OPTIMASI PENJADWALAN MATA KULIAH DENGAN ALGORITMA GENETIKA (Studi Kasus di AMIK JTC Semarang)

Entot Suhartono
(etnadiabpd@gmail.com)
AMIK JTC Semarang

Abstrak

Course scheduling problems in a campus is a complicated thing. There are several methods for solving scheduling problems and produce an optimum schedule, such as genetic algorithms. In this research, problem solving course scheduling using genetic algorithms. This study aims to create genetic models for optimization scheduling optimization problem subjects. The case studies are taken in this study is the AMIK JTC Semarang with scheduling data on Odd Semester Academic Year 2014/2015. Then the schedule data is processed using a genetic algorithm optimization scheduling in order to obtain a condition where there is the best combination between subjects and leisure time of students, as well as the limited availability of space for all the existing courses. Based on trials that have been conducted, scheduling optimization model subjects with.

Key word : Scheduling Problem, Genetic Algorithm, Problem Solving, Optimization

1. PENDAHULUAN

Pengaturan waktu terhadap suatu kegiatan merupakan hal yang penting dilakukan agar kegiatan tersebut berlangsung secara lancar. Pengaturan waktu tersebut biasa disebut penjadwalan. Penyusunan jadwal kegiatan berkaitan dengan berbagai batasan/kendala yang harus dipenuhi sehingga memerlukan banyak pertimbangan untuk mendukung kegiatan tersebut. Ada berbagai masalah penjadwalan di dunia nyata seperti alokasi kejadian, kegiatan, orang, kendaraan, dll. Sebagian besar kasus penentuan jadwal yang bisa diterapkan sangat sulit dicapai karena sumber daya (waktu, tempat, orang, dll) terbatas. Jadi penentuan sebuah jadwal yang efisien menjadi masalah penting.

Masalah penjadwalan bahwa setiap perguruan tinggi selalu menghadapi masalah pada awal setiap semester akademik. Hal tersebut adalah masalah alokasi suatu kejadian (sesi pengajaran, ujian, sesi lab, dll) pada sejumlah kelas, ruang, dan dosen pada periode waktu yang telah ditentukan. Masalah penjadwalan dalam dunia pendidikan dapat dibagi menjadi dua kategori, yaitu penjadwalan kuliah dan ujian. Penyelesaian masalah penjadwalan perkuliahan dalam jumlah yang sangat besar hingga saat ini masih menjadi permasalahan yang rumit untuk diselesaikan secara manual (Wiga dkk, 2013). Perguruan tinggi harus memberikan jadwal yang nantinya masuk ke dalam waktu tertentu dimana setiap perkuliahan tidak benturan. Penjadwalan pada umumnya diperlukan untuk mengantisipasi adanya benturan jam kuliah dan juga waktu dosen dalam mengajar. Jadwal yang dihasilkan juga harus memenuhi batasan dan syarat yang bertujuan agar jadwal yang dihasilkan sesuai saat digunakan.

Masalah penjadwalan dalam perguruan tinggi merupakan persoalan khusus dari masalah optimasi yang ditemukan pada situasi nyata. Masalah ini membutuhkan waktu komputasi yang cukup tinggi untuk pencarian solusinya, terlebih lagi jika ukuran permasalahan semakin besar dengan bertambahnya jumlah komponen dan tetapan atau syarat yang ditentukan oleh institusi tempat jadwal tersebut digunakan (Uning, 2014). Selama proses, banyak aspek yang harus dipertimbangkan untuk memperoleh jadwal yang optimal, dan seringkali tidak dapat memuaskan karena tidak semua kebutuhan terpenuhi. Oleh karena itu perlu ditetapkan suatu batasan dalam penyusunan jadwal yang bersifat harus dipenuhi (*hard constraints*) dan tidak harus dipenuhi (*soft constraints*), tetapi tetap menjadi acuan dalam proses pembuatan jadwal (Mawaddah, 2006).

Persoalan optimasi (*optimization problem*) adalah persoalan yang menuntut pencarian solusi optimum (Marwana, 2012). Persoalan optimasi dibagi menjadi dua macam, yaitu maksimasi (*maximization*) dan minimasi (*minimization*). Ada dua metode dalam penyelesaian masalah

optimasi, yaitu (1) Metode Konvensional yang diterapkan dengan menggunakan perhitungan matematika murni atau secara biasa. Ada beberapa metode konvensional yang sering digunakan untuk menyelesaikan masalah optimasi, diantaranya: algoritma Djikstra, algoritma Floyd-Warshall, dan algoritma Bellman-Ford. (2) Metode Heuristik salah satu dari bidang kecerdasan buatan yang digunakan untuk menyelesaikan masalah optimasi. Terdapat beberapa algoritma dari metode heuristik yang sering digunakan dalam permasalahan optimasi, diantaranya adalah algoritma genetika, algoritma pencarian tabu, jaringan saraf tiruan, algoritma semut dan lain-lain.

Penelitian-penelitian terbaru menyarankan bahwa algoritma genetika merupakan metode yang layak dan efektif dalam mengatasi masalah penjadwalan (Teno dan Palgunadi, 2014). Algoritma genetika merupakan metode heuristik adaptif yang memiliki dasar pemikiran atau gagasan untuk proses seleksi alam dan genetika berdasarkan penelitian Charles Darwin. Algoritma genetika cukup baik untuk digunakan dalam penjadwalan mata kuliah di sebuah perguruan tinggi. Algoritma genetika merupakan salah satu jalan untuk memecahkan masalah yang cukup besar dengan solusi yang cukup baik meskipun masalah tersebut membutuhkan waktu eksekusi yang lama bila dilakukan secara manual (A. Jain, dkk, 2010).

Berdasarkan latar belakang di atas, maka penelitian mengusulkan bagaimana optimasi penjadwalan penggunaan ruang laboratorium komputer dengan algoritma genetika pada perguruan tinggi. Penelitian ini akan membangun aplikasi komputer dan melakukan uji coba otomatisasi masalah penjadwalan kuliah di perguruan tinggi dengan metode *meta-heuristic*, yaitu: Algoritma Genetika. Permasalahan yang ditangani pada penelitian ini adalah 1) pengelolaan komponen-konponen jadwal, seperti: data dosen, mahasiswa, ruangan, mata kuliah, dan waktu perkuliahan dan 2) pembuatan jadwal penggunaan ruang laboratorium dan mata kuliah praktikum komputer. Di harapkan dengan menggunakan algoritma ini akan dapat menyelesaikan masalah penjadwalan penggunaan ruang laboratorium komputer di perguruan tinggi, karena algoritma genetika dapat menyelesaikan permasalahan yang kompleksitasnya tinggi (Desiani dkk, 2006).

