SPECTA Journal of Technology
E-ISSN : 2622-9099
P-ISSN : 2549-2713
Homepage jurnal: https://journal.itk.ac.id/index.php/sjt

Optimasi Jadwal Perkuliahan dengan Modifikasi Algoritme Genetika

Gusti Ahmad Fanshuri Alfarisy
Program Studi Informatika, Institut Teknologi Kalimantan, Balikpapan. Email: *gusti.alfarisy@lecturer.itk.ac.id*

Abstract

Finding an optimal schedule adjusting lecturer's preferences in a limited slot is an arduous task. The manual approach takes time and does not consider various preferences. One approach commonly used is local search. However, this technique tends to be easily trapped in local optima. Hence, a global search, Genetic Algorithm (GA), is employed. This paper proposes a Modified Genetic Algorithm (MGA) that use different search strategy based on the fitness level and additional mutation method. The comparative experimentation shows that MGA is superior to GA in which it can produce a better solution with increasing complexity. Furthermore, the iteration of 400 with the appropriate number of population (80, 90, 100, or above) can be used to maximize the MGA performance. These parameters can be utilized by the application developer when using MGA.

Keywords: genetic algorithm, optimization, scheduling.

Abstrak

Menentukan jadwal optimal yang sesuai dengan preferensi dosen pada tempat jadwal terbatas merupakan tugas yang kompleks. Menggunakan pendekatan manual akan memerlukan waktu dan tidak berorientasi pada kebutuhan preferensi dosen yang cukup beragam. Pendekatan pencarian lokal akan mudah terjebak pada optimum lokal dan tidak efektif dalam menemukan jadwal yang optimum. Sehingga pencarian global menggunakan Algoritme Genetika (GA) digunakan. Penelitian ini mengajukan memodifikasi GA (MGA) yang menggunakan strategi pencarian yang berbeda berdasarkan tingkat nilai kebugaran dan memanfaatkan penambahan strategi mutasi. Hasil uji komparasi menunjukkan bahwa MGA dapat memberikan nilai kebugaran yang lebih baik seiring dengan bertambahnya kompleksitas penjadwalan dibandingkan dengan GA. Selain itu, jumlah iterasi di atas 400 dengan pasangan jumlah populasi yang sesuai (80, 90, 100, atau di atasnya) merupakan parameter yang dapat dikatakan mampu memberikan potensi terbaik dari MGA. Parameter tersebut dapat digunakan oleh pengembang aplikasi penjadwalan yang menggunakan MGA.

Kata Kunci: algoritme genetika, optimasi, penjadwalan.

1. Pendahuluan

Jadwal mata kuliah merupakan komponen sangat penting dalam keberlangsungan proses belajar mengajar di suatu insitutusi pendidikan tinggi. Jadwal mata kuliah memberikan informasi berupa ruang, hari, sesi, serta dosen pengampu sebagai informasi bagi mahasiswa dan juga dosen dalam kegiatan tatap muka. Selain itu, informasi tersebut juga bermanfaat bagi civitas akademika untuk mengatur agenda lain agar tidak bertabrakan dengan jam mengajar dosen atau mengatur kelas pengganti.

Pentingnya jadwal tersebut membuat institusi perguruan tinggi untuk membuat jadwal di unitnya masing-masing. Proses tersebut cukup mudah untuk dipahami namun cukup sulit untuk diimplementasikan terutama saat beradaptasi dengan perubahan. Ditambah lagi, tingkat kerumitan

Submitted August 2020, Revised September 2020, Accepted October 2020, Published December 2020

penjadwalan akan semakin bertambah ketika menimbang banyak sekali preferensi dan batasan dari dosen yang bervariasi. Sehingga otomatisasi penjadwalan mata kuliah akan mempercepat proses penjadwalan.

Di lain sisi, pembangkitan jadwal mata kuliah yang mempertimbangkan banyak preferensi dan dengan keterbatasan sumber daya dapat dikatakan sangat rumit yang termasuk dalam permasalahan *NP-hard* (Lovelace, 2010). Diperlukan suatu metode yang tangguh untuk dapat menemukan solusi jadwal yang optimal. Algoritme Genetika (GA) sudah terbukti mampu menyelesaikan masalah penjadwalan mata kuliah dalam berbagai institusi pendidikan tinggi (Puspaningrum dkk., 2013 dan Sari dkk., 2019).

Sayangnya GA merupakan metode meta-heuristik berbasis populasi yang memerlukan komputasi CPU secara intensif (*CPU intensive task*). Tingkat kompleksitas waktu komputasi akan meningkat ketika banyak gen dan jumlah iterasi yang dilibatkan.

Kelemahan ini dapat dikurangi dengan mencari parameter terbaik dari GA dan melakukan modifikasi agar konvergensi menjadi lebih cepat. Melalui uji parameter, nilai yang membuat algortime tersebut konvergen akan terlihat. Alih-alih langsung menentukan parameter diawal, melakukan eksperimentasi pengujian sangat disarankan. Tanpa penemuan parameter terbaik, kita akan langsung menggunakan parameter tersebut yang mana memungkinkan adanya parameter lain yang jauh lebih efektif dan efisien untuk digunakan. Sedangkan modifikasi akan bermanfaat untuk membuat performa GA jauh lebih unggul dibandingkan GA konvesional yang akan berkontribusi dalam mengurangi biaya komputasi untuk mencapai hasil yang maksimal.

2. Metode

Penelitian ini menggunakan algoritme genetika yang dimodifikasi. Selain itu, perlu adanya penyesuaian pada tahap rekombinasi dan mutasi di permasalahan penjadwalan. Berikut penjelasan terkait GA dan modifikasinya.

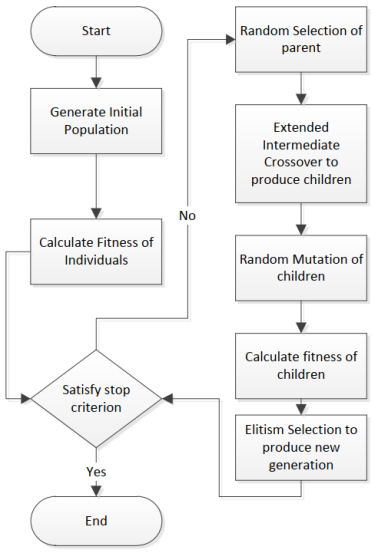
2.1. Algoritme Genetika

GA merupakan salah satu algoritme dari sub-himpunan algoritme evolusi. GA memiliki alur kerja baku yang dimulai dengan inisial populasi, lalu pemilihan induk, rekombinasi, mutasi, dan seleksi untuk generasi selanjutnya. Pada awalnya GA hanya memiliki representasi biner sebagai representasi solusi (Eiben dan Smith, 2015).

