
  PRIMARY KEY (`id`)
)
```

DDL yang digunakan untuk membentuk tabel jadwal_mapel adalah sebagai berikut:

```
CREATE TABLE `jadwal_mapel` (
  `id` int(100) NOT NULL AUTO_INCREMENT,
  `kelas` int(100) NOT NULL,
  `hari` varchar(100) NOT NULL,
  `jam` int(100) NOT NULL,
  `id_guru` int(100) NOT NULL,
  PRIMARY KEY (`id`)
)
```

4.2.3 Implementasi Pemrograman Sistem

Pemrograman sistem penjadwalan merupakan pemrograman untuk sistem algoritma genetika secara keseluruhan

```
int jumlah_generasi = 10;
    for (int v=1; v<=jumlah_generasi; v++){
    if (v==1){
        getjam();
        getrand();
        fitness();
        insert_fitness();
    }
get_seleksi_ortu();
rekombinasi();
mutasi();
elitisme(v);
}
get_individu_final();
get_jadwal_mapel();
```

Gambar 4.13 Kode program untuk sistem penjadwalan

Dari kode program di atas dapat dilihat bahwa terdapat beberapa fungsi yang membentuk sistem penjadwalan dengan algoritma genetika. Fungsi-fungsi tersebut antara lain:

1. Pembentukan individu awal

   Pembentukan individu awal dapat dilihat pada fungsi `getjam();`. Fungsi ini terdiri dari 2 proses, yang pertama proses memecah mata pelajaran yang nilainya lebih dari 2 jam menjadi masing-masing 2 jam dan menggabungkan dua mata pelajaran yang nilainya 1 jam menjadi 2 jam.

```
ResultSet hasilkelas = aplikasi.getkelas();
    while (hasilkelas.next()){
        int kode_kelas = hasilkelas.getInt("kode_kelas");
        ResultSet hasilguru = aplikasi.getguru(kode_kelas);
        while (hasilguru.next()){
            int kode_guru = hasilguru.getInt("kode_guru");
            x = hasilguru.getInt("jumlah_jam");
            if (x%2==0){
            y=x/2;
            if (y==1){
                aplikasi.insert2jam(kode_guru, kode_kelas);
            }
             else {
            jumlah = 1;
            while (jumlah <=y){
                aplikasi.insert2jam(kode_guru, kode_kelas);
                jumlah++;
                }
            }
            }
```

```
else if (x%2!=0){
   z=x%2;
   w=x-z;
   y=w/2;
   if (y==0){
   aplikasi.insert1jam(kode_guru, kode_kelas);
   } else {
   jumlah = 1;
   while (jumlah <=y){
      aplikasi.insert2jam(kode_guru, kode_kelas);
      jumlah++;
            }
            aplikasi.insert1jam(kode_guru, kode_kelas);
                  }
              }
           }
    }
```

Gambar 4.14 Kode program untuk memecah mata pelajaran yang nilainya lebih dari 2 jam menjadi masing-masing 2 jam

```
ResultSet hasil1kelas = aplikasi.get1kelas();
  kode_gabung = 700;
  while (hasil1kelas.next()){
   int id_kelas = hasil1kelas.getInt("id_kelas");
   int jum = aplikasi.count1jam(id_kelas);
   batas = 2;
   int batasloop = jum/batas;
   for (int l=1; l<=batasloop; l++ ){
       kode_gabung = kode_gabung +1;
       int posisi = (l-1)*batas;
       ResultSet gabung = aplikasi.get2(id_kelas, posisi, batas);
while (gabung.next()){
    int guru = gabung.getInt("id_guru");
    int kelas = gabung.getInt("id_kelas");
    aplikasi.insertgabung(guru, kelas, kode_gabung);
         }
      }
     }
```

Gambar 4.15 Kode program penggabungan 2 mata pelajaran yang nilainya 1 jam menjadi 2 jam

Untuk proses kedua dari pembentukan individu awal adalah mengambil data mata pelajaran yang sudah terbagi menjadi 2 jam untuk dimasukkan kedalam tabel populasi_awal sesuai dengan urutan id_kelas-nya.

```
ResultSet hasilkelas2 = aplikasi.getkelas();
    while (hasilkelas2.next()){
      kls = hasilkelas2.getInt("kode_kelas");
      ResultSet getpop1 = aplikasi.get2jam(kls);
      while (getpop1.next()){
          int pop_guru1 = getpop1.getInt("id_guru");
          int pop_kelas1 = getpop1.getInt("id_kelas");
          aplikasi.insertpop(pop_guru1, pop_kelas1);
          }
      ResultSet getpop2 = aplikasi.getgabung(kls);
      while (getpop2.next()){
          int pop_guru2 = getpop2.getInt("kode_gabung");
          int pop_kelas2 = getpop2.getInt("id_kelas");
          aplikasi.insertpop(pop_guru2, pop_kelas2);
                    }
               }
       }
```

Gambar 4.16 Kode program prmbentukan individu awal

2. Pembentukan kromosom baru

Pembentukan kromosom baru terdapat pada fungsi `getrand();` Proses dari fungsi ini adalah untuk membentuk individu sebanyak jumlah individu yang telah ditetapkan. Untuk membentuk 1 individu dilakukan dengan cara mengambil data mata pelajaran per kelas secara acak dari tabel populasi_awal ke dalam tabel individu.

```
public void getrand(){
try{
int jumindividu1 = aplikasi.jumindividu1();
                if (jumindividu1 > 0){
                    aplikasi.deleteindividu1();
                }
                for (int j=1; j<=jumlah_iterasi; j++){
                    ResultSet popkelas = aplikasi.getkelas();
                    while (popkelas.next()){
                        int kelas_pop = popkelas.getInt("kode_kelas");
                        ResultSet getpopulasi =
aplikasi.getpopulasi(kelas_pop);
                        while (getpopulasi.next()){
                            int id_guru = getpopulasi.getInt("id_guru");
                            int id_kelas = getpopulasi.getInt("id_kelas");
                            aplikasi.insertrand(j, id_guru, id_kelas);
                        }
                    }
                }
 }
```

Gambar 4.17 Kode program getrand(); untuk pembentukan kromosom baru

3. Perhitungan fitness

Perhitungan fitness terdapat pada fungsi `fitness();`. Pada fungsi `fitness();` terdapat 6 proses antara lain proses perhitungan bentrok, perhitungan jumlah mata pelajaran olahraga diatas jam ke-4, perhitungan jumlah mata pelajaran fisika diatas jam ke-4, perhitungan jumlah mata pelajaran matematika diatas jam ke-4, perhitungan jumlah jam guru yang mengajar lebih dari 6 jam sehari dan proses menghitung nilai fitness.

Untuk menghitung bentrok dilakukan dengan cara membandingkan 1 titik dengan titik yang lain yang nilai hari dan jamnya sama, misal titik ke-0 yang mewakili hari senin jam pertama dengan titik ke-23 yang juga mewakili hari senin jam pertama. Jika nilai di kedua titik tersebut sama maka dihitung sebagai pelanggaran.

```
//perhitungan jumlah pelanggaran
              int[] populasi = new int[600];
              populasi = list();
              int k;
                      //awal bentrok
                      int j;
                      int h1=0;
                      int i=0;
                      int jum_kls= aplikasi.countkelas(); // jumlah kls=23
                      int jum = jum_kls-1; //loop pencarian bentrok antar
kelas a dengan kelas yang lain
                      int input = 1;
                      int loop = jum_kls-1;
                      while (input<=loop){
                      for ( k=0+i; k<=22+i; k++){ // kelas yang dibandingkan
                      for ( j=1; j<=jum; j++){ // kelas pembanding
                          if((populasi[k]<700)&&(populasi[k+(j*23)]<700)){

                     if(populasi[k]==populasi[k+(j*23)]){
                              h1=h1+1;
                              }
                          }
else if ((populasi[k]<700)&&(populasi[k+(j*23)]>700)){
                              ResultSet gabungan1 =
aplikasi.getkodegabung(populasi[k+(j*23)]);
                              while(gabungan1.next()){
                                  int idj = gabungan1.getInt("id_guru");
                                  if(populasi[k]==idj){
                                   h1=h1+1;
                                   }
                              }
                          }
else if ((populasi[k]>700)&&(populasi[k+(j*23)]<700)){
        ResultSet gabungan2 = aplikasi.getkodegabung(populasi[k]);
        while(gabungan2.next()){
        int idk = gabungan2.getInt("id_guru");
        if(populasi[k+(j*23)]==idk){
        h1=h1+1;
            }
         }
      }
```

```
else if ((populasi[k]>700)&&(populasi[k+(j*23)]>700)){
        ResultSet gabungan3 = aplikasi.getkodegabung(populasi[k]);
        while(gabungan3.next()){
           int idk = gabungan3.getInt("id_guru");
           ResultSet gabungan4 = aplikasi.getkodegabung(populasi[k+(j*23)])
        while(gabungan4.next()){
        int idj = gabungan4.getInt("id_guru");
               if(idk==idj){
               h1=h1+1;
               }
            }
          }
        }

   }
   }
       i=i+23;
   jum--;
   input++;
    }
    aplikasi.insertfitnes("h1",pop, h1);
    //akhir bentrok
```

Gambar 4.18 Kode program untuk menghitung jumlah bentrok

Untuk menghitung jumlah mata pelajaran olahraga yang terletak diatas jam ke-4 dilakukan dengan cara mengecek isi slot yang terletak diatas jam ke-4. Jika isi slot tersebut terdapat kode guru yang mengajar mata pelajaran olahraga maka dihitung sebagai pelanggaran. Cara ini juga digunakan untuk menghitung jumlah mata pelajaran fisika dan matematika yang terletak diatas jam ke-4.