2. LANDASAN TEORI

2.1. Penjadwalan Mata Kuliah

Penjadwalan merupakan suatu kegiatan alokasi sumber daya dengan memiliki kendala (batasan) yang diberikan kepada suatu objek seperti di ruang-waktu, sedemikian rupa untuk memenuhi sedekat mungkin set tujuan yang diinginkan "(A. Wren, 1996). Definisi yang lebih umum adalah menugaskan satu set peristiwa (kuliah, kendaraan, acara-acara publik, dll) dengan set terbatas sumber dari waktu ke waktu sedemikian rupa untuk memenuhi kendala (batasan/*constraint*) yang telah ditetapkan. kendala ini dapat dikategorikan sebagai *hard constraint* dan *soft constraint* , di mana *hard constraint* memiliki prioritas yang lebih tinggi dari pada *soft constraint*. Terdapat dua batasan dalam penyusunan penjadwalan kuliah yang dikemukakan oleh Burke dkk (2002), yaitu : *hard constraint* (harus terpenuhi) dan *soft constraint* (diupayakan untuk terpenuhi). *Hard constraints* merupakan batas-batas yang harus diterapkan pada penjadwalan mata kuliah dan harus dipenuhi. Sebuah solusi hanya dapat dikatakan sahih dan valid apabila dalam solusi tersebut sama sekali tidak ada *hard constraint* yang terlanggar. *Hard constraints* yang umum dalam penjadwalan mata kuliah adalah sebagai berikut :

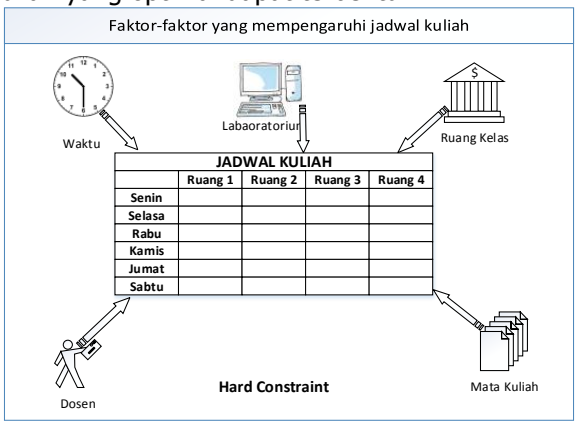
- Seorang dosen hanya dapat mengajarkan mata kuliah untuk satu lokasi pada waktu tertentu.
- Seorang mahasiswa hanya dapat mengikuti kuliah untuk satu lokasi pada waktu tertentu.
- Sebuah lokasi (ruangan/lab. komputer) hanya dapat digunakan untuk satu mata kuliah pada waktu tertentu.
- Mata kuliah dengan bobot 3 SKS dijadwalkan dengan satu kali pertemuan dalam seminggu.
- Hari aktif untuk perkuliahan adalah hari Senin sampai dengan Sabtu.

Berbeda dengan *hard constraint*, *soft constraint* merupakan kendala yang tidak selalu dapat terpenuhi dalam proses pembentukan jadwal, akan tetapi meskipun tidak harus terpenuhi, jadwal yang dihasilkan harus semaksimal mungkin berusaha memenuhi ketentuan *soft constraint*. Contoh *soft constraints* dalam penjadwalan mata kuliah antara lain :

- Dosen dapat meminta waktu mengajar tertentu yang diinginkan.

- Penempatan jadwal untuk waktu yang telah diminta dosen disesuaikan dengan prioritas dosen.

Berdasarkan Gambar 1.1, penjadwalan mata kuliah dipengaruhi oleh beberapa komponen yang terdiri atas: dosen, ruang kelas, mata kuliah, laboratorium dan waktu dengan sejumlah batasan-batasan tertentu, dimana batasan-batasan tersebut ada yang harus dipenuhi atau tidak boleh dilanggar. Batasan tersebut merupakan ukuran kualitas dari penjadwalan mata kuliah, sehingga suatu jadwal mata kuliah yang optimal dapat terbentuk.



Gambar 1.1. Faktor-faktor yang mempengaruhi penentuan jadwal kuliah
Sumber : (Mary, 2011)

2.2. Algoritma Genetika

2.2.1. Definisi Algoritma Genetika

Algoritma Genetika merupakan salah satu metode untuk menentukan optimalisasi atas dasar Teori Darwin. Langkah prosedur algoritma ini diawali dengan menentukan suatu set solusi potensial dan melakukan perubahan dengan beberapa perulangan (iterasi) dengan algoritma genetika untuk menghasilkan solusi terbaik. Set solusi potensial ini ditetapkan diawal dan disebut dengan kromosom. Kromosom ini dibentuk secara random berupa susunan angka binary yang di-generate dan dipilih. Keseluruhan set dari kromosom yang diobservasi mewakili suatu populasi (Heli, dkk, 2010). Kromosom-kromosom berevolusi beberapa kali tahapan iterasi yang disebut dengan generasi. Generasi baru (*offsprings*) di-*generate* dengan teknik kawin silang (*crossover*) dan mutasi (*mutation*). *Crossover* meliputi pemisahan atau pemecahan (*splitting*) dua kromosom, kemudian mengkombinasikan setengah bagian dari masing-masing kromosom dengan pasangan-pasangan lainnya. Mutasi meliputi pertukaran (*flipping*) satu bit (bagian) dari kromosom dengan satu bagian lain dari kromosom lain yang menjadi pasangannya. Kromosom-kromosom tersebut kemudian berevolusi dengan suatu kriteria kesesuaian (*fitness*) yang telah ditetapkan, hasil yang terbaik akan dipilih sedangkan yang lainnya diabaikan. Proses ini dilakukan secara berulang-ulang sampai menemukan suatu kromosom yang memiliki kesesuaian terbaik (*best fitness*) untuk dijadikan sebagai solusi terbaik dari suatu masalah.

2.2.3 Struktur Umum Algoritma Genetika

Pada ilustrasi Gambar1.2, struktur umum dari suatu algoritma genetika dapat didefinisikan dengan langkah-langkah sebagai berikut (Mery, 2011):

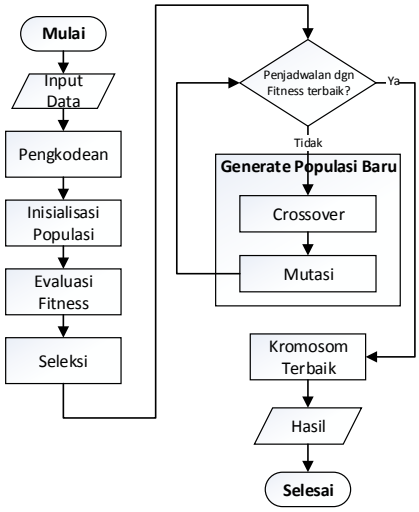
a. Menghasilkan atau membangkitkan populasi awal (*generate*)
   Populasi awal di-*generate* secara random, dimana populasi tersebut terdiri dari beberapa kromosom yang telah didefinisikan sehingga dapat dijadikan solusi awal. Populasi tersebut terdiri dari beberapa kromosom yang mewakili solusi yang diinginkan.
b. Membentuk generasi baru
   Membuat generasi baru menggunakan tiga operator yaitu operator reproduksi/seleksi, *crossover* dan mutasi. Tahapan tersebit dilakukan secara berulang-ulang sehingga diperoleh sejumlah kromosom yang cukup untuk menghasilkan generasi baru dimana generasi baru tersebut merupakan

representasi dari solusi baru. Generasi baru ini dikenal denga istilah anak (*offspring*).

c. Evaluasi solusi

Pada tiap generasi, kromosom akan diukur dengan fungsi fitness. Nilai fitness suatu kromosom menggambarkan kualitas kromosom dalam populasi tersebut. Proses ini akan mengevaluasi setiap populasi dengan menghitung nilai fitness setiap kromosom dan mengevaluasinya sampai terpenuhi kriteria berhenti. Apabila kriteria berhenti belum terpenuhi maka akan dibentuk lagi generasi baru dengan mengulangi langkah b. Beberapa kriteria berhenti yang sering digunakan antara lain:

- Berhenti pada generasi tertentu.
- Berhenti setelah beberapa generasi berturut-turut didapatkan nilai fitness tertinggi.
- Berhenti pada *n* generasi dimana nilai fitness dari populasi tidak mengalami perubahan.