Seperti terlihat pada Gambar 1. GA dimulai dengan membangkitkan populasi awal dan setiap individu memiliki tingkat kebugaran yang didapatkan melalui fungsi *fitness* (kebugaran). Selanjutnya terdapat kriteria pemberhentian, kriteria ini dapat berupa jumlah iterasi, batas waktu komputasi, atau tingkat konvergensi. Selama kriteria ini tidak terpenuhi, maka proses evolusi akan terus berlangsung yang meliputi kawin silang antar individu yang dipilih seacra random, diteruskan dengan mutasi, dan pengukuran kualitas individu melalui fungsi *fitness*. Selanjutnya generasi terbaik akan dipilih berdasarkan jumlah individu maksimal. Pemilihan ini merupakan proses penyeleksian individu yang akan diteruskan pada generasi selanjutnya (iterasi berikutnya). Ketika kriteria pemberhentian terpenuhi, proses GA berhenti dan individu terbaik menjadi solusi yang dihasilkan.

Representasi biner tersebut sayangnya tidak dapat digunakan pada permasalahan optimasi penjadwalan mata kuliah. Dengan menggunakan operator genetika biasa, solusi yang dihasilkan dapat melanggar susunan kombinasi dari permasalahan permutasi (Moraglio dan Poli, 2005). Karena representasi yang diperlukan bagi calon solusi jadwal matakuliah adalah kombinasi, maka representasi biner tidak dapat digunakan atau diperlukan perlakuan khusus jika tetap menggunakannya.

Submitted August 2020, Revised September 2020, Accepted October 2020, Published December 2020



Gambar 1: Proses Algoritme Genetika
Sumber: Alfarisy dkk., 2016

Ruang pencarian dari calon solusi yang bersifat kombinasi merupakan kumpulan kandidat solusi yang dapat di cek satu persatu melalui pendekatan enumerasi. Namun, jika ruang pencarian sangat besar, maka akan berkontribusi pada biaya komputasi (Mahmudy, 2015). Sehingga pencarian global seperti GA menjadi metode yang cukup menjanjikan untuk digunakan dalam menghasilkan solusi yang dapat diterima dalam waktu yang rasional.

2.1.1. Rekombinasi

Salah satu operasi kawin silang/rekombinasi pada GA yaitu *one-cut-point crossover* (Mahmudy, 2015). Pertama-tama, kedua individu dipilih secara acak untuk dikawinkan. Selanjutnya secara acak memilih posisi gen yang akan ditukar informasikan ke kromosom lain. Ketika terdapat nilai gen yang sudah ada, maka proses perbaikan dengan mencari informasi gen dari awal dilakukan.

2.1.2. Mutasi

Mutasi pada GA merupakan proses menghasilkan anak atau individu baru melalui perubahan informasi yang ada pada individu itu sendiri tanpa melibatkan individu lain. Berbeda dengan operasi mutasi pada representasi biner dan real, ketika metode mutasi berbeda, lanskap akan berubah sepenuhnya yang dapat mempengaruhi performa pencarian (Serpell dan Smith, 2010).

Jenis mutasi sendiri pada umumnya ada beberapa macam. *SWAP* (menukar nilai antar gen), *INSERT* (menyisipkan nilai gen di gen yang lain), *SCRAMBLE* (memilih kumpulan gen lalu mengacaknya), *INVERSION* (memilih kumpulan gen, lalu membalik urutannya), dan *ASSORTED* (menggunakan operasi yang ada pada *SWAP, INSERT, SCRAMBLE,* dan *INVERSION* secara berurutan) (Serpell dan Smith, 2010)

2.2. Modifikasi Algoritme Genetika

Algoritme Genetika (GA) telah dimodifikasi yang menunjukkan performa lebih baik (selanjutnya disebut MGA). Perubahan yang dilakukan adalah pada penentuan mutasi dan reproduksi individu baru melalui kawin silang.

Mutasi yang digunakan ada dua, yaitu *reciprocal exchange* dan *insertion mutation*. Selama proses evolusi berlangsung, *insertion mutation* digunakan. Setelah evolusi selesai, sebanyak 1000 kali atau iterasi *reciprocal exchange* diterapkan pada solusi terbaik dari populasi dan akan melakukan perbaikan individu setiap iterasinya. Hal ini diterapkan karena pencarian akan cenderung signifikan berubah ketika

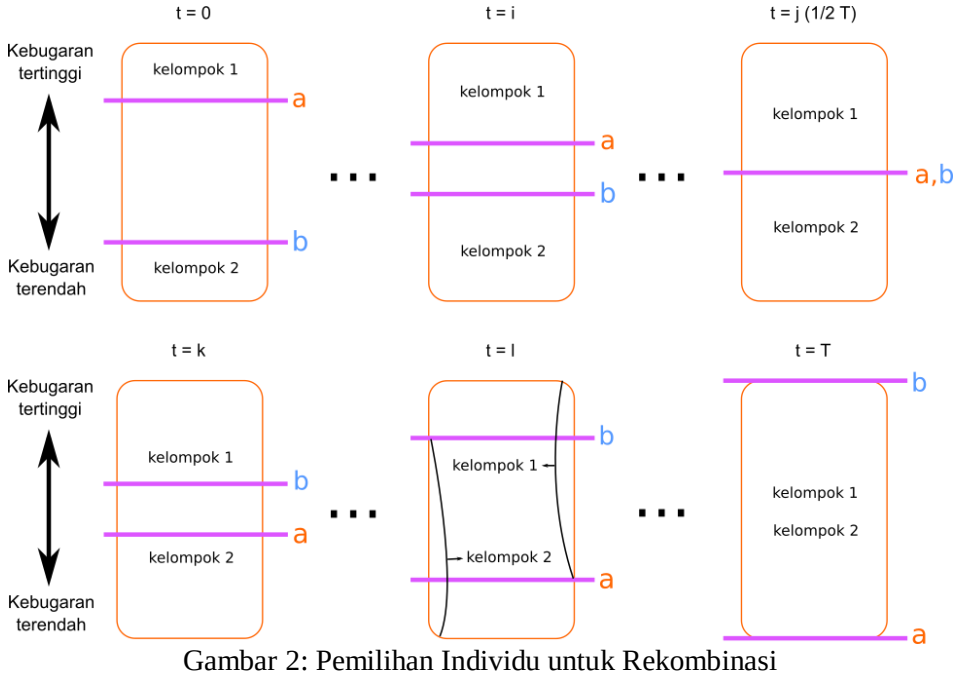
metode mutasi yang digunakan berbeda. Oleh sebab itu, diharapkan eksploitasi yang lebih baik dapat terjadi setelah penggunaan *insertion mutation*.