```
//awal penjas
                      int h2=0;
                      int juml_kls=aplikasi.countkelas();
                      int juml = juml_kls-1;
                      for (int l=0; l<=juml; l++){
                        //jam ke-3
                        ResultSet getjam3 = aplikasi.getjam3();
                        while(getjam3.next()){
                            int slot = getjam3.getInt("no_slot");
                            k = slot+(l*23);
                            ResultSet penjas = aplikasi.getpenjaskes();
                            while(penjas.next()){
                                int penjaskes = penjas.getInt("id_guru");
                                if (populasi[k]>700){
                                    ResultSet gabungan4 =
aplikasi.getkodegabung(populasi[k]);
                                    while(gabungan4.next()){
                                            int idk =
gabungan4.getInt("id_guru");
                                            if (idk == penjaskes){
                                            h2=h2+1;
                                            }
                                    }
                                }
```

```
else if(populasi[k]<700){
                                    if (populasi[k] == penjaskes){
                                    h2=h2+1;
                                    }
                                }
                            }
                        }
                        //jam ke -4
                        ResultSet getjam4 = aplikasi.getjam4();
                        while(getjam4.next()){
                            int slot = getjam4.getInt("no_slot");
                            k = slot+(1*23);
ResultSet penjas2 = aplikasi.getpenjaskes();
                            while(penjas2.next()){
                                int penjaskes2 = penjas2.getInt("id_guru");
                           if (populasi[k]>700){
                                    ResultSet gabungan4 =
aplikasi.getkodegabung(populasi[k]);
                                    while(gabungan4.next()){
                                            int idk =
gabungan4.getInt("id_guru");
                                            if (idk == penjaskes2){
                                            h2=h2+1;
             }
          }
        }

else if(populasi[k]<700){

    if (populasi[k] == penjaskes2){

     h2=h2+1;

    } } } }

   //akhir jam ke-4

}

aplikasi.insertfitnes("h2",pop, h2);

//akhir penjas
```

Gambar 4.19 Kode program untuk menghitung jumlah mata pelajaran olahraga yang terletak diatas jam keempat

Sedangkan untuk menghitung jumlah guru yang mengajar lebih dari 6 jam dalam sehari dilakukan dengan cara memilih kode guru kemudian mengecek jumlah mengajar dimulai pada hari senin. Jika pada hari senin terdapat jumlah jam mengajar lebih dari 6 jam maka dihitung sebagai pelanggaran, begitu seterusnya hingga hari sabtu.

```
//fitness >6jam mengajar
  int h5=0;
  int jum_kelas=aplikasi.countkelas();
  int jlh = jum_kelas-1;
  ResultSet allguru = aplikasi.getallguru();
  while (allguru.next()){
  int kode_guru = allguru.getInt("id_guru");
  ResultSet getallhari = aplikasi.gethari();
  while (getallhari.next()){
     String allhari = getallhari.getString("hari");
     ResultSet getallslot = aplikasi.getnoslot(allhari);
     int jum_sama =0;
     while (getallslot.next()){
     for (int p=0; p<=jlh; p++){
     int allslot = getallslot.getInt("no_slot");
     k = allslot+(p*23);
     if (populasi[k]>700){
     ResultSet gabungan7 = aplikasi.getkodegabung(populasi[k]);
     while(gabungan7.next()){
       int idk = gabungan7.getInt("id_guru");
       if (kode_guru==idk){
       jum_sama = jum_sama +1;
           }
         }
        } else if(populasi[k]<700){
          if (kode_guru == populasi[k]){
          jum_sama = jum_sama +1;
                                          }
                                       }
                                     }
                                   }
              if (jum_sama >3){
    h5 = h5+1;
      }
    }
  }
aplikasi.insertfitnes("h5",pop, h5);
 //akhir >6jam mengajar
```

Gambar 4.20 Kode program untuk menghitung jumlah mengajar guru yang lebih dari 6 jam sehari

Untuk proses menghitung nilai fitness dilakukan dengan cara menghitung jumlah pelanggaran masing-masing batasan yang telah dikalikan dengan nilai prioritasnya. Kemudian dimasukkan kedalam rumus f = $\frac{}{(\quad)}$

```
//batas perhitungan fitness
//perhitungan fitness
fitness = new int[100];
       int s = 0;
       int total = 0;
       int jum_fitness=0;
       double na = 0.0001;
       ResultSet batasan = aplikasi.getpelanggaran();
       while(batasan.next()){
           String pelanggaran = batasan.getString("pelanggaran");
           int nilai_prioritas = batasan.getInt("nilai_prioritas");
           ResultSet jumL = aplikasi.getjumpelanggaran(pelanggaran, pop);
           while (jumL.next()){
                int jum_pelanggaran = jumL.getInt("nilai_fitness");
                jum_fitness = jum_pelanggaran*nilai_prioritas;
                    } total += jum_fitness;
                 }
                double f = (1/(total+na));
                aplikasi.inserttotalfitnes(pop, f);
                }
  }
}
```

Gambar 4.21 Kode program dari proses menghitung nilai fitness

4. Pengambilan nilai fitness terbaik

Pengambilan nilai fitness terbaik terdapat pada fungsi `insert_fitness();`. Fungsi ini digunakan untuk memilih individu dengan nilai fitness terbaik dalam 1 generasi untuk dimasukkan kedalam tabel individu_terbaik.

```
public void insert_fitness(){
 try{
int n_fitness=0;
  ResultSet max_fitness = aplikasi.get_max_fitness();
  while (max_fitness.next()){
  int individu = max_fitness.getInt("id_populasi");
  double fitness2 = max_fitness.getDouble("fitness");
  aplikasi.insert_fitness_terbaik(n_fitness, individu, fitness2);
  ResultSet get_individu_terbaik = aplikasi.get_individu_terbaik(individu);
  while (get_individu_terbaik.next()){
  int isi_slot = get_individu_terbaik.getInt("isi_slot");
  int id_kelas = get_individu_terbaik.getInt("id_kelas");
  aplikasi.insert_individu_terbaik(n_fitness, individu, isi_slot, id_kelas);
                    }
                }
 }}
```

Gambar 4.22 Kode program untuk mengambil nilai fitness terbaik

5. Seleksi orang tua

Proses seleksi orang tua terdapat pada fungsi `get_seleksi_ortu ();` Langkah-langkah untuk melakukan seleksi orang tua pada fungsi ini antara lain:

a. Menghitung total fitness dengan cara menjumlahkan seluruh nilai fitness dalam 1 populasi.

```
//menghitung total fitness
              double total_fitness = 0;
              ResultSet get_fitness = aplikasi.getfitness();
              while (get_fitness.next()){
                    double nilai_fitness =
get_fitness.getDouble("fitness");
                    total_fitness += nilai_fitness;
              }
```

Gambar 4.23 Kode program untuk menghitung total fitness

b. Menghitung fitness relatif tiap-tiap individu dengan rumurs = _____.

```
//menghitung fitness relatif
ResultSet get_fitness2 = aplikasi.getfitness2();
              while (get_fitness2.next()){
                    int p = get_fitness2.getInt("id_populasi");
                    double nilai_fitness2 =
get_fitness2.getDouble("fitness");
                    double fitness_relative =
nilai_fitness2/total_fitness;
                    aplikasi.insertrelatif(p, fitness_relative );
              }
```

Gambar 4.24 Kode program untuk menghitung fitness relatif

c. Menghitung fitness kumulatif tiap-tiap individu dengan rumus:

- =
- = +

```
//menghitung fitness kumulatif
              ResultSet get_relatif = aplikasi.getrelatif();
              while (get_relatif.next()){
                    int po = get_relatif.getInt("id_populasi");
                    double pk = get_relatif.getDouble("fitness_relatif");
                    if (po==1){
                        double qk = pk;
                        aplikasi.insertkumulatif(po, pk, qk);
                    } else {
                        int po2 = po-1;
                        ResultSet get_kumulatif = aplikasi.getkumulatif(po2);
                        while (get_kumulatif.next()){
                            double qk2 =
get_kumulatif.getDouble("fitness_kumulatif");
                            double qk = pk + qk2;
                            aplikasi.insertkumulatif(po, pk, qk);
              }}}
```

Gambar 4.25 Kode program untuk menghitung fitness kumulatif

d. Membangkitkan bilangan acak sebanyak jumlah individu dalam 1 populasi.

```
//membangkitkan bilangan acak
ResultSet get_allkumulatif = aplikasi.getalltkumulatif();
        while (get_allkumulatif.next()){
        int individu = get_allkumulatif.getInt("id_populasi");
        double qk = get_allkumulatif.getDouble("fitness_kumulatif");
        Random random = new Random();
        double rand = random.nextDouble();
        aplikasi.insertacak(individu, rand, qk);
              }
```

Gambar 4.26 Kode program untuk membangkitkan bilangan acak

e. Memilih induk yang akan menjadi kandidat untuk direkombinasi dengan menggunakan aturan:

- Jika ≤ bil. Acak dan > bil. Acak, maka pilih individu ke (k+1) sebagai kandidat induk
- Jika > bli.acak, maka pilih individu ke *k* sebagai kandidat induk

```
int jml = jumlah_iterasi-1;
double[]acak = new double[100];
double[]komulatif = new double[100];
acak = listacak();
komulatif = listkomulatif();
for ( int c=0; c<=jml; c++){
  int individu = c+1;
  int g =0;
  int input=1;
  while (input<=jml){
    for (int q=0+g; q<=0+g; q++){
    int individu1 = q+1;
    int individu2 = q+2;
    if (( komulatif[q] <= acak[c])&&( komulatif[q+1] > acak[c])){
    ResultSet get_fitness3 = aplikasi.get_fitness(individu2);
    while (get_fitness3.next()){
    double nilai_fitness = get_fitness3.getDouble("fitness");
    aplikasi.insertkromosom(nilai_fitness, individu2);
    input=jml;
      }
    }
else if(( komulatif[q] > acak[c])) {
ResultSet get_fitness3 = aplikasi.get_fitness(individu1);
 while (get_fitness3.next()){
 double nilai_fitness = get_fitness3.getDouble("fitness");
 aplikasi.insertkromosom(nilai_fitness, individu1);
  input=jml;
        }
       }
   }
   g++;
   input++;
    }
 }
```

Gambar 4.27 Kode program untuk memilih kandidat induk

f. Memasukkan kandidat induk yang terpilih kedalam tabel kromosom_baru.