Gambar 2. Flowchart Operasi Algoritma Genetika

2.2.4. Komponen-komponen Utama Algoritma Genetika

Ada 6 komponen utama yang terdapat di dalam algoritma genetika, (Mery, 2011) yaitu:

1. Teknik Pengkodean

Suatu teknik bagaimana mengkodekan gen dari kromosom. Teknik ini merupakan teknik untuk menyatakan populasi awal sebagai calon solusi suatu masalah ke dalam suatu kromosom sebagai suatu kunci pokok masalah. Teknik pengkodean meliputi pengkodean gen dan kromosom. Gen adalah bagian dari kromosom yang dapat direpresentasikan dalam bentuk string bit, tree, array bilangan real, daftar aturan, elemen permutasi, elemen program, atau representasi lain yang dapat diimplementasikan untuk operator genetika.

2. Prosedur Inisialisasi (generate populasi awal)

Suatu proses yang menghasilkan sejumlah individu secara acak (random). Banyaknya populasi tergantung pada masalah yang akan diselesaikan dan jenis operator genetika yang akan diterapkan. Setelah jumlah populasi ditentukan, selanjutnya dilakukan inisialisasi terhadap kromosom yang ada di dalam populasi tersebut. Inisialisasi kromosom dilakukan secara acak, dengan tetap memperhatikan domain solusi dan kendala permasalahan yang ada.

3. Fungsi Evaluasi

Individu dievaluasi berdasarkan fungsi tertentu sebagai ukuran kinerjanya. Individu dengan nilai fitness tinggi pada kromosomnya yang akan dipertahankan, sedangkan individu yang pada kromosomnya bernilai fitness rendah akan diganti. Fungsi fitness tergantung pada permasalahan tertentu dari representasi yang digunakan.

Rumus fitness yang digunakan, (Lee dkk, 2001) adalah sebagai berikut:

$$Fitness = \frac{1}{1+(F_1B_1+F_2B_2+\cdots)} \quad (1)$$

Keterangan : Bn = Bobot pelanggaran
Fn = Banyaknya pelanggaran
n = 1 . . . n

4. Seleksi

Proses seleksi bertujuan untuk memilih individu-individu yang akan dipilih untuk proses persilangan dan mutasi, sehingga akan diperoleh calon induk yang baik. Induk yang baik akan menghasilkan keturunan yang baik. Langkah pertama dalam seleksi yaitu pencarian nilai fitness. Masing-masing individu dalam suatu wadah seleksi akan menerima probabilitas reproduksi yang tergantung pada nilai objektif dirinya sendiri terhadap nilai objektif dari semua individu dalam wadah seleksi tersebut. Nilai fitness inilah yang nantinya akan digunakan pada tahap seleksi berikutnya.
Ada beberapa metode untuk memilih kromosom yang digunakan antara lain :

a. Roulette wheel selection

Seleksi ini dilakukan dengan cara menyeleksi parent dengan tujuan untuk mempertahankan nilai fitness-nya agar memiliki kesempatan untuk diseleksi adalah kromoson yang baik. Proses ini diibaratkan seperti permainan roda rolet (roullette wheel), di mana semua kromosom ditempatkan dalam populasi, setiap tempat besar sesuai dengan fungsi fitness. Kromosom dipilih berdasarkan nilai fitness, semakin besar nilai fitness maka kromosom tersebut mempunyai peluang untuk dipilih beberapa kali. Metode seleksi roda rolet merupakan metode yang paling sederhana, dan sering juga dikenal dengan nama stochastic sampling with replacement.

b. Selekssi good fitness

Seleksi ini dilakukan dengan cara setengah dari jumlah populasi yang memiliki nilai fitness yang paling rendah akan dihilangkan, sehingga selalu hanya tersisa sekelompok solusi yang terbaik. Solusi yang tersisa hasil dari seleksi tersebut disebut populasi induk. Karena jumlah populasi harus tetap, maka perlu di-generate solusi baru sebanyak setengah dari jumlah populasi yang ada. Ada 2 cara yang digunakan untuk men-generate solusi baru, yaitu dengan cara reproduksi kromosom baru dan mutasi dari solusi induk. Tujuan men-generate solusi baru ini adalah untuk menemukan alternatif solusi yang lebih baik dari solusi-solusi yang sudah dihasilkan.

5. Operator Genetika

Algoritma genetika merupakan proses pencarian yang heuristik dan acak sehingga penekanan pemilihan operator yang digunakan sangat menentukan keberhasilan algoritma genetik dalam menemukan solusi optimum suatu masalah yang diberikan. Hal yang harus diperhatikan adalah menghindari terjadinya konvergensi premature, yaitu mencapai solusi optimum yang belum waktunya, dalam arti bahwa solusi yang diperoleh adalah hasil optimum lokal. Ada dua operator genetika yaitu:

a. Crossover (Persilangan)

Crossover merupakan proses di dalam algoritma genetika yang bekerja untuk meng-gabungkan dua kromosom parent menjadi kromosom baru (*offspring*) pada suatu waktu. Sebuah kromosom yang mengarah pada solusi baik dapat diperoleh melalui proses crossover pada dua buah kromosom. Cara sederhana pada proses crossover yaitu dengan memilih satu titik yang dipisahkan secara acak dan kemudian membentuk *offspring* dengan mengkombinasikan segmen dari satu induk ke sebelah kiri dari titik yang dipisahkan dengan segmen dari induk yang lain ke sebelah kanan dari titik yang dipisahkan.

b. Mutation (Mutasi)

Proses mmerupakan untuk mengubah salah satu atau lebih beberapa gen dari suatu kromosom. Proses ini berfungsi untuk menggantikan gen yang hilang dari populasi akibat proses seleksi yang memungkinkan munculnya kembali gen yang tidak muncul pada inisialisasi populasi. Beberapa cara operasi mutasi yang diterapkan dalam algoritma genetika menurut jenis pengkodeannya, antara lain:

6. Parameter Kontrol

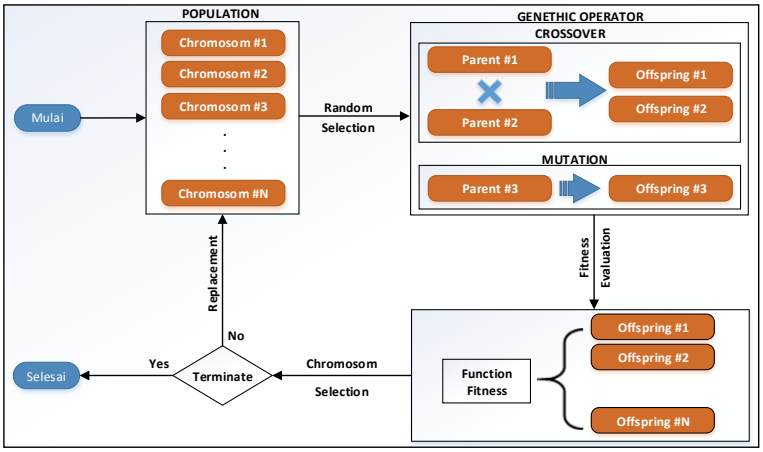
Parameter kontrol genetika diperlukan untuk mengendalikan operator-operator seleksi. Pemilihan parameter genetika menentukan penampilan kinerja algoritma genetika dalam memecahkan masalah (Desiani dkk, 2006). Ada dua parameter dasar dari algoritma genetika, yaitu probabilitas crossover (Pc) dan probabilitas mutasi (Pm).

a. Probabilitas Crossover (Pc)

Probabilitas crossover akan mengendalikan operator crossover dalam setiap generasi dalam populasi yang mengalami crossover. Semakin besar nilai probabilitas crossover, akan semakin cepat struktur individu baru terbentuk ke dalam populasi. Apabila nilai probabilitas crossover terlalu besar, maka individu yang merupakan kandidat solusi terbaik mungkin akan dapat hilang lebih cepat pada generasi selanjutnya. Nilai probabilitas crossover yang disarankan adalah berkisar antara 80 % - 95 %.

b. Probabilitas mutasi (Pm)

Probabilitas mutasi akan mengendalikan operator mutasi pada setiap generasi dengan peluang mutasi yang digunakan lebih kecil daripada peluang crossover. Pada seleksi alam murni, mutasi jarang sekali muncul, sehingga operator mutasi pada algoritma genetik tidak selalu terjadi. Nilai probabilitas mutasi yang disarankan kecil antara 0.5% - 1%.



Gambar 3. Komponen-kompenen Algoritma Genetika
Sumber : Merry (2011)

3. PERANCANGAN DAN IMPLEMENTASI OPTIMASI PENJADWALAN

Hal pertama yang kita harus dipertimbangkan saat kita berurusan dengan algoritma genetika adalah bagaimana untuk merepresentasikan solusi sedemikian rupa bahwa solusi tersebut layak untuk operasi genetik seperti *crossover* dan mutasi, bagaimana menentukan seberapa baik solusi bisa digunakan. Untuk itu kita harus dapat menghitung nilai *fitness* dari solusi tersebut.

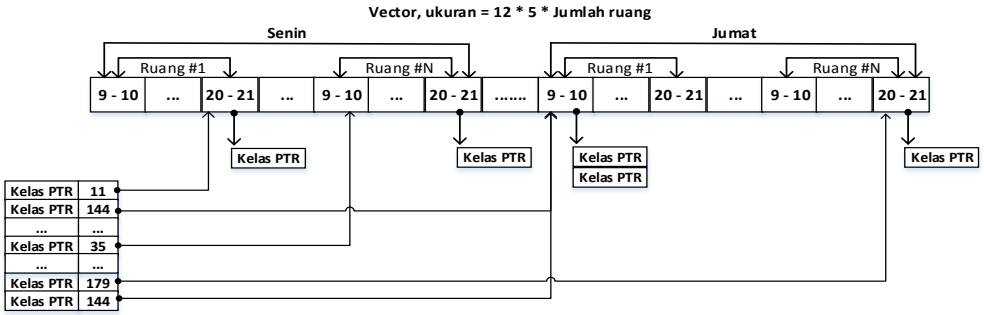
3.1. Representasi Kromosom

Pembuatan representasi kromosom untuk penjadwalan perkuliahan, diperlukan slot waktu jam perkuliahan. Pada kasus disini digunakan istilah waktu kuliah adalah satu sesi perkuliahan pada setiap ruang kuliah untuk setiap harinya. Diasumsikan bahwa kegiatan

perkuliahan diselenggarakan dimulai jam 9 pagi sampai dengan jam 9 malam (total 12 jam), dan hari kerja adalah Senin sampai Jumat (total 5 hari). Ilustrasi representasi kromosom dapat dilihat pada Gambar 4.

Representasi kromosom pada penelitian ini menggunakan std::vector dengan ukuran 12*5*number_of_rooms. Slot waktu yang digunakan adalah std::list karena selama ekseskusi algoritma, dimungkinkan beberapa perkuliahan dilaksanakan pada slot waktu yang sama. Ada peta *hash* tambahan yang digunakan untuk mendapatkan slot waktu pertama di mana kelas dimulai (posisinya di vektor) dari objek kelas. Setiap jam kelas memiliki entri terpisah dalam vektor, tapi hanya ada satu entri per kelas di peta *hash*. Misalnya, jika kelas dimulai pada 01:00 dan berlangsung selama tiga jam, ia memiliki entri dalam slot 01:00, 02:00, dan 03:00.

Penentuan populasi awal adalah proses membangkitkan sejumlah kromosom secara acak (*random*). Kromosom menyatakan salah satu alternatif solusi yang dimungkinkan. Kromosom dapat dikatakan sama dengan individu. Ukuran populasi tergantung pada masalah yang akan diselesaikan. Setelah ukuran populasi ditentukan, langkah selanjutnya adalah membangkitkan populasi awal dengan cara melakukan inisialisasi solusi yang dimungkinkan kedalam sejumlah kromosom. Panjang satu kromosom ditentukan berdasarkan permasalahan yang diteliti.