Sedangkan modifikasi lainnya pada GA diterapkan pada pemilihan individu. Awalnya, GA memilih 2 individu secara acak. Penentuan acak ini dimodifikasi dengan membagi dua kelompok individu yaitu kelompok pertama yang merupakan individu dengan nilai kebugaran cenderung tinggi dan kelompok kedua yang merupakan individu dengan nilai kebugaran cenderung lebih rendah di populasi. Individu pada kelompok pertama dan kedua dipilih kedalam proses rekombinasi. Pembagian kedua kelompok ini pada awalnya memiliki rentang yang sangat jauh, yaitu sangat tinggi dan sangat rendah. Namun seiring berjalannya waktu, rentang ini semakin kecil dan pada akhir generasi tidak memiliki rentang sama sekali (layaknya pemilihan secara acak). Mekanisme ini diilustrasikan pada Gambar 2.

Waktu/iterasi/generasi pada proses evolusi di MGA dinotasikan dengan t sedangkan waktu terakhir dinotasikan dengan T. Notasi i, j, k, dan l, masing-masing menunjukkan bilangan bulat sehingga $0 < i < j < k < l < T$ dan j merupakan separuh waktu atau total iterasi/generasi dari total keseluruhan.

Pada saat $t=0$, kelompok pertama ialah individu pertama sampai individu ke-a dan kelompok kedua merupakan individu ke-b sampai akhir jumlah individu yang dinotasikan dengan N. Individu yang akan direkombinasikan dipilih secara acak pada masing-masing kelompok tersebut sehingga individu dari kelompok 1 akan dikawin silangkan dengan individu di kelompok 2. Pada generasi selanjutnya, pada saat $t=i$, posisi a dan b akan semakin mendekat yang menunjukan bahwa pemilihan individu dari masing-masing kelompok semakin luas. Sampai pada generasi terakhir ($t=T$) tidak terdapat lagi perbedaan antara kelompok satu dan dua yang menunjukkan pemilihan individu layaknya GA pada umumnya.

Dari mekanisme di atas, nilai a dan b tergantung dari nilai t yang didapatkan dari Persamaan (1) dan (2). Nilai awal a dan b pada t=0 masing-masing direpresentasikan dengan simbol α dan β. Fungsi batas bawah digunakan untuk memastikan bahwa pembulatan tidak terjadi dan menghasilkan bilangan bulat yang menunjukkan indeks dari individu.



Gambar 2: Pemilihan Individu untuk Rekombinasi

$$a(t) = \left\lfloor \left(\alpha + \frac{t}{T} . (1 - \alpha) \right) . N \right\rfloor \quad (1)$$

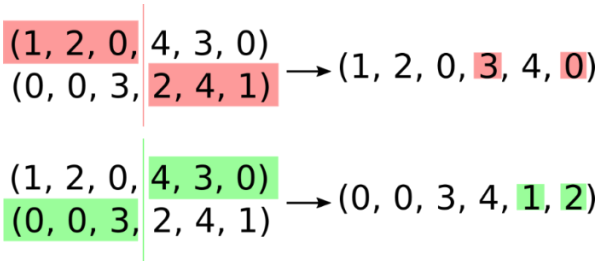
$$b(t) = \left\lfloor \left(\beta . \left(\frac{T-t}{T} \right) \right) . N \right\rfloor \quad (2)$$

Algoritme MGA (*T, cr, mr*)

```
1  Inisialisasi Populasi
2  Evaluasi kebugaran individu
3
4  Selama t < T lakukan
5     Rekombinasi (one-cut-point crossover) dengan pemilihan individu melalui persamaan (1) dan (2)
6     Mutasi dengan insertion mutation
7     Evaluasi kebugaran semua individu
8     Seleksi semua individu dengan elitism
9  Selesai perulangan
10
11 solusi = individu dengan nilai bugar tertinggi dari populasi
12
13 Untuk i = 1 sampai 1000 lakukan
14    individuBaru = dapatkan anak dari reciprocal exchange
15    Evaluasi kebugaran individuBaru
16    Jika nilai bugar individuBaru > nilai bugar solusi maka
17       solusi = individuBaru
18    Selesai Seleksi
19 Selesai Perulangan
20
21 Kembalikan solusi
```

2.2.1. Rekombinasi dengan One-Cut-Point Crossover

Variasi individu melalui rekombinasi pada penelitian ini menggunakan metode *One-Cut-Point Crossover*, namun ada sedikit yang berbeda. Pada umumnya, teknik ini akan langsung mengambil gen pasangannya yang akan disandingkan dengan gen yang dikawinkan. Jika nilai gen sudah dimiliki, maka akan melakukan pengambilan pada gen lainnya. Namun dalam kasus penjadwalan, terdapat gen yang bernilai 0 yang menunjukkan tidak ada jadwal pada slot tabel jadwal. Konsekuensinya adalah diperlukannya pengecekan lebih lanjut karena sangat memungkinkan penggabungan antara keduanya tidak akan meghasilkan kandidat solusi yang sesuai.



Gambar 3: Rekombinasi *One-Cut-Point Crossover* pada Penjadwalan

Sebagai contoh, proses rekombinasi kromosom penjadwalan dapat dilihat pada Gambar 3. Pada contoh tersebut, individu baru yang dihasilkan harus dipastikan memiliki jumlah 0 yang sama dengan kedua induknya dan tidak memiliki nilai gen yang sama (selain 0). Pada contoh pertama (berwarna merah muda), dua gen dari kromosom tersebut digabungkan. 1,2,0 dengan 2,4,1. Nilai gen dimasukkan ke individu baru sesuai dengan urutan gen yang ada pada induk. Namun, pada saat penggabungan, 2 sudah dimiliki oleh individu sebelumnya, sehingga dilakukan pencarian untuk menemukan nilai yang tepat dari gen lainnya pada kromosom yang sama yang ditemukan bernilai 3, sehingga nilai tersebut yang dimasukkan pada individu baru. Hal tersebut juga berlaku untuk gen terakhir (gen dengan nilai 1 yang dimasukkan ke individu baru), karena nilai tersebut sudah dimiliki oleh kromosom baru, maka dilakukan

Submitted August 2020, Revised September 2020, Accepted October 2020, Published December 2020

pencarian. Pada saat dilakukan pencarian tidak ditemukan nilai yang tepat, maka akan diganti dengan 0. Sehingga dengan strategi ini, nilai 0 dapat dikatakan sama untuk semua individu.