```
ResultSet get_kromosom = aplikasi.get_kromosom();
while (get_kromosom.next()){
int id_kromosom = get_kromosom.getInt("id");
int k_individu = get_kromosom.getInt("individu");
ResultSet get_individu_kromosom = aplikasi.get_individu_kromosom(k_individu);
while (get_individu_kromosom.next()){
int isi_slot = get_individu_kromosom.getInt("isi_slot");
int id_kelas = get_individu_kromosom.getInt("id_kelas");
aplikasi.insert_genkromosom(k_individu, isi_slot, id_kelas, id_kromosom);
                    }
 }}}
```

Gambar 4.28 Kode program untuk memasukkan kandidat induk yang terpilih kedalam tabel kromosom_baru.

6. Rekombinasi

Proses rekombinasi terdapat pada fungsi `rekombinasi();`. Langkah-langkah untuk melakukan rekombinasi pada fungsi ini antara lain:

a. Membangkitkan bilangan acak sebanyak jumlah individu dalam populasi

```
//mencari probabilitas crossover
double Pc = 0.5;
ResultSet get_kromosombaru = aplikasi.get_kromosombaru();
while (get_kromosombaru.next()){
    int id_kromosom = get_kromosombaru.getInt("id");
    int individu = get_kromosombaru.getInt("individu");
    Random randoms = new Random();
    double rands = randoms.nextDouble();
    aplikasi.insert_probabilitas(individu, Pc, rands, id_kromosom);
   }
```

Gambar 4.29 Kode Program untuk membangkitkan bilangan acak sebanyak jumlah individu dalam populasi

b. Menghitung jumlah pasangan orang tua. Langkah-langkahnya adalah sebagai berikut:

   i. Periksa bilangan acak tiap-tiap individu. Jika nilai bilangan acak lebih kecil dari Pc, maka individu tersebut terpilih sebagai orang tua.

```
ResultSet getProbabilitas = aplikasi.get_probabilitas();
while (getProbabilitas.next()){
   int individu = getProbabilitas.getInt("individu");
   double pc = getProbabilitas.getDouble("pc");
   double bilAcak = getProbabilitas.getDouble("acak");
   int id_kromosom = getProbabilitas.getInt("id");
   if(bilAcak < pc ){ // cek jika bil.acak <Pc maka terpilih sebagai induk
       aplikasi.insert_orangtua(individu,id_kromosom );
       }
    }
```

Gambar 4.30 Kode program untuk memilih individu sebgai orang tua

ii. Memeriksa jumlah orang tua yang terpilih. Jika jumlahnya kurang dari 2 maka ulangi proses rekombinasi dari awal. Jika jumlah orang tua lebih dari dua maka periksa apakah jumlah orang tua bernilai genap atau ganjil. Jika genap maka hitung jumlah pasangan orang tua dan jika ganjil maka jumlah orang tua dikurangi 1 kemudian hitung jumlah pasangan orang tua.

```
//memilih pasangan oranng tua
   int ot = aplikasi.count_orangtua(); //hitung jumlah orangtua
   if (ot<=1){ //jika jml orangtua <=1 maka ulangi proses dari awal
       continue probabilitas;
    }
else {
   if (ot%2==0){ // jika jml orangtua genap, hitung jml pasangan orangtua
     int jum_pasangan = ot/2;
     int batas_ot = 2;
     for ( int t=1; t<=jum_pasangan; t++){
     int posisi_ot = (t-1)*batas_ot;
     ResultSet pasangan = aplikasi.get_orangtua(posisi_ot, batas_ot);
     while (pasangan.next()){
         int individu_ot = pasangan.getInt("individu");
         int id_kromosom = pasangan.getInt("id_kromosom");
         aplikasi.insert_pasangan_orangtua(t, individu_ot, id_kromosom );
          }
        }
     } else {
       int ot2 = ot-1; //jika ganjil kurangi 1, hitung jml pasangan orangtua
         int jum_pasangan = ot2/2;
         int batas_ot = 2;
         for ( int t=1; t<=jum_pasangan; t++){
            int posisi_ot = (t-1)*batas_ot;
         ResultSet pasangan = aplikasi.get_orangtua(posisi_ot, batas_ot);
         while (pasangan.next()){
             int individu_ot = pasangan.getInt("individu");
             int id_kromosom = pasangan.getInt("id_kromosom");
             aplikasi.insert_pasangan_orangtua(t, individu_ot, id_kromosom);
       }
                   }
               }
```

Gambar 4.31. Kode program untuk memeriksa jumlah orangtua dan jumlah pasangan orangtua

c. Mengambil pasangan orangtua untuk direkombinasi

```
int[]individu = new int[100];
int[]orangtua1 = new int[100];
int[]orangtua2 = new int[100];
int random_y1;
int random_y2;
int jum_pas = aplikasi.jumlah_pasangan();
int jumlah_pasangan = jum_pas/2;
for (int ps=1; ps<=jumlah_pasangan; ps++){
individu = individu(ps);
 int id1 = individu[0];
 int id2 = individu[1];
 int id_kromosom1 = aplikasi.id_kromosom(ps, id1);
 int id_kromosom2 = aplikasi.id_kromosom(ps, id2);
 if (id1!=id2){
 orangtua1 = gen(id1,id_kromosom1);
 orangtua2 = gen(id2,id_kromosom2);
```

Gambar 4.32. Kode program untuk mengambil pasangan orangtua yang akan direkombinasi

d. Membangkitkan dua titik secara acak untuk masing-masing range kelas.

```
 int min = aplikasi.min_slot();
 int max = aplikasi.max_slot();
 int jarak = (max-min)+1;
 int jum_kls = aplikasi.jumlah_kelas();
 int s=1;
 loop:
    while (s<=jum_kls){
    int j = s-1;
    int x1 = min + (j*jarak); //batas slot awal kelas
    int x2 = max + (j*jarak);//batas slot akhir kelas
//--------------proses loop--------->
 random_y1 = random_range(x1, x2); //Titik potong 1
 random_y2 = random_range(x1, x2); //Titik potong 2
 if (random_y1==random_y2){
 s=s;
 continue loop;
  }
```

Gambar 4.33 Kode program untuk membangkitkan dua titik secara acak untuk tiap-tiap range kelas

e. Melakukan pindah silang antara orangtua untuk menghasilkan anak. Untuk melakukan pindah silang, proses pertama yang dilakukan adalah membangkitan 2 titik potong secara acak di antara range kelas. Kemudian isi slot diantara dua titik tersebut di kopikan ke slot anak dengan posisi titik yang sama. Selanjutnya untuk menghasilkan anak2, isi slot orangtua1 dibandingkan dengan isi slot diantara 2 titik potong orangtua2. Isi slot orangtua 1 yang akan dibandingkan dimulai dari titik setelah titik potong 2 orangtua1 ( ). Jika isi slot pada titik nilainya ada yang sama dengan isi slot diantara 2 titik potong orantua2 maka dilanjutkan ke pembandingan titik , jika nilainya tidak sama maka isi slot titik dikopikan ke anak2 dengan posisi dimulai dari setelah titik potong 2, begitu seterusnya sampai semua isi slot orangtua1 yang nilainya tidak sama dengan isi slot diantara 2 titik potong orangtua2 dikopikan ke anak2. Proses ini juga berlaku untuk menghasilkan anak1 dengan membandingkan orangtua2 terhadap orangtua1.

```
if (random_y2 > random_y1){
int selisih =random_y2-random_y1;
if (selisih>=5 && selisih<=10){
for (int g=random_y1; g<=random_y2; g++){
int gen1 = orangtua1[g];
int gen2 = orangtua2[g];

aplikasi.insert_anak(id_kromosom1, id1, g, gen1);
aplikasi.insert_anak(id_kromosom2, id2, g, gen2);
  }
int TP1 = random_y1;
int TP2 = random_y2;
int x2a = TP2 +1;

for (int g=x2a; g<=x2; g++){//ortu1 dibandingkan ortu2 menghasilkan anak2
int jumlah_gen = 0;
for (int h=TP1; h<=TP2; h++){
if (orangtua1[g]==orangtua2[h]){
jumlah_gen=jumlah_gen+1;
            }
     }
 int jumlah_gen_ot = aplikasi.jumlah_gen_ot(id1, id_kromosom1, orangtua1[g],
s);
if(jumlah_gen<jumlah_gen_ot){
jumlah_gen_ot = jumlah_gen_ot-1;
aplikasi.update_gen(jumlah_gen_ot, id1, s, orangtua1[g],id_kromosom1 );
```

```
int no_slot = aplikasi.max_noslot(id2, id_kromosom2, x2, TP2);
int slot_awal = aplikasi.slot_awal(id2,id_kromosom2, x1);
if (no_slot<x2){
int noslot = no_slot+1;

aplikasi.insert_anak(id_kromosom2,id2, noslot, orangtua1[g]);
}else if ((no_slot==x2)&&(slot_awal==0)){
int noslot = x1;

aplikasi.insert_anak(id_kromosom2,id2, noslot, orangtua1[g]);
}else if ((no_slot==x2)&&(slot_awal==1)){
int no_slot2 = aplikasi.max_noslot(id2,id_kromosom2, TP0, x1);
if (no_slot2<TP1){
no_slot2 = no_slot2+1;

aplikasi.insert_anak(id_kromosom2,id2, no_slot2, orangtua1[g]);
                }
              }
            }
          }

   for (int g=x1; g<=TP2; g++){
   int jumlah_gen = 0;
   for (int h=TP1; h<=TP2; h++){
   if (orangtua1[g]==orangtua2[h]){
   jumlah_gen=jumlah_gen+1;
            }
         }
 int jumlah_gen_ot = aplikasi.jumlah_gen_ot(id1, id_kromosom1, orangtua1[g],
s);
 if(jumlah_gen<jumlah_gen_ot){
 jumlah_gen_ot = jumlah_gen_ot-1;

aplikasi.update_gen(jumlah_gen_ot, id1, s, orangtua1[g], id_kromosom1);
int no_slot = aplikasi.max_noslot(id2,id_kromosom2, x2, TP2);
int slot_awal = aplikasi.slot_awal(id2,id_kromosom2, x1);
if (no_slot<x2){
int noslot = no_slot+1;

aplikasi.insert_anak(id_kromosom2,id2, noslot, orangtua1[g]);
    }else if ((no_slot==x2)&&(slot_awal==0)){
     int noslot = x1;
     aplikasi.insert_anak(id_kromosom2, id2, noslot, orangtua1[g]);
    }else if ((no_slot==x2)&&(slot_awal==1)){
     int no_slot2 = aplikasi.max_noslot(id2,id_kromosom2, TP0, x1);
     if (no_slot2<TP1){
     no_slot2 = no_slot2+1;
     aplikasi.insert_anak(id_kromosom2,id2, no_slot2, orangtua1[g]);
            }
         }
    }
  }
```

Gambar 4.34 Kode program untuk melakukan pindah silang antara orangtua1 dengan orangtua2 untuk menghasilkan anak2

f. Mengganti individu lama (orang tua) dengan individu baru (anak).