Gambar 4. Representasi Kromosom Optimasi Jadwal
Sumber : Teno, dkk (2014)

Pseudo Code Kromosom

```
function Fitness (Kromosom[i]) → integer
{menghitung nilai fitness dari masing-masing kromosom}
Deklarasi
    Jum : integer
    j : integer
    Kromosom[][] : array of integer of integer
Algoritma
Jum ← (A,Kromosom[i][1])
for ← 2 to 4 do
  Jum ← Jum + Ruang(Kromosom[i][j-1],Kromosom[i][j])
endfor
Jum ← sum + Ruang (Kromosom[i][4],A)
→ Jum
```

Selanjutnya, komrosom menampung nilai-nilai dan parameter fitnes-nya yang akan digunakan dalam operasi genetik. Nilai fitness ditampung di sini melalui Pseudo Code adalah sebagai berikut:

3.2. Penentuan Fungsi Fitness Penjadwalan

Suatu individu dievaluasi berdasarkan suatu fungsi tertentu sebagai ukuran performansinya. Didalam evolusi alam, individu yang bernilai fitness tinggi yang akan bertahan hidup. Sedangkan individu yang bernilai fitness rendah akan mati. Fungsi yang digunakan untuk mengukur nilai kecocokan atau derajat optimalitas suatu kromosom disebut dengan *fitness function* (Sam'ani, 2012). Nilai yang dihasilkan dari fungsi tersebut menandakan seberapa optimal solusi yang diperoleh. Nilai yang dihasilkan oleh fungsi fitness merepresentasikan seberapa banyak jumlah persyaratan yang dilanggar, sehingga dalam kasus penjadwalan perkuliahan semakin kecil jumlah pelanggaran yang dihasilkan maka solusi yang dihasilkan akan semakin baik. Untuk setiap pelanggaran yang terjadi akan diberikan nilai 1. Agar tidak terjadi nilai fitness yang tak terhingga maka jumlah total semua pelanggaran akan ditambahkan 1.

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

Keterangan :

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

Beberapa batasan yang digunakan dalam penyusunan penjadwalan ini adalah :

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

Dari contoh yang ada akan menghasilkan nilai fitness sebagai berikut :

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

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

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

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

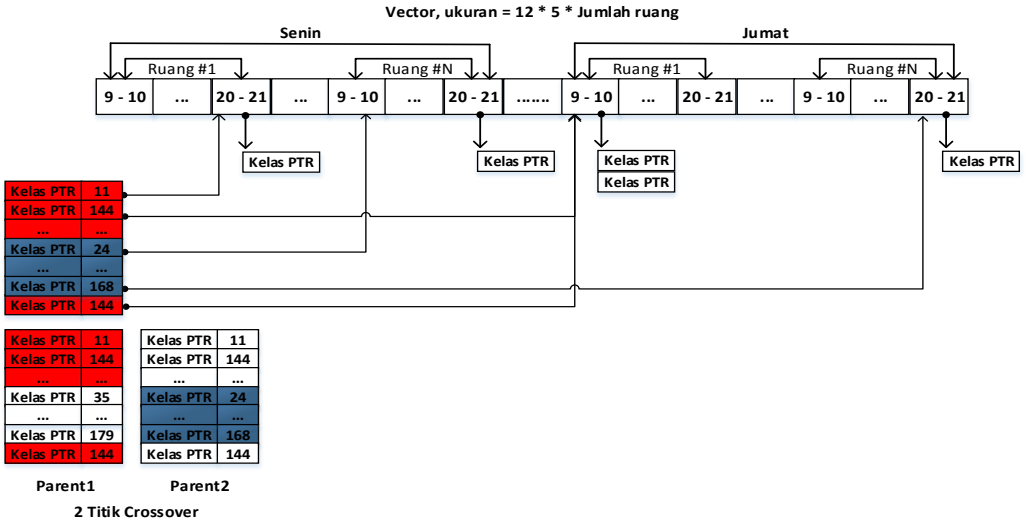
Penetapan nilai fitness kromosom hanya persyaratan keras yang digunakan untuk menghitung fitness jadwal kelas. Adapun cara penetapan nilai fitness kromosom adalah sbb:

- Setiap kelas dapat memiliki poin 0 sampai 5.
- Jika kelas menggunakan ruang kelas kosong, maka nilai akan dinaikan
- Jika kelas membutuhkan komputer dan menempatkannya ke dalam ruang kelas secara bersamaan atau tidak, maka nilai skor kelas akan bertambah.
- Jika kelas ditempatkan dalam ruang kelas yang tidak mencukupi kapasitasnya, maka nilai skor akan bertambah.
- Jika dosen tidak memiliki kelas-kelas lain pada waktu yang sama, maka nilai skor kelas akan dinaikan lagi.
- Jika terdapat beberapa mahasiswa yang menghadiri kelas memiliki kelas lain pada waktu yang sama, nilai skor kelas akan turun, jika tidak nilai skor kelas akan bertambah.
- Jika kelas melanggar aturan setiap slot waktu kosong yang ditempatinya, maka nilai skor untuk aturan itu tidak bertambah.
- Total skor jadwal kelas adalah jumlah poin dari semua kelas.
- Nilai fitness dihitung schedule_score/maximum_score, dan maximum_score adalah number_of_classes*5.

Nilai-nilai fitness yang diwakili oleh *floating point* nomor presisi tunggal (float) di antara 0 sampai 1.

3.3. *Crossover* (Kawin Silang/ Rekombinasi) Kromosom

Suatu operasi crossover yang menggabungkan data dalam peta hash dari dua orang tua, dan kemudian menciptakan vektor slot sesuai dengan isi peta hash baru. Operasi crossover kemudian akan memisahakan peta dari kedua orang tua menjadi dua bagian secara acak. Jumlah bagian didefinisikan dengan jumlah titik crossover (ditambah satu) parameter kromosom. Kemudian, secara bergantian salinan bagian tersebut membentuk orang tua untuk kromosom baru, dan membentuk vektor slot baru. Dengan kata lain, crossover merupakan suatu proses pertukaran nilai gen pada posisi gen yang sama dari kedua orang tua (induk). Penukaran gen tersebut juga harus dilakukan pengecekan apakah individu baru yang terbentuk sesuai dengan aturan yang berlaku. Operasi ini pada penjadwalan penggunaan ruang kelas dan laboratorium dapat di lihat pada ilustrasi berikut ini.



Gambar 5. Representasi Operasi *Crossover*

```
// Pseudo Code Operasi Crossover
procedure Crossover (input populasi: integer, pc: real)
{melakukan pemilihan induk pada proses cross
over}
Deklarasi
    k : integer
    R[] : array of integer
function random (input a-b : integer) → integer
{menghasilkan bilangan random bilangan a hingga b}
 Algoritma
 k= 0
 While k <= populasi do
    R[k] ← random(0-1)
    if R[k] < pc then
        pilih Kromosom[k][] sebagai induk
    endif
    k ←  k+1
 endwhile
```

3.4. Mutasi Kromosom

Proses mutasi ini adalah suatu proses eksploitasi terhadap kemungkinan-kemungkinan modifikasi pada jadwal yang telah ada. Perubahan posisi beberapa mata kuliah ini (mutasi) dapat membuat solusi duplikasi ini menjadi memiliki nilai fitness yang lebih rendah maupun lebih tinggi. Mutasi dapat dilakukan dengan dua cara, yaitu cara random dan cara swap atau penukaran. Mutasi cara pertama adalah dengan menentukan dua gen yang akan dimutasi. Setelah itu nilai kedua gen tersebut dirandom ulang untuk mendapatkan nilai yang baru. Pada cara kedua adalah dengan menukar langsung nilai dari gen. Pemilihan cara mutasi dilakukan secara random.

```
// Pseudo Code Mutasi
function JumlahMutasi(input JumGen,
JumlahKromosom: integer, pm: real) → integer
{menghitung jumlah proses mutasi}
Deklarasi
    TotalGen : integer
    JumMutasi : integer
Algoritma
TotalGen ← JumGen * JumlahKromosom
pm ←  0.2
JumMutasi ← 0.2*TotalGen
→ JumMutasi
```

3.5. Pembagian Ruang

Pembagian ruangan dilakukan setelah kelas mata kuliah yang diujikan menempati slot waktu yang tersedia. Pada tahap ini dilakukan pembagian alokasi ruangan sekaligus dilakukan pengecekan jumlah peserta mata kuliah. Apabila jumlah peserta sesuai dengan kapasitas, maka pengecekan selanjutnya adalah penggunaan ruangan tersebut. Ruangan yang sudah digunakan kelas mata kuliah yang lain pada waktu dan jam sama akan mempunyai nilai 100 dan jika belum nilainya 0.