Sedangkan pada contoh yang kedua, nilai 0,0,3 dimasukkan dalam individu baru. Selanjutnya adalah 4,3,0. Namun 3 sudah dimiliki oleh individu baru, maka dilakukan pencarian dari gen awal dan ditemukan nilai yang cocok adalah 1. Sedangkan nilai gen berikutnya, yaitu 0, tidak dapat dimasukkan dalam individu baru karena jumlah 0 yang dimiliki sudah sama dengan induknya (yaitu 2) sehingga pencarian nilai gen baru dilakukan dan ditemukan 2.

2.2.2. Mutasi dengan Insertion Mutation

Salah satu teknik mutasi yang digunakan pada penjawalan mata kuliah adalah *Insertion Mutation*. Teknik ini tidak jauh berbeda dengan yang sudah ada, yaitu dengan menyisipkan gen ke dalam gen yang lain. Gen titik penyisipan (*insertion point/ip)* dan pemilihan (*selection point/sp*) dipilih secara acak. Sp akan disisipkan pada gen ip seperti terlihat pada Gambar 4.

$$(1, \overset{ip}{2}, 0, 4, \overset{sp}{3}, 0) \longrightarrow (1, 3, 2, 0, 4, 0)$$

Gambar 4: *Insertion Mutation*

2.2.3. Mutasi dengan Reciprocal Exchange

Teknik mutasi *reciprocal exchange* sejatinya sederhana, yaitu menukarkan posisi kedua gen. kedua gen dipilih (xp1 dan xp2) secara acak seperti terlihat pada Gambar 5.

$$(1, \overset{xp_1}{2}, 0, 4, \overset{xp_2}{3}, 0) \longrightarrow (1, 3, 0, 4, 2, 0)$$

Gambar 5: *Reciprocal Exchange*

2.3. Pemodelan Jadwal Perkuliahan

Dalam menyusun jadwal perkuliahan terdapat tiga objek yang dibutuhkan yaitu jadwal, dosen, dan slot. Jadwal terdiri dari kode mata kuliah, kelas, sks, jumlah mahasiswa, dan dosen yang dapat dinotasikan sebagai himpunan Jadwal seperti terlihat pada Persamaan (3). Kumpulan dosen yang akan mengajar dijadwal tersebut dinotasikan dengan himpunan *D* seperti pada Persamaan (4). Sedangkan slot dinotasikan dengan s pada Persamaan (5).

$$J = \{j_1, j_2, \dots, j_i, \dots, j_n\} \quad (3)$$
$$D = \{d_1, d_2, \dots, d_i, \dots, d_n\} \quad (4)$$
$$s = (0, 1, 4, 5, \dots) \quad (5)$$

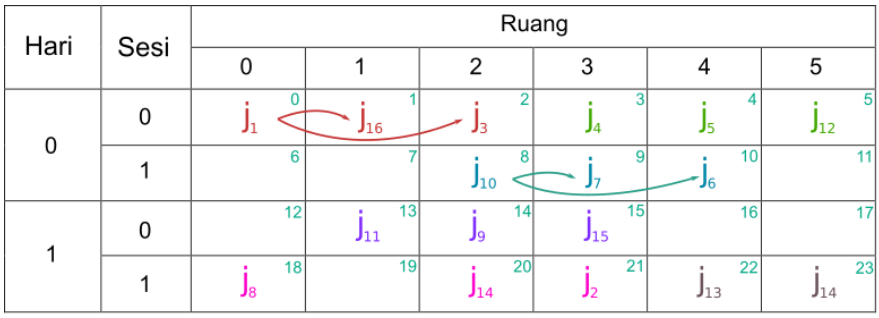
Tiap elemen pada himpunan *J*, terdapat suatu fungsi yang dapat memetakan jadwal ke dalam properti yang tersedia di jadwal tersebut. Pada pemodelan ini fungsi *kelas, sks, jumlah,* dan *dosen* masing-masing akan mengembalikan kelas, sks, jumlah mahasiswa, dan dosen. Sebagai contoh, $kelas(j_3) = "A"$, $sks(j_3) = 3$, $jumlah(j_3) = 78$, dan $dosen(j_3) = d_6$

Dosen sendiri dapat memiliki informasi berupa preferensi yang dapat direpresentasikan dengan fungsi *pref*. Fungsi tersebut memetakan dosen dan indeks pada slot kepada boolean yang menunjukkan apakah pada indeks slot tersebut termasuk preferensi dosen atau tidak. Contoh, $pref(d_2, 6) = true$.

Sedangkan slot sendiri merupakan ruang yang dapat ditempatkan oleh j_i yang jumlah indeksnya adalah perkalian antara jumlah hari, sesi, dan ruang yang digunakan. Slot direpresentasikan dengan array yang berisi j_i dan dinotasikan dalam bentuk vektor *s*. Angka 0 menunjukkan bahwa tidak ada jadwal yang terisi sedangkan nilai vektor > 0 menunjukkan elemen jadwal, contoh 4 merupakan j_4.

$$kebugaran(s) = 1000 - \left(5 \times \left(jkc(s) + jsb(s)\right) + 3{,}3 \times jkl(s) + 1{,}5 \times kpd(s)\right) \quad (6)$$

Baik buruknya suatu jadwal yang dihasilkan sangat tergantung dari beberapa faktor yaitu jumlah kelas yang bentrok (*jkc*), jumlah sks jadwal yang bentrok dengan yang ada di slot (*jsb*), jumlah kelas yang melebihi kapasitas mahasiswa (*jkl*) dan jumlah ketidaksesuaian jadwal dengan preferensi dosen (*kpd*). Tujuan dari jadwal yang dihasilkan adalah meminimalkan semua faktor tersebut.