```
//update atau mengganti individu lama dengan individu baru hasil crossover
ResultSet individu_anak = aplikasi.get_individu_anak();
while (individu_anak.next()){
  int anak_individu = individu_anak.getInt("individu");
  int id_kromosom = individu_anak.getInt("id_kromosom");
  aplikasi.deleteindividu(anak_individu,id_kromosom);
  ResultSet anak = aplikasi.get_anak(anak_individu, id_kromosom);
  while(anak.next()){
    int isi_slot = anak.getInt("isi_slot");
    int no_slot = anak.getInt("no_slot");
    int ind = anak.getInt("individu");
    int kromosom = anak.getInt("id_kromosom");
    ResultSet get_batas = aplikasi.get_batas();
    while (get_batas.next()){
      int bawah = get_batas.getInt("batas_bawah");
      int atas = get_batas.getInt("batas_atas");
      int k_kelas = get_batas.getInt("kelas");
      if(no_slot>=bawah && no_slot<=atas){
      aplikasi.insert_new_individu(ind, isi_slot, k_kelas,kromosom );
   } }} }}
 o++;
        }
  }
}
```

Gambar 4.35 Kode program untuk menggantikan individu lama dengan individu baru

7. Mutasi

Proses Mutasi terdapat pada fungsi `mutasi();`.Langkah-langkah untuk melakukan mutasi pada fungsi ini antara lain:

a. Menentukan nilai Ps, Pm dan membangkitkan bilangan acak

```
//generate nilai random yang dibandingkan dengan Probabilitas Seleksi Mutasi
   double Ps=0.1;
   double Pm=0.001;
   ResultSet get_individu_baru = aplikasi.get_individu_baru();
    while (get_individu_baru.next()){
         int individu_mutasi = get_individu_baru.getInt("id_populasi");
         int id_kromosom = get_individu_baru.getInt("id_kromosom");
         Random randoms = new Random();
         double rands = randoms.nextDouble();
         aplikasi.insert_seleksi_mutasi(id_kromosom, individu_mutasi, Ps, rands);}
```

Gambar 4.36 Kode program untuk menentukan nilai Ps, Pm dan membangkitkan bilangan acak

b. Menentukan individu yang akan mengalami mutasi dengan cara memeriksa nilai bilangan acak. Jika bilangan acak individu tersebut kurang dari nilai Ps, maka individu tersebtu terpipih untuk mengalami mutasi.

```
//mencari individu yang akan di mutasi (bil acak<ps)
ResultSet getSeleksiMutasi = aplikasi.get_seleksi_mutasi();
  while (getSeleksiMutasi.next()){
     int id_kromosom = getSeleksiMutasi.getInt("id_kromosom");
     int individu_mutasi = getSeleksiMutasi.getInt("individu");
     double ps = getSeleksiMutasi.getDouble("ps");
     double bil_acak = getSeleksiMutasi.getDouble("acak");
     if(bil_acak < ps ){
         aplikasi.insert_individu_mutasi(individu_mutasi,id_kromosom );
                  }
               }
```

Gambar 4.37 Kode program untuk menentukan individu yang mengalami mutasi

c. Menghitung jumlah bit yang mengalami mutasi. Untuk menghitung jumlah bit dilakukan dengan menghitung jumlah total bit kemudian dikalikan dengan nilai Pm.

```
    //menghitung bit mutasi
     int total_kelas = aplikasi.total_kelas();
     int jumlah_slot = aplikasi.jumlah_slot();
     int range_slot = total_kelas*jumlah_slot;
     int popsize = jumlah_iterasi;
     int total_bit = range_slot*popsize;
     int bit_mutasi = (int)(Pm*total_bit);
```

Gambar 4.38 Kode program untuk menghitung bit yang menglami mutasi

d. Melakukan proses mutasi. Sebelum melakukan proses mutasi program akan memeriksa terlebih dahulu berapa jumlah individu yang akan mengalami mutasi, jika tidak ada individu yang akan mengalami mutasi, maka proses mutasi diulang dari awal. Jika ada maka program akan memeriksa apakah bit mutasi bernilai genap atau ganjil, jika bernilai ganjil maka nilai bit mutasi akan dikurangi 1. Langkah selanjutnya adalah nilai bit mutasi dibagi 2 untuk mengetahui jumlah pasangan untuk mutasi. Jumlah pasangan digunakan sebagai pembanding dengan jumlah individu. Misal, jika jumlah pasangan bit yang akan dimutasi ada 4 dan jumlah individu yang mengalami mutasi ada 2 maka masing-masing individu akan memiliki 2 pasangan bit yang akan dimutasi (*swap*).

```
//menghitung jumlah kelas yang akan dimutasi serta menghitung
// berapa titik yang akan dimutasi untuk masung-masing kelas
     int jum_individu_mutasi = aplikasi.count_individu_mutasi();
     if (jum_individu_mutasi==0){
         e=1;
         continue loop_mutasi;
     } else {
       if (bit_mutasi%2==0){
       int pasang_bit = bit_mutasi/2;
        if(pasang_bit>jum_individu_mutasi){
           int swap = pasang_bit/jum_individu_mutasi;
           ResultSet individu_mutasi = aplikasi.get_individu_mutasi();
           while (individu_mutasi.next()){
              int ind_mutasi = individu_mutasi.getInt("individu");
              int id_kromosom = individu_mutasi.getInt("id_kromosom");
              aplikasi.insert_swap(ind_mutasi, id_kromosom, swap);
           }
int sisa_swap = pasang_bit%jum_individu_mutasi;
ResultSet jumlah_swap = aplikasi.get_jumlah_swap(sisa_swap);
while (jumlah_swap.next()){
   int ind_swap = jumlah_swap.getInt("individu");
   int id_kromosom = jumlah_swap.getInt("id_kromosom");
   int jum_swap = jumlah_swap.getInt("jumlah_swap");
   int tot_swap = jum_swap + 1;
   aplikasi.update_jumlah_swap(tot_swap, ind_swap,id_kromosom );
   }
}else if(pasang_bit==jum_individu_mutasi){   int swap =
pasang_bit/jum_individu_mutasi;
   ResultSet individu_mutasi = aplikasi.get_individu_mutasi();
   while (individu_mutasi.next()){
   int ind_mutasi = individu_mutasi.getInt("individu");
   int id_kromosom = individu_mutasi.getInt("id_kromosom");
   aplikasi.insert_swap(ind_mutasi, id_kromosom, swap);
   }
}else if(pasang_bit<jum_individu_mutasi){
 ResultSet ind_mutasi_limit = aplikasi.get_individu_mutasi_limit(pasang_bit);
 while (ind_mutasi_limit.next()){
 int ind_limit = ind_mutasi_limit.getInt("individu");
 int id_kromosom = ind_mutasi_limit.getInt("id_kromosom");
 int jum_swap = 1;
 aplikasi.insert_swap(ind_limit,id_kromosom, jum_swap);
       }
    }
 }
```

Gambar 4.39 Kode program untuk menentukan jumlah pasang bit dan jumlah *swap*.

Setelah individu yang mengalami mutasi diketahui jumlah pasangan bitnya yang mengalami mutasi (*swap*), program akan memilih secara acak range kelas yang akan mengalami mutasi. Di dalam range kelas tersebut akan dibangkitkan dua titik secara acak dan isi slot dari dua titik tersebut ditukarkan.

```
ResultSet get_swap = aplikasi.get_swap();
while(get_swap.next()){
  int ind_swap = get_swap.getInt("individu");
  int id_kromosom = get_swap.getInt("id_kromosom");
  ResultSet get_individiu_swap = aplikasi.get_individu_swap(ind_swap,
id_kromosom );
  int awal=0;
  while (get_individiu_swap.next()){
     int isi_slot = get_individiu_swap.getInt("isi_slot");
     int id_kelas = get_individiu_swap.getInt("id_kelas");
     aplikasi.insert_gen_swap(ind_swap, id_kromosom, awal, isi_slot, id_kelas);
     awal++;
   }
int jum_swap = get_swap.getInt("jumlah_swap");
int u=1;
loop_acak:
while (u<=jum_swap){
   ResultSet kelas_rand = aplikasi.get_kelas_rand(); //memilih range kelas acak
   while (kelas_rand.next()){
     int kode_kelas = kelas_rand.getInt("kode_kelas");
     ResultSet get_batas = aplikasi.get_batas(kode_kelas);
     while (get_batas.next()){
       int bawah = get_batas.getInt("batas_bawah");
       int atas = get_batas.getInt("batas_atas");
       int acak1 = random_range(bawah, atas);
       int acak2 = random_range(bawah, atas);
       if (acak1==acak2){
           u=u;
           continue loop_acak;
      }
else if (acak1!=acak2){ //membangkitkan 2 titik dlm range kelas secara acak
   int isi_slot1 = aplikasi.nilai_swap(ind_swap, acak1,id_kromosom );
   int isi_slot2 = aplikasi.nilai_swap(ind_swap, acak2,id_kromosom);
   aplikasi.update_gen_swap(isi_slot1, ind_swap, acak2, id_kromosom);
   aplikasi.update_gen_swap(isi_slot2, ind_swap, acak1, id_kromosom);
    }}}
     u++;
   }
  aplikasi.deleteindividu(ind_swap, id_kromosom );
  ResultSet get_all_swap = aplikasi.get_all_swap(ind_swap,id_kromosom );
  while (get_all_swap.next()){
    int isi_slot = get_all_swap.getInt("isi_slot");
    int id_kelas = get_all_swap.getInt("id_kelas");
    int id_krom = get_all_swap.getInt("id_kromosom");
    aplikasi.insert_individu_swap(ind_swap, isi_slot, id_kelas, id_krom);
       }}}
       e++ ;
     } aplikasi.deleteindividu1();
ResultSet get_kromosom_final = aplikasi.get_kromosom_final();
while (get_kromosom_final.next()){
  int id = get_kromosom_final.getInt("id_kromosom");
  int isi_slot = get_kromosom_final.getInt("isi_slot");
  int id_kelas = get_kromosom_final.getInt("id_kelas");
  aplikasi.insert_individu(id, isi_slot, id_kelas);
      }}}
```

Gambar 4.40 Kode program pertukaran sepasang gen pada kelas yang termutasi untuk masing-masing individu

8. Elitisme

Proses Elitisme terdapat pada fungsi `elitisme();`. Pada fungsi elitisme proses yang berjalan adalah:

a. Memeriksa nilai fitness seluruh individu dalam 1 populasi kemudian mengambil nilai fitness terkecil dan diabndingkan dengan nilai fitness terbaik dari generasi sebelumnya. Jika nilai fitness terkecil tersebut lebih kecil maka akan diganti dengan nilai fitness terbaik dari generasi sebelumnya. Jika nilai fitness terkecil tersebut lebih besar dari nilai fitness terbaik dari generasi sebelumnya, maka lanjut ke proses berikutnya