3.6. Kondisi Selesai

Terdapat tiga kondisi selesai yang dapat menghentikanproses algoritma pemrograman ini, yaitu:

a. Jika setelah beberapa generasi berturut-turut nilaifitnessterbaik dari populasi tidak mengalami perubahan kembali
b. Jika jumlah generasi atau iterasi maksimum telah tercapai.
c. Jika nilai fitness terbaik minimal telah tercapai. Jika salah satu kondisi di atas telah diperoleh makaiterasi akan dihentikan dan jika salah satu kondisi selesai ini belum tercapai maka program akan mengulang kembali proses ini / iterasi dari langkah keempat yaitu evaluasi fitness terhadap populasi baru tadi.

4. HASIL PENELITIAN DAN PEMBAHASAN

4.1. Perangkat Pendukung Penelitian

Pada penelitian ini terdapat 2 (tiga) perangkat yang dibutuhkan untuk mendukung penelitian yaitu perangkat keras minimal *(hardware)* yang digunakan, perangkat lunak *(software)* yang digunakan untuk mengimplementasikan perancangan serta data yang diolah oleh sistem.
Perangkat keras (*hardware*) yang digunakan dalam penelitian ini adalah seperangkat komputer (laptop) dengan spesifikasi :

a. Processor Intel CORE I5 2.5 GHz
b. RAM 4 GB
c. Harddisk 500 GB

Perangkat lunak (*software*) yang digunakan dalam penelitian ini adalah:

a. Sistem operasi menggunakan Windows Seven (7).
b. Microsoft Visual C++ sebagai perangkat lunak pembuatan program aplikasinya
c. Notepad ++ sebagai teks editor untuk membuat file konfigurasi penjadwalan

Data yang digunakan penelitian ini adalah jadwal perkuliahan pada AMIK JTC Semarang semester Gasal tahun akademik 2015-2016.

4.2. Implementasi Aplikasi Optimasi Penjadwalan

4.2.1. Persiapan Data Optimasi Penjadwalan

Penelitian ini menggunakan data dosen, mata kuliah, kelompok kuliah, kelas, dan ruang kelas pada AMIK JTC Semarang tahun akademik 2014/2015 semester ganjil. Penelitian ini diasumsikan bahwa hari kuliah adalah mulai hari Senin sampai hari Jumat. Sedangkan waktu kuliah yang dapat diselanggarakan adalah mulai pukul 09.00 sampai dengan pukul 21.00. Adapun data dosen, mata kuliah, daftar ruang, dan kelas adalah sebagai berikut:

a. Daftar Dosen yang mengampu pada AMIK JTC Semarang Semster Ganjil tahun akademik 2014/2015 adalah sebanyak 28 orang.

Daftar Dosen		
1. Dr. Alex Sujanto,SE, SPd., MM	11. Dr. Haryono, MHum	21. Dhani Hirnawan, SS
2. Wahjono, SE, MSi	12. Drs. Bambang Hartono, MHum	22. Anita Sulistyawati, SSi
3. Saiful Umam, SAg.	13. Rusito, SKom, MKom	23. Eko Adi, SPd, MPd
4. Sugeng Murdowo, MKom	14. Budi Aprianto, MKom	24. Wisnu Widi, SSi
5. Mochamad Zainuri, SE, MSi	15. Umi Kulsum, SE	25. Riesky Ferdian, SKom
6. Eko Nurhidayat SSi, MKom	16. Ira Setyawati, SE, MSi	26. Subianto, SKom
7. Muhamad Danuri, MKom	17. Sumardi, SKom, MKom	27. Febrian Wahyu, MCs
8. Entot Suhartono, MKom	18. Kasdur Witarto, SE, MM	28. Arini Novalinda, SE, MSi
9. Kristiawan Nugroho, MKom	19. Prind Tri Ajeng, MKom	29. Andi Natalistyo, SE, MSi
10. Heru Sulistyo, SE, MSi, Akt	20. Ninon Saptarelan, SPd	

Sumber: Data Dosen, BAAK AMIK JTC Semarang 2015

b. Daftar Mata Kuliah yang ditawarkan pada Semster Ganjil tahun akademik 2014/2015

Daftar Mata Kuliah		
1. Sistem Informasi Manajemen	12. Bahasa Indonesia	23. Desain Grafis
2. Manajemen Umum	13. Aplikasi Pemrograman	24. Teknik Riset Operasi
3. Pendidikan Agama	14. Metode Observasi dan Laporan	25. Pemrograman WEB
4. Algoritma dan Pemrograman	15. Pemrograman Berbasis Visual I	26. Manajemen Keuangan
5. Praktikum Akuntansi	16. Pemrograman Berbasis Visual II	27. Komputer Akuntansi
6. Analisa dan Perancangan Sistem	17. Pemograman Berbasis Objek I	28. Rekayasa WEB
7. E-Bisnis	18. Sistem Operasi	
8. Akuntansi Biaya	19. Akuntansi Dasar	
9. PPKN	20. Pengantar Bisnis	
10. Aljabar Linier	21. Pengantar Teknologi Informasi	
11. Matematika Ekonomi	22. Bahasa Inggris II	

Sumber: Data Dosen, BAAK AMIK JTC Semarang 2015

c. Daftar Ruang yang ada di AMIK JTC Semarang

Daftar Ruang dan Kapasitas Kelas/Laboratorium		
1. Ruang = Riyadi 1 Kapasitas = 35	6. Ruang = Ruang 4 Kapasitas = 35	10. Ruang= Lab. 1 Kapasitas = 28
2. Ruang = Riyadi 2 Kapasitas = 50	7. Ruang= Ruang 6 Kapasitas = 35	11. Ruang = Lab. 2 Kapasitas = 28
3. Ruang = Gasebo Kapasitas = 35	8. Ruang = Lab. Bahasa Kapasitas = 15	12. Ruang = Lab. 3 Kapasitas = 28
5. Ruang = Ruang 3 Kapasitas = 45	9. Ruang = Ruang 6 Kapasitas = 35	

Sumber: Data Dosen, BAAK AMIK JTC Semarang 2015

4.2.2. Penyusunan File Konfigurasi Penjadwalan Algoritma Genetika

Aplikasi Optimasi Penjadwalan ini membutuhkan file konfigurasi yang berisi data dosen, mata kuliah, ruangan kelas/lab, dan kelompok kuliah. File konfigurasi ini merupakan file berbasis teks, sehingga dapat dibuat dengan menggunakan aplikasi teks editor. File konfigurasi tersebut nantinya akan diproses oleh aplikasi penjadwalan dengan algoritma genetika untuk menghasilkan penjadwalan perkuliahan secara optimal. Adapun data yang harus diisikan ke dalam file konfigurasi tersebut adalah terdiri dari objek data :

- professor (#prof tag) – berisi data dosen.
- course (#course tag) – berisi data mata kuliah yang ditawarkan.
- room (#room tag) – berisi data ruang kelas atau laboratorium.
- group (#group tag) – berisi data kelompok kuliah.