Gambar 6: Jumlah Kelas Bentrok

Vektor *s* hanyalah kumpulan tempat jadwal yang dapat diisi oleh anggota *J*, tidak terdapat informasi yang spesifik terkait hari, sesi dan ruang dari vektor tersebut. Makna dari s akan sangat bergantung dari penentuan hari, sesi, dan ruang. Sebagai contoh, seperti terlihat pada Gambar 6, indeks ke 8 menunjukkan hari ke-0, sesi ke-1, dan ruang ke-2 yang berisi nilai 10 (j_{10}) jika ditentukan hari sebanyak 2, sesi sebanyak 2, dan ruangan sebanyak 6. Makna dari indeks akan berganti tergantung dari penentuan informasi penting tersebut. Secara keseluruhan representasi jadwal pada Gambar 6 adalah $s = (1, 16, 3, 4, 5, 12, 0, 0, 10, 7, 6, 0, 0, 11, 9, 15, 0, 0, 8, 0, 14, 2, 13, 14)$.

Melalui informasi penting di atas, kita dapat menemukan jumlah kelas yang bentrok satu sama lain melalui fungsi *kelas*. Pada Gambar 3, jumlah kelas bentrok adalah 11 yang dapat dicek dari jadwal di sesi dan hari yang sama apabila terdapat kelas yang sama. Jadwal yang memiliki warna yang sama menunjukkan kelas yang sama yang menunjukkan kumpulan mahasiswa yang sama pada jadwal tersebut. J_1, j_{16}, dan j_3 memiliki kelas yang sama pada hari dan sesi yang sama, dikarenakan j_1 bentrok dengan j_{16} dan j_3 atau j_{16} bentrok dengan j_1 dan j_3. Sehingga, terdapat dua kelas yang bentrok dari ketiga kelas tersebut. Hal tersebut juga berlaku untuk jadwal lainnya.

Selanjutnya, jumlah sks bentrok dapat dilihat pada Gambar 7. Warna merah menunjukkan bahwa jadwal tersebut melebihi durasi sesi yang memungkinkan. Sebagai contoh pada j_{10}, jumlah sks adalah 3, namun ditempatkan pada sesi yang hanya dapat menampung 2 sks (misal sesi 1 adalah dari jam 10:00 sampai jam 11:00). Pada kondisi tersebut, jadwal dikatakan bentrok, oleh karenanya, jumlah sks bentrok pada *s* adalah 8.

Untuk penentuan jumlah kelas yang mahasiswanya melebihi kapasitas ruang, diilustrasikan pada Gambar 8. Fungsi jumlah dapat digunakan pada jadwal dan slot sebagai fungsi untuk mendapatkan jumlah mahasiswa pada jadwal tersebut dan jumlah kapasitas ruangan yang ditunjukkan dari indeks slot. Contoh pada j_3 jumlah mahasiswa adalah 75, namun kapasitas ruangan tersebut hanya 65, kondisi seperti ini dianggap bahwa jadwal yang ditempatkan tidak tepat. Kesuluruhan terdapat 3 kelas yang melebihi kapasitas yang ditunjukkan dengan warna merah.

Hari	Sesi	Ruang					
		0	1	2	3	4	5
0	0	j_1 0	j_{16} 1	j_3 2	j_4 3	j_5 4	j_{12} 5
	1	6	7	j_{10} 8	j_7 9	j_6 10	11
1	0	12	j_{11} 13	j_9 14	j_{15} 15	16	17
	1	j_8 18	19	j_{14} 20	j_2 21	j_{13} 22	j_{14} 23

jsb(s) = 8

$sks(j_{10}) = sks(j_7) = sks(j_6) = 3$
$sks(s, 8) = sks(s, 9) = sks(s, 10) = 2$

Gambar 7: Jumlah SKS Jadwal yang Tidak Sesuai dengan Slot

Hari	Sesi	Ruang					
		0	1	2	3	4	5
0	0	j_1 0	j_{16} 1	j_3 2	j_4 3	j_5 4	j_{12} 5
	1	6	7	j_{10} 8	j_7 9	j_6 10	11
1	0	12	j_{11} 13	j_9 14	j_{15} 15	16	17
	1	j_8 18	19	j_{14} 20	j_2 21	j_{13} 22	j_{14} 23

jkl(s) = 3

$jumlah(j_3) = 75$ $sks(s, 2) = 65$
$jumlah(j_7) = 70$ $sks(s, 2) = 65$
$jumlah(j_8) = 100$ $sks(s, 2) = 80$

Gambar 8: Jumlah Kelas Melebihi Kapasitas

Tanda yang sama (merah) juga ditunjukkan pada Gambar 9 sebagai indikasi bahwa slot tersebut tidak termasuk dalam preferensi dosen.Total keseluruhan jadwal yang tidak memenuhi keinginan dosen sebanyak 2.

Setiap faktor memiliki tingkat kepentingan sendiri, seperti *jkc* dan *jsb* lebih penting dibandingkan dengan *jkl* dan *jkl* lebih penting dibandingkan dengan *kpd*. Dengan kata lain, dua faktor pertama menjadi fokus utama untuk dipenuhi terlebih dahulu dibandingkan dengan *kpd*. Preferensi dosen tidak terpenuhi tidak menjadi masalah yang besar dibandingkan dengan jumlah kelas yang bentrok. Semua faktor tersebut termasuk dalam kendala lembut. Oleh karenanya masing-masing faktor diberikan bobot tertentu sesuai tingkat kepentingannya.

Hari	Sesi	Ruang					
		0	1	2	3	4	5
0	0	j_1 0	j_{16} 1	j_3 2	j_4 3	j_5 4	j_{12} 5
	1	6	7	j_{10} 8	j_7 9	j_6 10	11
1	0	12	j_{11} 13	j_9 14	j_{15} 15	16	17
	1	j_8 18	19	j_{14} 20	j_2 21	j_{13} 22	j_{14} 23

kpd(s) = 2

$pref(dosen(j_5), 4) = false$
$pref(dosen(j_9), 14) = false$

Gambar 9: Jumlah Kelas yang Tidak Sesuai dengan Preferensi Dosen

Melalui bobot pada tiap-tiap faktor, nilai kebugaran dapat dirumuskan seperti pada Persamaan (6) melalui fungsi yang disebut kebugaran. Semakin tinggi nilai bugar, semakin baik individu tersebut. Untuk dapat memperolah nilai bugar, maka semua faktor dijumlahkan dengan masing-masing bobot yang nantinya akan dijadikan pengurang dengan nilai yang cukup besar (1000), dikarenakan invers perkalian dapat memicu pembagian dengan nol.