```
fitness();
ResultSet min_fitness = aplikasi.get_min_fitness();
while (min_fitness.next()){
    int individu_min = min_fitness.getInt("id_populasi");
    double fitness_min = min_fitness.getInt("fitness");
    int generasi_terbaik = v-1;
    double get_fitness_terbaik =
    aplikasi.get_fitness_terbaik(generasi_terbaik);
   if (fitness_min<get_fitness_terbaik){
       aplikasi.deleteindividu(individu_min);
      aplikasi.update_fitness(get_fitness_terbaik, individu_min);
      ResultSet individu_terbaik2  =
      aplikasi.individu_terbaik(generasi_terbaik);
     while (individu_terbaik2.next()){
        int isi_slot = individu_terbaik2.getInt("isi_slot");
        int id_kelas = individu_terbaik2.getInt("id_kelas");
        aplikasi.insert_individu(individu_min, isi_slot, id_kelas);
        }
    }
}
```

Gambar 4.41 Kode program untuk proses penggantian individu dengan nilai fitness terkecil dengan individu dengan nilai fitness terbaik dari generasi sebelumnya

b. Setelah proses diatas, selanjutnya adalah memeriksa nilai fitness terbaik dari generasi saat ini untuk diambil dan disimpan dan digunakan dalam proses elitisme pada generasi berikutnya.

```
ResultSet max_fitness = aplikasi.get_max_fitness();
while (max_fitness.next()){
    int individu_max = max_fitness.getInt("id_populasi");
    double fitness2 = max_fitness.getDouble("fitness");
    aplikasi.insert_fitness_terbaik(v, individu_max, fitness2);
    ResultSet get_individu_terbaik =
    aplikasi.get_individu_terbaik(individu_max);
    while (get_individu_terbaik.next()){
        int isi_slot = get_individu_terbaik.getInt("isi_slot");
        int id_kelas = get_individu_terbaik.getInt("id_kelas");
        aplikasi.insert_individu_terbaik(v, individu_max, isi_slot,
id_kelas);
    }}}}
```

Gambar 4.42 proses penyimpanan nilai fitness terbaik untuk proses elitisme pada generasi berikutnya

9. Mendapatkan individu terbaik

Proses mendapatkan individu terbaik terdapat pada fungsi `get_individu_final();` Setelah proses iterasi mencapai generasi terakhir, sistem akan mengambil individu terbaik untuk dimasukkan kedalam tabel individu_final.

```
ResultSet get_fitness_final = aplikasi.get_fitness_final();
 while(get_fitness_final.next()){
     int generasi_final = get_fitness_final.getInt("generasi");
     ResultSet get_kelas = aplikasi.getkelas();
     while (get_kelas.next()){
         int ko_kls = get_kelas.getInt("kode_kelas");
         ResultSet get_individu_final =
aplikasi.get_individu_final(generasi_final, ko_kls);
         int slot = 0;
while (get_individu_final.next()){
   int isi_slot = get_individu_final.getInt("isi_slot");
   int id_kelas = get_individu_final.getInt("id_kelas");
   aplikasi.insert_individu_final(slot, isi_slot, id_kelas);
   slot++;
         }
     }
   }
}
}
```

Gambar 4.43 Kode program untuk mendapatkan individu terbaik

10. Membentuk Jadwal Mata Pelajaran

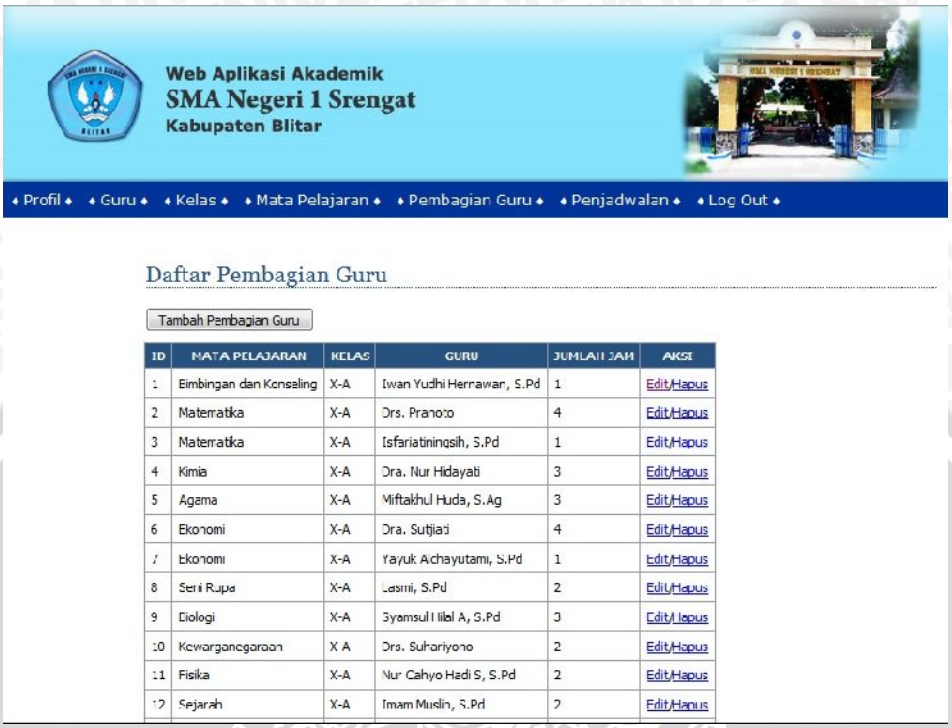
Proses membentuk jadwal mata pelajaran terdapat pada fungsi `get_jadwal_mapel();`. Jadwal mata pelajaran dibentuk dari individu terbaik. Setelah individu tebaik didapatkan, proses elanjtunya adalah mengkopikan isi slot dari individu terbaik kedalam tabel jadwal_mapel. Isi slot yang nilai kodenya kurang dari 700 akan dimasukkan sebanyak dua kali, sedangkan isi slot yang kodenya lebih dari 700 akan dilihat kode gurunya di tabel kode_gabung dan isi kode guru tersebut dimasukkan ke tabel jadwal_mapel.