Setiap objek data diawali dengan tag # dan diakhiri dengan tag #end, semua tag harus dalam baris terpisah. Dalam tubuh objek data, setiap barisnya berisi satu kunci dan nilai pasangan (*atribut*) yang dipisahkan oleh tanda "=". Setiap atribut harus ditentukan hanya satu kali, kecuali untuk atribut kelompok kuliah dalam objek data #group dapat memiliki beberapa atribut kelompok. Tag dan nama kunci adalah case sensitif.

Berikut struktur atau sintak penulisan atribut data objek pada file konfigurasi:

#prof
id (tipe number) – ID dosen.
name (tipe string) – nama dosen.

#course
id (tipe number) – ID mata kuliah.
name (tipe string) – nama mata kuliah.

#room
name (tipe string) – nama ruang kelas/ lab.
size (tipe number) – jumlah kursi dalam kelas (kapasitas ruang kelas).
lab (tipe boolean, sifat optional) – indikasi jika ruangan adalah ruang laboratorium;
Jika tidak diisi maka menggunakan nilai default, yaitu false.

#group
id (tipe number) - ID kelompok kelas mahasiswa.
name (tipr string) – nama kelompok mahasiswa.
size (tipr number) – jumlah mahasiswa dalam kelompok.

#class
professor (tipe number) - ID dosen; untuk menghubungkan dosen dengan kelas.
course (tipe number) - ID mata kuliah; untuk menghubungkan mata kuliah dengan kelas.
group (tipe number) - ID kelompok kelas mahasiswa untuk dihubungkan dengan kelas;
setiap kelas dapat digunakan oleh beberapa kelompok kelas mahasiswa.
duration (tipe number, sifat optional) – durasi mata kuliah (sesi jam); jika tidak diisi, maka default-nya adalah 1 sesi jam perkuliahan.
lab (tipe boolean, sifat optional) – jika kelas membutuhkan praktek; Jika tidak diisi, maka nilai default adalah false.

4.2.3. *Parsing* File Konfigurasi Penjadwalan Algoritma Genetika

Parsing atau penguraian file konfigurasi dilakukan oleh class Configuration. Method ParseFile akan membuka dan menguraikan file konfigurasi. Method ini akan mencari tag-tag objek (#prof; #class, dan lainnya) dan memanggil method yang berkaitan dengan pemrosesan objek. Methode ParseFile juga akan membersihkan atau mengosongkan objek yang telah diproses sebelumnya. Berikut ini adalah class parsing file konfigurasi penjadwalan dengan algoritma genetika.

Class Parsing file dan Tag Objek

```
// Class parsing file
Configuration::GetInstance().ParseFile( "GaSchedule.cfg" );
// Class parsing taf Objek
public:
   void ParseFile(char* fileName);
private:
   Professor* ParseProfessor(ifstream& file);
   StudentsGroup* ParseStudentsGroup(ifstream& file);
   Course* ParseCourse(ifstream& file);
   Room* ParseRoom(ifstream& file);
   CourseClass* ParseCourseClass(ifstream& file);
```

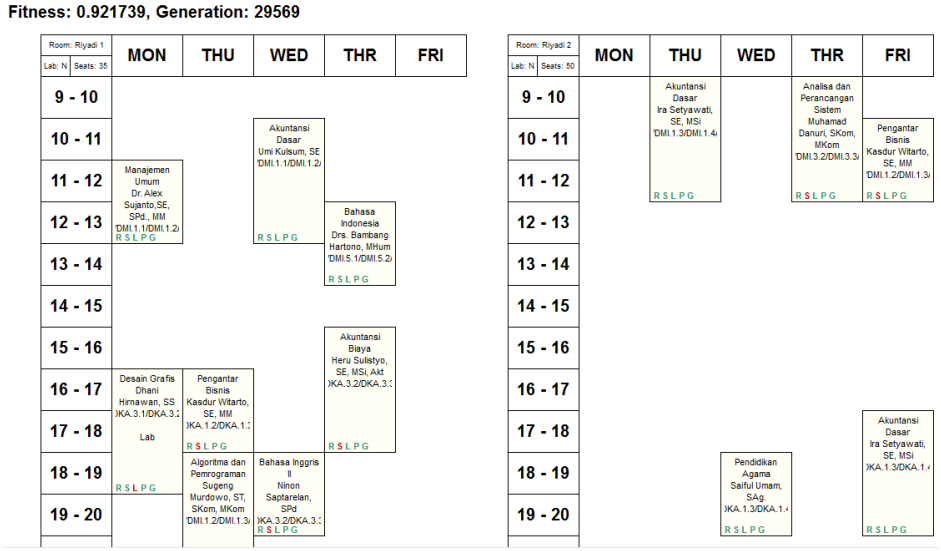
Objek-objek yang telah di-*parsing* (diproses) disimpan ke dalam *hash map*, kecuali class-class course agar class-class tersebut dapat diakses dengan mudah dan cepat.

Proses Penampungan hasil parsing pada objek
private: hash_map<int, Professor*> _professors; hash_map<int, StudentsGroup*> _studentGroups; hash_map<int, Course*> _courses; hash_map<int, Room*> _rooms; list<CourseClass*> _courseClasses;

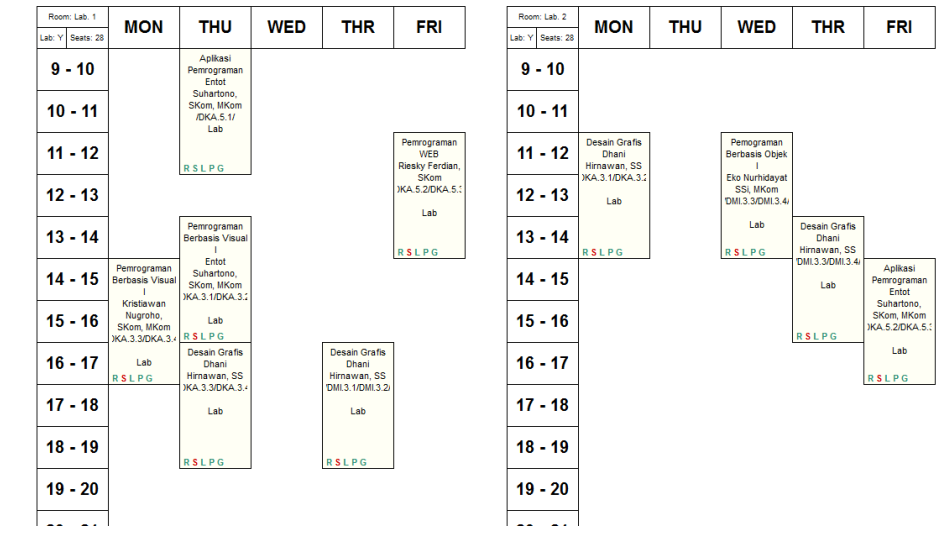
Class Configuration juga berisi method yang dapat memanggil kembali objek atau informasi yang telah di-*parsing* (diuraikan atau diproses).