Jumlah kelas dan sks yang bentrok sangat krusial untuk dipenuhi sehingga memiliki bobot paling tinggi. Namun, pada penelitian ini, kedua faktor tersebut tidak dijadikan kedalam kendala keras. Hal ini dimaksudkan agar pencarian solusi oleh MGA akan lebih efektif karena tidak dibatasi atau dilakukan perbaikan saat terjadi jadwal yang bentrok. Selain itu proses perbaikan akan meningkatkan biaya pencarian. Pemberian bobot yang tinggi diharapkan dapat menggantikan peran kendala berat tanpa menganggu proses pencarian alamiah oleh MGA.

2.4. Skenario Uji Parameter dan Komparasi

Skenario pengujian dibuat dengan dua tujuan yaitu untuk menemukan parameter yang dapat dikatakan baik, untuk mencapai solusi optimum lebih efektif, untuk dapat digunakan oleh pengembang atau peneliti lain, dan untuk melihat keandalan dari algoritme yang dimodifikasi. Karena bersifat stokastik, pengujian dilakukan sebanyak 50 kali dan rata-ratanya dijadikan acuan untuk mengukur kinerja dari parameter. Sedangkan uji komparasi akan melihat maksimum, minumum, rata-rata, dan standard deviasinya.

Pengujian parameter diterapkan pada jumlah populasi, iterasi, *crossover rate*, dan *mutation rate* seperti terlihat pada Tabel 1. Jumlah populasi dan iterasi sejatinya adalah dua hal yang memiliki potensi saling berhubungan. Ketika jumlah populasi rendah dengan iterasi yang rendah akan menghasilkan kebugaran yang rendah pula, sebaliknya apabila jumlah populasi dan iterasi tinggi maka kebugaran memiliki nilai yang tinggi. Disini, akan dianalisis nilai parameter yang dapat dikatakan baik dalam menemukan solusi optimum. Parameter tersebut ditentukan dari tingkat signifikansi kenaikan. Perubahan parameter yang tidak berdampak signifikan tidak akan dijadikan pilihan parameter walau memberikan kenaikan pada nilai kebugaran. Pemilihan tersebut dilakukan agar parameter rasional untuk digunakan, semakin tinggi populasi dan iterasi akan berkontribusi pada waktu komputasi. Sedangkan pada *crossover rate* dan *mutation rate* diuji secara enumerasi dengan kenaikan atau penurunan 0,1. Nilai ini merupakan presentase dari jumlah anak yang dihasilkan.

Tabel 1: Skenario pengujian

Skenario Pengujian	**Parameter**
Uji Jumlah Populasi dan Iterasi	Populasi 10 sampai 100 dengan kenaikan 10 untuk tiap iterasi 50 – 800 dengan kenaikan 50
Uji *Crossover* dan *Mutation Rate*	*Crossover rate* mulai dari 0 sampai 1 dengan kenaikan 0,1 yang dipasangkan dengan *mutation rate* mulai dari 1 sampai 0 dengan penurunan 0,1
	Cr 0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 Mr 1.0 0.9 0.8 0.7 0.6 0.5 0.4 0.3 0.2 0.1 0
Komparasi	Komprasi dilakukan dengan GA dengan jumlah jadwal kelas yang berbeda yaitu 50, 88, 100, 150, 200, 250, dan 300 pada slot jadwal yang sama. Sedangkan jumlah iterasi disesuaikan dengan jumlah variasi pada masing-masing algoritme agar didapatkan jumlah iterasi yang sama

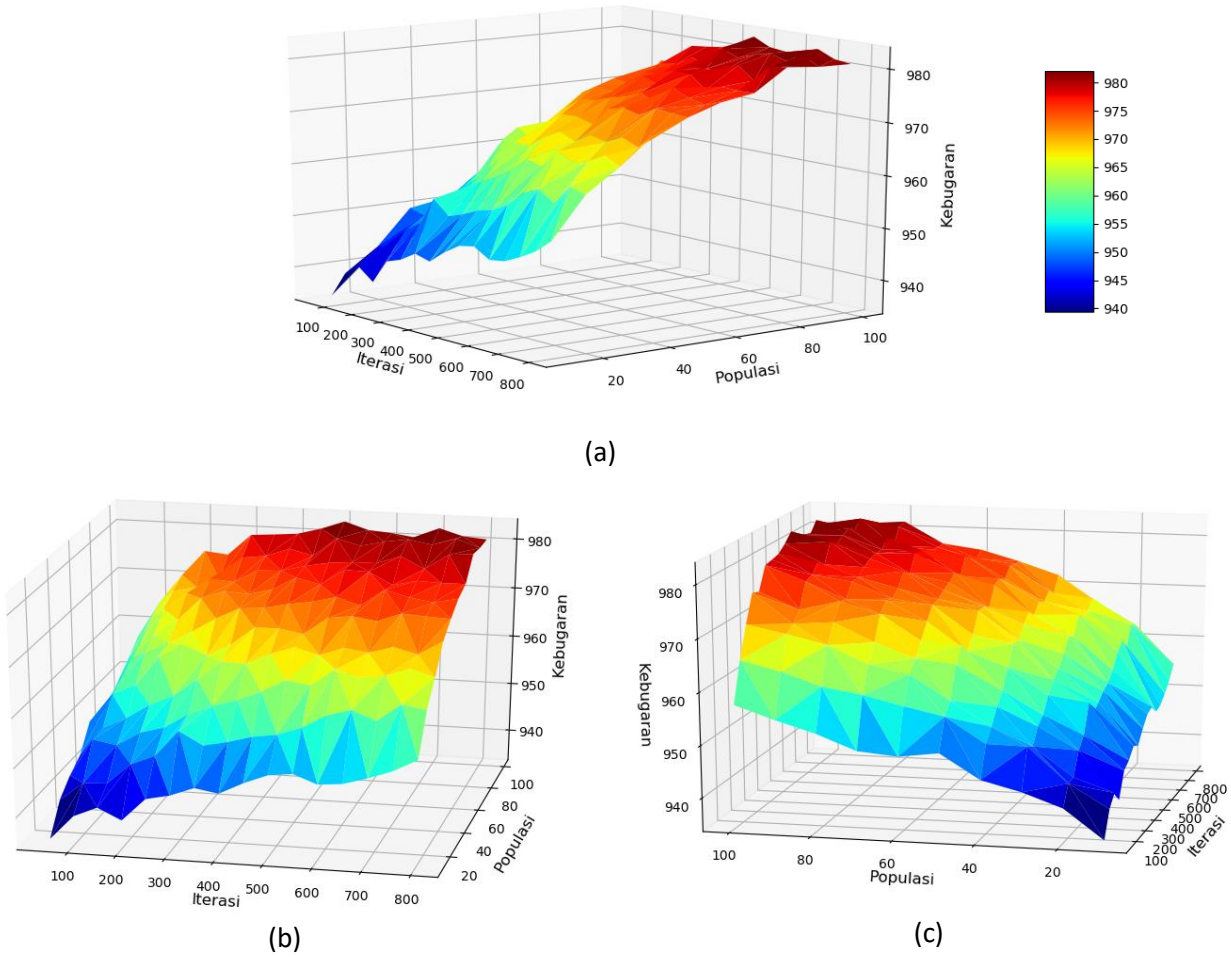
Pada uji komparasi, jumlah jadwal yang berbeda ditentukan untuk mengukur keandalan dari modifikasi algoritme genetika. Selain itu terdapat jumlah jadwal yang menyerupai kasus sebenarnya di Institut Teknologi Kalimantan (jadwal dengan jumlah 88) untuk mengukur tingkat keberhasilan pada kasus yang nyata. Jumlah jadwal mula-mula dimuat sebesar 50 lalu naik 100, lalu 150, sampai 300 dengan kelipatan 50.