```
ResultSet getkelas = aplikasi.getkelas();
while (getkelas.next()){
   int k_kls = getkelas.getInt("kode_kelas");
   ResultSet get_hasil_individu = aplikasi.get_hasil_individu(k_kls);
   while (get_hasil_individu.next()){
      int slot = get_hasil_individu.getInt("slot");
      int isi_slot = get_hasil_individu.getInt("isi_slot");
      ResultSet get_slot = aplikasi.get_slot(slot);
      while (get_slot.next()){
      String hari = get_slot.getString("hari");
      int jam = get_slot.getInt("jam");
if(isi_slot<700){
   int d=1;
   while (d<=2){
       String hari_final = hari;
       int jam_final = jam;
       int id_guru_final = isi_slot;
       aplikasi.insert_jadwal_mapel(k_kls, hari_final, jam_final,
id_guru_final);
       d++;
    }
  }else if(isi_slot>700){
     ResultSet getkodegabung = aplikasi.getkodegabung(isi_slot);
     while(getkodegabung.next()){
       int id_guru_final = getkodegabung.getInt("id_guru");
       String hari_final = hari;
       int jam_final = jam;
       aplikasi.insert_jadwal_mapel(k_kls, hari_final, jam_final,
id_guru_final);
              }
           }
         }
     }
   }
}
}
```

Gambar 4.44 kode program untuk membentuk jadwal mata pelajaran

4.2.4 Implementasi Antarmuka

1. Tampilan Antarmuka Pemasukan Data Pembagian Guru

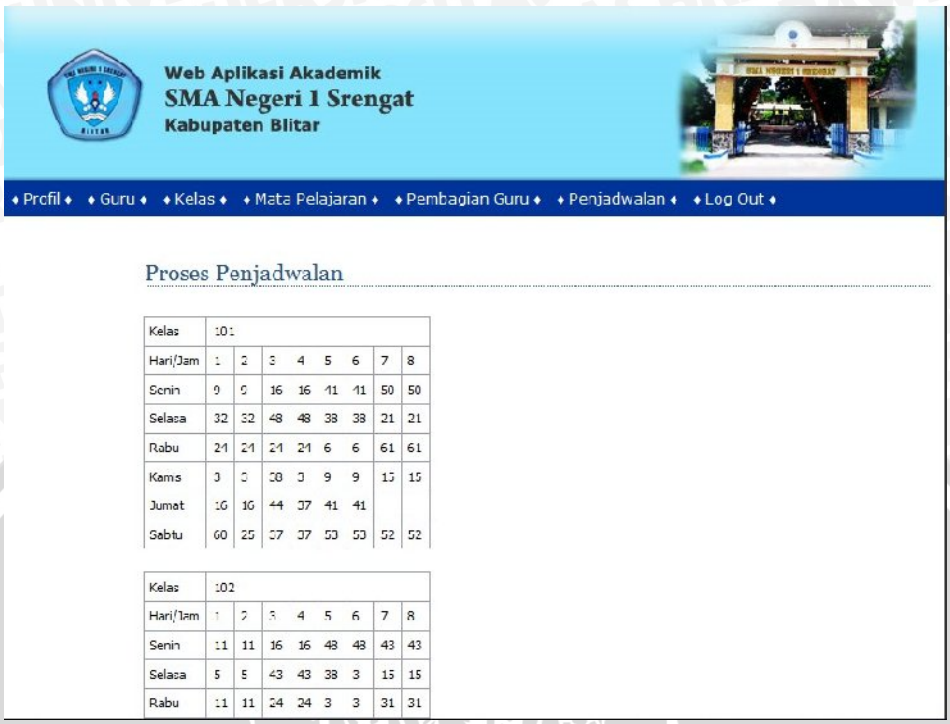


Gambar 4.45 Antarmuka Pemasukan Data Pembagian Guru

2. Tampilan Antarmuka Proses Penjadwalan

Gambar 4.46 Antarmuka Proses Penjadwalan

3. Tampilan Hasil Proses Penjadwalan



Gambar 4.47 Antarmuka Hasil Proses Penjadwalan

BAB V
PENGUJIAN

Pada pengujian, aplikasi diuji dengan menguji semua elemen program yang dibangkitkan dengan mengetahui struktur internal (kode sumber) dari unit terkecil aplikasi (pengujian unit atau *white box testing*) kemudian melakukan pengujian dengan menggabung semua unit-unit tersebut (pengujian integrasi) serta mengecek fungsionalitas aplikasi bekerja dengan baik (pengujian validasi atau *black box testing*).

5.1 Pengujian Unit

Pengujian Unit dilakukan pada tahap implementasi perancangan ke dalam kode pemrograman, yaitu dengan menjalankan unit program (modul) yang dibuat. Setiap modul langsung diuji ketika selesai dikodekan ke dalam bahasa pemrograman dengan memberikan input yang dibutuhkan kemudian diperiksa output berupa data yang masuk ke dalam database atau tampilan yang ditampilkan oleh *interface*. Jika terdapat kesalahan output atau hasil output tidak benar maka perlu ditinjau kembali dan dilakukan perbaikan.

Tabel 5.1 Pengujian Unit

Unit Program	Hasil Keluaran	Kesimpulan
Pembentukan populasi awal	Populasi awal terbentuk dari dari tabel pembagian guru yang tiap-tiap nilai slotnya mewakili nama guru dan mata pelajaran	Benar
Penghitungan nilai fitness	Menghasilkan nilai fitness yang sesuai dengan fungsi fitnessnya	Benar
Seleksi Orangtua (induk)	Menghasilkan beberapa kandidat induk yang diperoleh dari dengan nilai fitnessnya berdasarkan metode *roulette-wheel*	Benar
Rekombinasi	Dapat melakukan proses pertukaran gen dan menghasilkan anak dari kromosom induk yang terpilih sesuai dengan metode order crossover	Benar
Mutasi	Dapat melakukan proses pertukaran nilai sepasang	Benar

	gen dalam 1 kromosom berdasarkan metode *swap mutation*	
Elitisme	Dapat menyimpan individu dengan nilai fitness terbaik dari sekumpulan individu dalam 1 generasi	Benar
Pembentukan jadwal	Dapat mengambil individu terbaik dari total generasi untuk dibentuk jadwal mata pelajaran	Benar

5.2 Pengujian Integrasi

Pengujian integrasi dilakukan setelah melakukan semua pengujian unit dan menghasilkan pengujian unit yang benar dan sesuai harapan.

Pengujian dilakukan dengan menguji semua unit program (modul) yang telah dipadukan dengan memberikan nilai input yang dibutuhkan dan memeriksa hasil output yang diperoleh yang ditamplikan oleh *interface*. Jika terdapat kesalahan atau hasil output tidak sesuai dengan harapan maka perlu ditinjau kembali dan dilakukan perbaikan.

Tabel 5.2 Pengujian Integrasi

Program terintegrasi	Data input	Hasil output	kesimpulan
Proses penjadwalan mata pelajaran	Data pembagian guru yang berisi kode guru dan jumlah jam mengajar	Sistem menampilkan jadwal mata pelajaran yang berisikan kode guru	Sesuai dengan harapan pengguna

5.3 Pengujian Validasi Algoritma Genetika Untuk Penjadwalan Mata Pelajaran

Dalam penjadwalan mata pelajaran menggunakan algoritma genetika harus sesuai dengan kriteria penjadwalan pada SMAN 1 Srengat. Kriteria penjadwalan di sini meliputi tidak adanya bentrok mengajar guru, mata pelajaran olahraga terletak pada jam pertama-jam keempat, mata pelajaran matematika dan fisika diharapakan terletak pada jam pertama-jam keempat, guru diharapkan tidak mengajar lebih dari 6 jam dalam 1 hari. Berikut hasil dari dua pengujian validasi untuk penjadwalan menggunakan algoritma genetika.

1. Pengujian pertama, parameter yang digunakan selama proses pengujian validasi adalah:

- Jumlah Generasi = 10
- Jumlah Populasi = 10
- Probabilitas Crossover, = 0,7
- Probabilitas Seleksi Mutasi, = 0,1 dan Probabilitas Mutasi, = 0,001
- Jumlah Kelas = 23 kelas

2. Pengujian kedua, parameter yang digunakan selama proses pengujian validasi adalah:

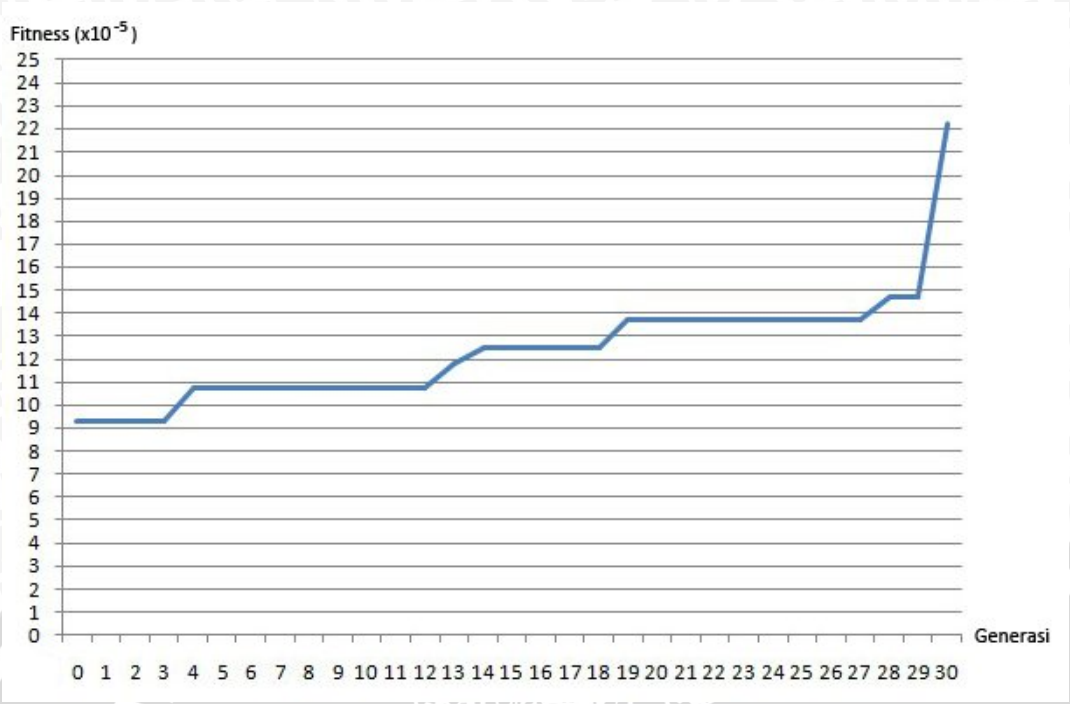
- Jumlah Generasi = 30
- Jumlah Populasi = 10
- Probabilitas Crossover, = 0,7
- Probabilitas Seleksi Mutasi, = 0,1 dan Probabilitas Mutasi, = 0,001
- Jumlah Kelas = 4 kelas

Hasil dari pengujian validasi dengan parameter tersebut dapat dilihat dari tabel dibawah ini:

1. Nilai Fitness terbaik dari tiap-tiap generasi

Tabel 5.3 Nilai Fitness terbaik dari tiap-tiap generasi dengan parameter pertama

Id	Generasi	Id_Individu	Fitness
1	0	1	8.30564783363318 x 10
2	1	2	8.30564783363318 x 10
3	2	10	8.30564783363318 x 10
4	3	4	8.4602368794526 x 10
5	4	4	9.63391135873419 x 10
6	5	4	9.63391135873419 x 10
7	6	2	9.63391135873419 x 10
8	7	7	9.63391135873419 x 10
9	8	8	9.63391135873419 x 10
10	9	6	9.63391135873419 x 10
11	10	2	9.63391135873419 x 10



Grafik 5.1 Nilai Fitness terbaik dari tiap-tiap generasi dengan parameter kedua

2. Nilai fitness pada generasi terakhir

Tabel 5.5 Nilai fitness pada generasi terakhir dengan parameter pertama

Id_fitness	Id_individu	Fitness
1	1	8.03858520254133 x 10
2	2	9.63391135873419 x 10
3	3	7.67459707776316 x 10
4	4	7.27272726743802 x 10
5	5	7.67459707776316 x 10
6	6	9.63391135873419 x 10
7	7	8.00640511768903 x 10
8	8	8.20344544035812 x 10
9	9	7.79423226204658 x 10
10	10	7.27272726743802 x 10

Tabel 5.6 Nilai fitness pada generasi terakhir dengan parameter kedua

Id_fitness	Id_individu	Fitness
1	1	0.000222222217283951
2	2	8.19672124428917 x 10
3	3	0.000147058821366782
4	4	0.000136986299493338
5	5	0.000105263156786704
6	6	0.000149253729115616
7	7	0.000149253729115616