4.2.4. Hasil Optimasi Penjadwalan Algoritma Genetika

Berdasarkan file konfigurasi yang telah disusun dan berdasarkan data yang ada, kemudian file konfigurasi diproses dengan metode Algoritma Genetika, maka diperoleh jadwal perkuliahan seperti pada Gambar 6 dan Gambar 7.



Gambar 6. Hasil penjadwalan dengan algoritma genetika pada ruang kelas



Gambar 7. Hasil penjadwalan dengan algoritma genetika pada ruang laboratorium

5. KESIMPULAN

Dengan bantuan Algoritma Genetika penyusunan penjadwalan mata kuliah dapat dioptimalkan. Program dapat mencari solusi penjadwalan pada waktu yang dapat digunakan baik oleh mahasiswa dan ruangan yang terlibat dalam suatu mata kuliah. Dengan ini kita dapat menghasilkan sebuah solusi untuk mahasiswa yang mempunyai waktu terbatas dan memanfaatkan jumlah ruangan yang terbatas. Dengan menggunakan metode *best fitness*, maka Algoritma Genetika akan selalu menunjukkan kenaikan *fitness* atau dengan kata lain generasi selanjutnya lebih baik atau minimal sama dengan generasi sebelumnya.

DAFTAR PUSTAKA

Anita Desiani & Muhammad Arhami, 2006, *Konsep Kecerdasan Buatan*. Andi Offset. Yogyakarta.

A. Jain, D.S. Jain, dan D.P. Chande, 2010, "Formulation of Genetic Algorithm to Generate Good Quality Course Timetable". *International Journal of Innovation, Management and Technology*).

A. Wren. (1996). Scheduling, timetabling and rostering - A special relationship? In: E.K. Burke and P. Ross (eds). (1996). *Practice and Theory of Automated Timetabling: Selected Papers from the 1st International Conference*. LNCS 1153. Springer-Verlag, Berlin, Heidelberg.

Burke and Sanja Petrovic, 2002, Burke Recent research directions in automated timetabling, *European Journal of Operational Research 140* .

Heli, Y. Shanshan, dan Lijia, 2010, "*The Application of Genetich Algorithm Based on Multi-dimension Code Scheme on Course Scheduling In Adult Education*". Proceedings of the Third Inrenational Syposium on Electronic Commerce and Security Workshop (ISECS'10).

Lee Spector et al, editor, GECCO 2001: *Proceedings of the Genetic and Eovlutionary Computation Conference*, pages 211–218, San Francisco, CA, 2001. Morgan Kaufmann.

Marwana, 2012, *Optimasi Penjadwalan Mata Kuliah Menggunakan Algoritmat Genetika Berbasis Permintaan Mahasiswa*, Prosiding Konferensi Nasional Ilmu Komputer 2012.

Mawaddah, NK & Mahmudy, WF 2006, Optimasi penjadwalan ujian menggunakan algoritma genetika, *Jurnal Ilmu Komputer Kursor, vol. 2, no. 2*.

Mery Hanita, 2011, *Penerapan Algoritma Genetika pada Penjadwalan Mata Kuliah (Studi Kasus: Program Studi Matematika FMIPA Universitas Bengkulu)*, Non Publikasi, Universitas Bengkulu.

Sam'ani, 2012, *Rancang Bangun Sistem Penjadwalan Perkuliahan dan Ujian Akhir Semester Dengan Algoritma Genetika*, Non Publikasi, Magister Sistem Informasi, Undip, Semarang.

Teno Siswono dan Sarngadi Palgunadi, 2014, *Analisa Kombinasi Algoritma Genetika Dengan Algoritma Palgunadi Untuk Penjadwalan Mata Kuliah Di Universitas Sebelas Maret*, Prosiding SNST ke-5, 2014.

Uning L., Naniek, dan Desti, 2014, *Implementasi Algoritma Genetika pada Penjadwalan Perkuliahan*, Prosiding Seminar Nasional Aplikasi Sains & Teknologi (SNAST) 2014. Yogyakarta

Wiga Ayu P., Arif Junaidi, dan Retno Aulia V., 2013, Penjadwalan Mata Kuliah Menggunakan goritma Genetika di Jurusan Sistem Informasi ITS, *Jurnal Teknik POMITS Vol 2. No. 1, 2013*.

Lampiran: Class Genetic Algorithm

```
class Algorithm
{private:
    // Population of chromosomes
    vector<Schedule*> _chromosomes;
    // Inidicates wheahter chromosome belongs to best chromosome group
    vector<bool> _bestFlags;
    // Indices of best chromosomes
    vector<int> _bestChromosomes;
    // Number of best chromosomes currently saved in best chromosome group
    int _currentBestSize;
```

```
    // Number of chromosomes which are replaced in each generation by offspring
    int _replaceByGeneration;
    // Pointer to algorithm observer
    ScheduleObserver* _observer;
    // Prototype of chromosomes in population
    Schedule* _prototype;
    // Current generation
    int _currentGeneration;
    // State of execution of algorithm
    AlgorithmState _state;
    // Synchronization of algorithm's state
    CCriticalSection _stateSect;
    // Pointer to global instance of algorithm
    static Algorithm* _instance;
    // Synchronization of creation and destruction of global instance
    static CCriticalSection _instanceSect;
public:
    // Returns reference to global instance of algorithm
    static Algorithm& GetInstance();
    // Frees memory used by gloval instance
    static void FreeInstance();
    // Initializes genetic algorithm
    Algorithm(int numberOfChromosomes, int replaceByGeneration, int trackBest,
            Schedule* prototype, ScheduleObserver* observer);
    // Frees used resources
    ~Algorithm();
    // Starts and executes algorithm
    void Start();
    // Stops execution of algoruthm
    void Stop();
    // Returns pointer to best chromosomes in population
    Schedule* GetBestChromosome() const;
    // Returns current generation
    inline int GetCurrentGeneration() const { return _currentGeneration; }
    // Returns pointe to algorithm's observer
    inline ScheduleObserver* GetObserver() const { return _observer; }
private:
    // Tries to add chromosomes in best chromosome group
    void AddToBest(int chromosomeIndex);
    // Returns TRUE if chromosome belongs to best chromosome group
    bool IsInBest(int chromosomeIndex);
    // Clears best chromosome group
    void ClearBest();
};
```