3. Hasil dan Pembahasan

Peneliti telah melakukan eksperimentasi melalui skenario pengujian dan didapatkan hasil dari uji parameter dan komparasi. Hasil dan analisis dapat dilihat pada sub-bab berikut.

3.1. Uji Parameter

Hasil dari uji parameter iterasi dan populasi dapat dilihat pada Gambar 10. Semakin banyak iterasi dan jumlah populasi yang diberikan, nilai kebugaran akan semakin meningkat dan landai pada populasi 80 – 100 dengan iterasi 700 – 800 (a). Lebih jauh lagi, walaupun jumlah iterasi tinggi namun dengan populasi yang rendah akan memberikan nilai kebugaran yang rendah sekitar 950 - 960 (b). Begitu pula dengan jumlah populasi, walaupun tinggi namun dengan iterasi yang rendah akan memberikan nilai kebugaran yang rendah pula sekitar 955 – 965 (c). Oleh karenanya, paremeter yang baik akan dipilih dengan nilai rata-rata kebugaran di atas 980.



Gambar 10: Grafik Hasil Parameter Iterasi dan Populasi terhadap Kebugaran. Ketiga Gambar Tersebut (a, b, dan c) adalah Sama Namun Dilihat dari Sudut yang Berbeda.

Submitted August 2020, Revised September 2020, Accepted October 2020, Published December 2020

Nilai rata-rata kebugaran di atas standar tersebut dapat dilihat pada Tabel 2. Rata-rata menunjukkan rata-rata kebugaran pada 50 kali percobaan dan variasi menunjukkan jumlah anak yang dihasilkan dari operasi genetika (rekombinasi dan mutasi). Semakin tinggi nilai variasi menunjukkan semakin tinggi waktu komputasi, nilai ini didapatkan dari perkalian antara jumlah iterasi dengan jumlah populasi.

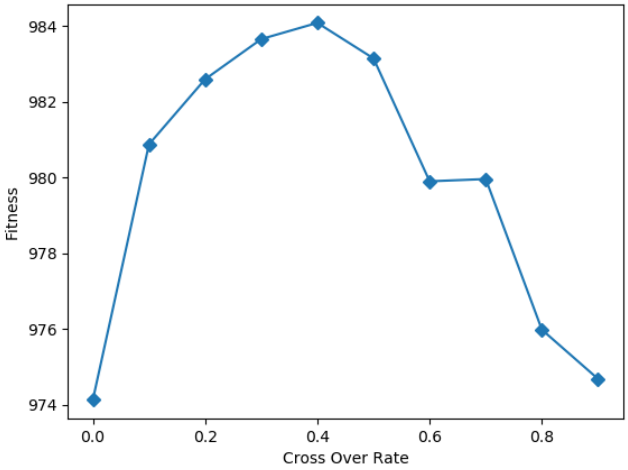
Tabel 2: Jumlah iterasi dan populasi dengan kebugaran di atas 980

Iterasi	Populasi	Rata-rata	Variasi
400	100	980.104	40000
450	100	981.378	45000
500	90	981.378	45000
500	**100**	**983.056**	**50000**
550	90	980.156	49500
550	100	981.55	55000
600	90	981.21	54000
600	100	981.168	60000
650	80	980.364	52000
650	90	981.918	58500
650	100	980.492	65000
700	90	981.052	63000
700	100	982.608	70000
750	80	981.11	60000
750	90	982.498	67500
750	100	981.212	75000
800	80	981.506	64000
800	90	980.93	72000
800	100	980.82	80000

Hasilnya menunjukkan iterasi sebanyak 500 dengan jumlah populasi sebesar 100 memberikan nilai kebugaran paling tinggi dibandingkan dengan parameter lain. Ini tentu tidak membuat parameter tersebut menjadi parameter yang terbaik karena model bersifat stokastik. Dengan mengambil batas bawah sebesar 980, maka dapat dikatakan jumlah iterasi di atas 400 dengan pasangan populasinya seperti terlihat pada Tabel 2. dapat dipilih sebagai parameter untuk menghasilkan jadwal yang optimal pada kasus penjadawalan dengan 88 jadwal (besaran jadwal dari kasus nyata di Institut Teknologi Kalimantan). Hal ini juga menunjukkan bahwa walaupun populasi dan iterasi meningkat, kebugaran tidak dapat memberikan nilai yang lebih baik atau memberikan nilai yang signifikan. Sehingga parameter tersebut dipilih untuk kasus penjadwalan pada MGA.

Sedangkan hasil uji dari pasangan *crossover rate* dengan *mutation rate* tergambar pada Gambar 11. Pada grafik tersebut juga menunjukkan informasi berupa Mr yang mana jika Cr adalah 0,6 maka Mr adalah 0,4 sesuai dengan skenario pengujian yang sudah dibuat. Pada gambar tersebut terlihat dengan meningkatnya Cr mulai dari 0 terjadi peningkatan nilai kebugaran sampai pada Cr sebesar 40% (0,4). Ini menunjukkan bahwa dengan melakukan hanya mutasi saja akan memberikan kebugaran yang rendah yaitu sekitar 974. Dengan ditambahkan rekombinasi sebesar 10% (0,1) dari total populasi mampu meningkatkan kebugaran secara signifikan dan terus bertambah sampai total rekombinasi sebesar 40% (0,4). Pada nilai inilah, kebugaran mencapai kemampuan maksimalnya yang mana ketika rekombinasi ditingkatkan dan mutasi dikurangkan akan memicu penurunan seperti terlihat pada Gambar 11 yang dimulai dari 50% (0,5) sampai 100% (1,0) rekombinasi. Dari hasil empiris tersebut, Cr sebesar 0,4 dan Mr sebesar 0,6 dipilih sebagai parameter yang baik yang akan diuji dengan GA standar.