8	8	0.000103092782442342
9	9	0.000103092782442342
10	10	5.81395345457004 x 10

3. Jumlah pelanggaran pada generasi terakhir

Tabel 5.7 Jumlah pelanggaran pada generasi terakhir dengan parameter pertama

Id_ fitness	Nama	Id_ Individu	Pelanggaran	Id_ fitness	Nama	Id_ Individu	Pelanggaran
1	h1	1	94	26	h1	6	77
2	h2	1	12	27	h2	6	10
3	h3	1	33	28	h3	6	28
4	h4	1	9	29	h4	6	9
5	h5	1	26	30	h5	6	25
6	h1	2	116	31	h1	7	96
7	h2	2	13	32	h2	7	11
8	h3	2	33	33	h3	7	36
9	h4	2	8	34	h4	7	7
10	h5	2	24	35	h5	7	20
11	h1	3	95	36	h1	8	90
12	h2	3	13	37	h2	8	9
13	h3	3	36	38	h3	8	39
14	h4	3	12	39	h4	8	11
15	h5	3	35	40	h5	8	28
16	h1	4	107	41	h1	9	99
17	h2	4	9	42	h2	9	9
18	h3	4	35	43	h3	9	35
19	h4	4	11	44	h4	9	12
20	h5	4	29	45	h5	9	21
21	h1	5	95	46	h1	10	107
22	h2	5	13	47	h2	10	9
23	h3	5	36	48	h3	10	35
24	h4	5	12	49	h4	10	11
25	h5	5	35	50	h5	10	29

Tabel 5.8 Jumlah pelanggaran pada generasi terakhir dengan parameter kedua

Id_ fitness	Nama	Id_ Individu	Pelanggaran	Id_ fitness	Nama	Id_ Individu	Pelanggaran
1	h1	1	1	26	h1	6	4
2	h2	1	1	27	h2	6	1
3	h3	1	4	28	h3	6	3
4	h4	1	2	29	h4	6	2
5	h5	1	2	30	h5	6	0
6	h1	2	7	31	h1	7	4
7	h2	2	2	32	h2	7	1
8	h3	2	6	33	h3	7	3
9	h4	2	3	34	h4	7	2
10	h5	2	1	35	h5	7	0
11	h1	3	15	36	h1	8	4
12	h2	3	2	37	h2	8	3
13	h3	3	7	38	h3	8	6
14	h4	3	1	39	h4	8	2
15	h5	3	2	40	h5	8	2
16	h1	4	3	41	h1	9	4
17	h2	4	1	42	h2	9	3
18	h3	4	8	43	h3	9	6
19	h4	4	1	44	h4	9	2
20	h5	4	0	45	h5	9	2
21	h1	5	6	46	h1	10	11
22	h2	5	1	47	h2	10	4
23	h3	5	6	48	h3	10	5
24	h4	5	0	49	h4	10	2
25	h5	5	2	50	h5	10	3

4. Penyusunan jadwal dari individu terbaik

Penyusunan jadwal mata pelajaran diperoleh dari nilai fitness terbaik dari generasi terakhir. Dari tabel 5.5, nilai fitness terbaik diperoleh dari individu ke-2 dengan nilai 9.63391135873419 x 10 dan dari tabel 5.6, nilai fitness terbaik diperoleh dari individu ke-1 dengan nilai 0.000222222217283951. Sedangkan dari tabel 5.7 dan 5.8 diperoleh jumlah pelanggaran pada individu dengan nilai fitness terbaik.

Persentase pelanggaran yang terjadi pada individu tersebut dapa dilihat pada tabel

berikut:

Tabel 5.9 Persentase pelanggaran pada individu terbaik dengan parameter pertama

No	Nama	Jumlah Pelanggaran	Total Pertemuan	Persentase Pelanggaran
1	h1	77	529	14, 5%
2	h2	10	529	1,89%
3	h3	28	529	5,29%
4	h4	9	529	1,7%
5	h5	25	529	4,73%

Tabel 5.10 Persentase pelanggaran pada individu terbaik dengan parameter kedua

No	Nama	Jumlah Pelanggaran	Total Pertemuan	Persentase Pelanggaran
1	h1	1	92	1,08%
2	h2	1	92	1,08%
3	h3	4	92	4,34%
4	h4	2	92	2,17%
5	h5	2	92	2,17%

Untuk hasil penjadwalan dari individu dengan nilai fitness terbaik dapat dilihat dari gambar dibawah ini:

Kelas	101							
Hari/Jam	1	2	3	4	5	6	7	8
Senin	5	5	30	30	38	38	41	41
Selasa	8	8	41	41	33	33	25	25
Rabu	16	16	2	2	49	49	46	46
Kamis	16	16	25	25	52	52	15	15
Jumat	21	21	56	25	41	33		
Sabtu	60	60	8	8	53	53	38	2

Kelas	103							
Hari/Jam	1	2	3	4	5	6	7	8
Senin	16	16	52	52	8	8	26	26
Selasa	38	2	26	26	46	46	16	16
Rabu	33	33	41	41	15	15	21	21
Kamis	49	49	41	33	56	25	53	53
Jumat	30	30	38	38	2	2		
Sabtu	8	8	60	60	5	5	41	41

Kelas	102							
Hari/Jam	1	2	3	4	5	6	7	8
Senin	56	25	53	53	25	25	60	60
Selasa	16	16	33	33	41	41	2	2
Rabu	52	52	49	49	8	8	38	2
Kamis	25	25	16	16	30	30	38	38
Jumat	41	41	21	21	15	15		
Sabtu	8	8	5	5	41	33	46	46

Kelas	104							
Hari/Jam	1	2	3	4	5	6	7	8
Senin	53	53	38	38	8	8	46	46
Selasa	21	21	41	41	27	27	56	28
Rabu	8	8	52	52	27	27	16	16
Kamis	38	2	5	5	49	49	41	33
Jumat	30	30	41	41	60	60		
Sabtu	33	33	16	16	2	2	15	15

Kelas	105							
Hari/Jam	1	2	3	4	5	6	7	8
Senin	41	41	49	49	60	60	8	8
Selasa	53	53	30	30	41	41	2	2
Rabu	46	46	16	16	34	34	56	25
Kamis	44	34	5	5	16	16	21	21
Jumat	25	25	15	15	8	8		
Sabtu	38	38	25	25	52	52	38	2

Kelas	107							
Hari/Jam	1	2	3	4	5	6	7	8
Senin	34	34	9	9	38	3	57	25
Selasa	52	52	60	60	25	25	17	17
Rabu	3	3	38	38	49	49	53	53
Kamis	15	15	32	32	9	9	5	5
Jumat	43	43	48	48	43	43		
Sabtu	17	17	25	25	43	34	21	21

Kelas	106							
Hari/Jam	1	2	3	4	5	6	7	8
Senin	38	2	17	17	5	5	25	25
Selasa	49	49	52	52	43	43	43	43
Rabu	17	17	46	46	2	2	60	60
Kamis	25	25	53	53	56	25	30	30
Jumat	43	34	34	34	15	15		
Sabtu	21	21	38	38	9	9	9	9

Kelas	108							
Hari/Jam	1	2	3	4	5	6	7	8
Senin	9	9	52	52	49	49	17	17
Selasa	21	21	34	34	9	9	39	39
Rabu	17	17	31	31	45	45	45	45
Kamis	26	26	57	26	15	15	44	34
Jumat	48	48	60	60	53	53		
Sabtu	5	5	3	3	39	3	26	26

Kelas	211							
Hari/Jam	1	2	3	4	5	6	7	8
Senin	60	60	32	32	26	26	15	15
Selasa	9	9	39	39	3	3	17	17
Rabu	26	26	35	35	39	39	35	35
Kamis	9	9	6	6	32	32	57	26
Jumat	53	53	35	32	32	32		
Sabtu	17	17	22	22	49	49	39	39

Kelas	213							
Hari/Jam	1	2	3	4	5	6	7	8
Senin	60	60	3	3	6	6	31	31
Selasa	26	26	39	39	31	31	50	50
Rabu	18	18	14	14	18	18	26	26
Kamis	39	39	39	39	57	26	36	36
Jumat	36	31	31	31	54	54		
Sabtu	10	10	10	10	22	22	36	36

Kelas	212							
Hari/Jam	1	2	3	4	5	6	7	8
Senin	9	9	35	31	50	50	17	17
Selasa	35	35	6	6	39	39	39	39
Rabu	22	22	9	9	31	31	57	26
Kamis	39	39	31	31	3	3	31	31
Jumat	14	14	53	53	26	26		
Sabtu	60	60	17	17	26	26	35	35

Kelas	221							
Hari/Jam	1	2	3	4	5	6	7	8
Senin	6	6	46	46	27	27	18	18
Selasa	10	10	55	55	50	50	42	42
Rabu	42	42	14	14	57	27	52	52
Kamis	46	46	18	18	54	54	10	10
Jumat	3	3	27	27	14	46		
Sabtu	61	61	50	50	42	42	22	22

Kelas	222							
Hari/Jam	1	2	3	4	5	6	7	8
Senin	10	10	22	22	42	42	50	50
Selasa	14	14	47	47	6	6	54	54
Rabu	55	55	42	42	50	50	18	18
Kamis	58	27	47	47	14	47	3	3
Jumat	27	27	27	27	10	10		
Sabtu	61	61	18	18	52	52	42	42

Kelas	224							
Hari/Jam	1	2	3	4	5	6	7	8
Senin	22	22	42	42	50	50	6	6
Selasa	18	18	14	14	15	47	15	15
Rabu	47	47	52	52	61	61	58	27
Kamis	14	14	50	50	47	47	42	42
Jumat	4	4	27	27	42	42		
Sabtu	55	55	54	54	18	18	27	27

Kelas	223							
Hari/Jam	1	2	3	4	5	6	7	8
Senin	18	18	6	6	50	50	58	27
Selasa	42	42	55	55	18	18	3	3
Rabu	54	54	14	47	10	10	14	14
Kamis	22	22	42	42	50	50	27	27
Jumat	42	42	27	27	47	47		
Sabtu	10	10	61	61	47	47	52	52

Kelas	225							
Hari/Jam	1	2	3	4	5	6	7	8
Senin	6	6	51	51	47	47	43	43
Selasa	47	47	28	28	4	4	51	51
Rabu	13	13	54	54	52	52	58	28
Kamis	11	11	4	13	19	19	28	28
Jumat	19	19	43	43	43	43		
Sabtu	61	61	11	11	55	55	22	22

Kelas	311							
Hari/Jam	1	2	3	4	5	6	7	8
Senin	40	40	36	36	11	11	35	35
Selasa	61	61	30	30	22	22	4	4
Rabu	14	14	11	11	28	28	40	40
Kamis	7	7	30	30	40	40	54	54
Jumat	19	19	45	45	19	19		
Sabtu	28	28	58	36	28	28	36	36

Kelas	313							
Hari/Jam	1	2	3	4	5	6	7	8
Senin	7	7	23	23	40	40	59	37
Selasa	11	11	37	37	11	11	40	40
Rabu	32	32	40	40	28	28	54	54
Kamis	61	61	44	44	28	28	37	37
Jumat	28	28	13	13	19	19		
Sabtu	19	19	37	37	4	4	32	32

Kelas	312							
Hari/Jam	1	2	3	4	5	6	7	8
Senin	32	32	11	11	28	28	61	61
Selasa	32	32	19	19	36	36	28	28
Rabu	28	28	37	37	13	13	11	11
Kamis	45	45	40	40	23	23	7	7
Jumat	19	19	40	40	54	54		
Sabtu	4	4	40	40	37	37	58	37

Kelas	321							
Hari/Jam	1	2	3	4	5	6	7	8
Senin	59	29	12	12	43	43	12	12