Submitted August 2020, Revised September 2020, Accepted October 2020, Published December 2020



Gambar 11: Hasil Cr dan Mr terhadap Kebugaran

3.2. Uji Komparasi

Hasil komparasi MGA dengan GA dapat dilihat pada Tabel 3. Mean menunjukkan rata-rata, min menunjukkan hasil minimum, max menunjukkan hasil maksimum, dan std menunjukan standar deviasi. Semua nilai diambil dari hasil 50 kali percobaan.

Tabel 3: Hasil komparasi

Jumlah Jadwal	GA				MGA			
	Mean	Min	Max	Std	Mean	Min	Max	Std
50	980.092	958.5	989.5	7.368116	**984.812**	**973.2**	**994**	**4.929002**
88	973.97	950.6	991	10.75203	**981.608**	**961.8**	**994**	**7.213483**
100	960.74	926.4	983.5	11.5421	**974.426**	**952.8**	**988**	**8.335366**
150	871.544	827.3	929	21.64502	**917.002**	**887.8**	**949.1**	**14.02506**
200	740.262	671.3	803.3	32.21765	**820.754**	**765.7**	**861.8**	**20.56738**
250	538.108	469.3	611.1	**31.00871**	**641.566**	**572**	**718.3**	31.08093
300	319.028	237.8	408.5	39.17694	**430.11**	**358.9**	**495.7**	**32.35831**

Dari hasil komparasi tersebut MGA lebih andal dibandingkan dengan GA dan pada kasus yang lebih kompleks MGA jauh lebih unggul. Pada semua jumlah jadwal yang diujikan, MGA mampu meberikan hasil yang lebih baik sebagaimana terlihat pada gaya huruf tebal. Hal ini menunjukkan bahwa MGA lebih efektif dalam menghasilkan jadwal optimum dibandingkan dengan GA walau dengan jumlah variasi yang sama. Selain itu, standar deviasi MGA meunjukkan hasil yang lebih rendah pada hampir semua jadwal yang menunjukkan tingkat konsistensi dari model yang diajukan.

4. Kesimpulan

Penulis telah melakukan modifikasi terhadap algoritme genetika (MGA) dalam mencari solusi jadwal yang optimal. Dari hasil pengujian, MGA mampu memberikan hasil yang lebih efektif dibandingkan dengan algoritme genetika standar dengan jumlah variasi yang sama. Selain itu, ketika kompleksitas pencarian semakin meningkat, MGA memberikan hasil yang jauh lebih efektif.

Sedangkan pada parameter, hasil uji menunjukkan iterasi di atas 400 dengan pasangan jumlah populasinya (80, 90, dan 100) dapat memberikan nilai kebugaran yang lebih baik. Hal ini juga menunjukkan bahwa terdapat kemungkinan hubungan yang kuat antara jumlah iterasi dengan jumlah populasi. Jumlah populasi yang tinggi dengan iterasi yang cukup rendah tidak dapat memberikan hasil yang efektif. Begitu juga dengan iterasi yang tinggi dengan jumlah populasi yang rendah. Sehingga

pengujian diperlukan secara simultan terhadap kedua parameter tersebut. Dilain pihak, *crossover rate* sebesar 0,4 dengan *mutation rate* sebesar 0,6 mampu memberikan hasil yang terbaik untuk kasus 88 jadwal. Proporsi mutasi pada MGA yang sedikit lebih besar dapat memberikan hasil yang lebih baik.

Untuk studi selanjutnya MGA dapat ditingkatkan melalui strategi mutasi, konkuren, dan penambahan faktor optimalitas. Mutasi tambahan pada MGA dapat memberikan solusi yang lebih baik, sehingga berbagai macam teknik mutasi yang digunakan di GA berpotensi untuk memberikan performa yang lebih baik. Model ini juga dapat ditingkatkan dengan komputasi konkuren dimana masing-masing populasi dijalankan di core CPU yang berbeda secara bersama-sama sehingga memungkinkan untuk mempercepat waktu komputasi. Lebih jauh lagi, optimasi ini akan sangat berguna ketika mempertimbangkan faktor lain seperti jarak kelas atau jumlah jadwal berturut-turut.

Daftar Pustaka

Alfarisy, G. A. F., Sihananto, A. N., Fatyanosa, T. N., Burhan, M. S., dan Mahmudy, W. F. (2016) 'Hybrid genetic algorithm and simulated annealing for function optimization', *Journal of Information Technology and Computer Science*, Vol. 1, No. 2: 82–97.

Eiben, A. E. dan Smith, J. E. (2015) *Introduction to Evolutionary Computing* (Volume 12), New York.

Lovelace, A. L. (2010) *On the Complexity of Scheduling University Courses* California Polytechnic State University.

Mahmudy, W. F. (2015) 'Dasar-dasar algoritma evolusi' *Modul Kuliah: Program Teknologi Informasi Dan Ilmu Komputer (PTIIK) Universitas Brawijaya*.

Moraglio, A. dan Poli, R. (2005) 'Topological crossover for the permutation representation' dalam *Proceedings of the 2005 Workshops on Genetic and Evolutionary Computation - GECCO '05*, 332.

Puspaningrum, W. A., Djunaidy, A. dan Hakim, J. A. R. (2013) 'Penjadwalan Mata Kuliah Menggunakan Algoritma Genetika di Jurusan Sistem Informasi ITS', *Jurnal Teknik POMITS*, Vol. 2, No. 1: 5.

Sari, Y., Alkaff, M., Wijaya, E. S., Soraya, S., dan Kartikasari, D. P. (2019) 'Optimasi Penjadwalan Mata Kuliah Menggunakan Metode Algoritma Genetika dengan Teknik Tournament Selection', *Jurnal Teknologi Informasi dan Ilmu Komputer*, Vol. 6, No. 1: 85.

Serpell, M. dan Smith, J. E. (2010) 'Self-Adaptation of Mutation Operator and Probability for Permutation Representations in Genetic Algorithms', *Evolutionary Computation*, Vol. 18, No. 3: 491–514.

Submitted August 2020, Revised September 2020, Accepted October 2020, Published December 2020