Selasa	61	61	51	51	4	4	26	13
Rabu	45	45	13	13	29	29	48	48
Kamis	43	43	7	7	43	43	29	29
Jumat	20	20	51	51	23	23		
Sabtu	55	55	48	48	20	20	43	43

Kelas	322							
Hari/Jam	1	2	3	4	5	6	7	8
Senin	13	13	12	12	44	44	55	55
Selasa	48	48	7	7	29	13	59	29
Rabu	4	4	20	20	44	44	61	61
Kamis	20	20	51	51	12	12	23	23
Jumat	44	44	44	44	29	29		
Sabtu	44	44	29	29	48	48	51	51

Kelas	323							
Hari/Jam	1	2	3	4	5	6	7	8
Senin	48	48	44	44	29	29	12	12
Selasa	44	44	61	61	13	13	48	48
Rabu	7	7	4	4	51	51	44	44
Kamis	29	29	55	55	44	44	44	44
Jumat	29	29	20	20	59	13		
Sabtu	12	12	23	23	51	51	20	20

Kelas	324							
Hari/Jam	1	2	3	4	5	6	7	8
Senin	48	48	20	20	7	7	13	13
Selasa	29	29	45	45	23	23	29	29
Rabu	51	51	12	12	61	61	29	29
Kamis	45	45	59	13	45	45	20	20
Jumat	51	51	55	55	45	45		
Sabtu	12	12	45	45	4	4	48	48

Gambar 5.1 Jadwal Mata Pelajaran yang Terbentuk dari Individu dengan Nilai Fitness Terbaik untuk parameter pertama

Kelas	101							
Hari/Jam	1	2	3	4	5	6	7	8
Senin	41	33	52	52	8	8	41	41
Selasa	15	15	25	25	16	16	38	2
Rabu	21	21	60	60	30	30	53	53
Kamis	25	25	5	5	16	16	49	49
Jumat	41	41	56	25	8	8		
Sabtu	2	2	38	38	46	46	33	33

Kelas	102							
Hari/Jam	1	2	3	4	5	6	7	8
Senin	15	15	16	16	5	5	41	33
Selasa	41	41	2	2	25	25	21	21
Rabu	30	30	49	49	41	41	60	60
Kamis	52	52	25	25	38	38	16	16
Jumat	8	8	8	8	56	25		
Sabtu	46	46	53	53	33	33	38	2

Kelas	103							
Hari/Jam	1	2	3	4	5	6	7	8
Senin	5	5	38	38	49	49	2	2
Selasa	26	26	15	15	52	52	16	16
Rabu	8	8	26	26	60	60	33	33
Kamis	41	33	38	2	46	46	56	25
Jumat	53	53	16	16	30	30		
Sabtu	21	21	41	41	8	8	41	41

Kelas	104							
Hari/Jam	1	2	3	4	5	6	7	8
Senin	53	53	27	27	41	33	8	8
Selasa	33	33	21	21	38	2	52	52
Rabu	60	60	41	41	16	16	49	49
Kamis	16	16	27	27	56	28	8	8
Jumat	30	30	46	46	2	2		
Sabtu	41	41	5	5	38	38	15	15

Gambar 5.2 Jadwal Mata Pelajaran yang Terbentuk dari Individu dengan Nilai Fitness Terbaik untuk parameter kedua

Pada gambar diatas, angka di kolom kelas menunjukkan nama kelas dan tingkat kelas, sedangkan angka-angka pada baris hari mewakili guru dan mata pelajaran.

Dari hasil pengujian validasi diatas, untuk membentuk jadwal mata pelajaran dengan menggunakan parameter pertama membutuhkan waktu sekitar 20 jam, sedangkan untuk parameter kedua membutuhkan waktu sekitar 3 jam.

Berikut ini adalah hasil pengujian dengan nilai parameter yang berbeda:

No	Jumlah Kelas	Nilai Parameter	h1 (%)	h2 (%)	h3 (%)	h4 (%)	h5 (%)
1	23	Jumlah generasi: 10 Jumlah populasi:10 Pc: 0,7 Ps:0,1 Pm: 0,001	14,5	1,89	5,29	1,7	4,73
2	23	Jumlah generasi: 10 Jumlah populasi:10 Pc: 0,6 Ps:0,1 Pm: 0,001	14,9	2,08	8,1	1,7	3,97
3	23	Jumlah generasi: 10 Jumlah populasi:10 Pc: 0,5 Ps:0,1 Pm: 0,001	15,7	2,64	10,29	2,1	5,41
4	4	Jumlah generasi: 30 Jumlah populasi:10 Pc: 0,7 Ps:0,1 Pm: 0,001	1,08	1,08	4,37	2,17	2,17
5	4	Jumlah generasi: 25 Jumlah populasi:10 Pc: 0,7 Ps:0,1 Pm: 0,001	4,37	2,17	7,61	1,08	1,08
6	4	Jumlah generasi: 15 Jumlah populasi:10 Pc: 0,7 Ps:0,1 Pm: 0,001	5.43	3.26	6.52	1.08	1.08

BAB VI
PENUTUP

6.1 Kesimpulan

Berdasarkan hasil pengujian dan analisis yang dilakukan terhadap kinerja sistem dapat diambil kesimpulan sebagai berikut:

1. Berdasarkan hasil pengujian sistem yang dilakukan dengan jumlah kelas sebanyak 23 kelas dan menggunakan nilai probabilitas *crossover* (Pc) yang berbeda (Pc = 0,5, Pc = 0,6, dan Pc = 0,7) sedangkan parameter yang lain bernilai sama, nilai *fitness* tertinggi diperoleh pada pengujian dengan nilai probabilitas *crossover* = 0,7, sedangkan dari hasil pengujian sistem yang dilakukan dengan jumlah kelas sebanyak 4 kelas dan menggunakan jumlah generasi yang berbeda (15 generasi, 25 generasi dan 30 generasi) sedangkan parameter yang lain bernilai sama, nilai *fitness* tertinggi diperoleh pada pengujian dengan jumlah generasi sebanyak 30 generasi.

2. Berdasarkan hasil pengujian unit dan intergrasi, sistem sudah berjalan dengan benar. Sedangkan berdasarkan pengujian validasi, untuk pengujian dengan jumlah kelas sebanyak 23 kelas, jumlah generasi sebanyak 10 generasi, jumlah populasi sebanyak 10 individu, probabilitas *crossover* sebesar 0,7, probabilitas seleksi mutasi sebesar 0,1, dan probabilitas mutasi sebesar 0,001 menghasilkan jadwal dengan bentrok mengajar sebanyak 77, terdapat jam mata pelajaran olahraga diatas jam keempat sebanyak 10, terdapat jam mata pelajaran matematika diatas jam keempat sebanyak 28, terdapat jam mata pelajaran fisika diatas jam keempat sebanyak 9, dan terdapat jumlah jam mengajar guru lebih dari 6 jam sehari sebanyak 25 , sedangkan pengujian dengan jumlah kelas sebanyak 4 kelas, jumlah generasi sebanyak 30 generasi, jumlah populasi sebanyak 10 individu, probabilitas crossover sebesar 0,7, probabilitas seleksi mutasi sebesar 0,1, dan probabilitas mutasi sebesar 0,001 menghasilkan jadwal dengan bentrok mengajar sebanyak 1, terdapat jam mata pelajaran olahraga diatas jam keempat sebanyak 1, terdapat jam mata pelajaran matematika diatas jam keempat sebanyak 4, terdapat jam mata pelajaran fisika diatas jam keempat sebanyak 2, dan terdapat jumlah jam mengajar guru lebih dari 6 jam sehari sebanyak 2.

3. Berdasarkan hasil pengujian yang telah dilakukan, terjadi kenaikan nilai fitness dalam setiap generasi (iterasi) yang berarti setiap kenaikan iterasi, jumlah pelanggaran yang terjadi semakin berkurang.

6.2 Saran

Berikut saran yang mungkin perlu dilakukan dalam pengembangan selanjutnya terhadap aplikasi penjadwalan mata pelajaran menggunakan algoritma genetika.

1. Menggunakan komputer dengan spesifikasi yang tinggi untuk mempercepat proses pembentukan jadwal.
2. Memanfaatkan teknik *multithreading* agar proses pembentukan jadwal berjalan lebih cepat.
3. Melakukan eksplorasi parameter algoritma genetika lebih mendalam untuk memperoleh hasil penjadwalan yang optimal.

DAFTAR PUSTAKA

Cahyono, Setiyo. 2006. ***Panduan Praktis Pemrograman Database menggunakan MySQL dan java.*** Bandung: Informatika.

Foenadioen, Prakoso Samuel. 2008. ***Pedoman Praktis Java Pengembangan Aplikasi Web Database Menggunakan Java Server Pages.*** Yogyakarta:Andi.

Hariyanto, B. 2007. ***Esensi-esensi Pemrograman Bahasa Java***. Bandung: Informatika.

Kadir, Abdul. 2004. ***Dasar Pemrograman Java 2 .*** Yogyakarta: Andi

Kusumadewi,Sri. 2003. ***Artificial Intelligence (Teknik dan Aplikasinya).*** Yogyakarta: Graha Ilmu.

Shalahuddin, M. & Rosa, A. S. 2010. ***Modul Pembelajaran Algoritma dan Pemrograman.*** Bandung: Modula.

SMAN 1 Srengat. 2009. ***Kurikulum SMA Negeri 1 Srengat: Tahun Pelajaran 2009/2010.*** Blitar: SMAN 1 Srengat.

Suyanto. 2007. ***Artificial Intelligence: Searching, Reasoning, Planning and Learning.*** Bandung: Penerbit Informatika.

Suyanto. 2008. ***Evolutionary Computation: Komputasi Berbasis "Evolusi" dan "Genetika".*** Bandung: Penerbit Informatika.