semanTIK, Vol.7, No.2, Jul-Des 2021, pp. 175-182
ISSN: 2502-8928 (Online)
DOI: 10.5281/zenodo.5793087
Terakreditasi "Peringkat 4 (Sinta 4)" oleh Kemenristekdikti
Website: http://ojs.uho.ac.id/index.php/semantik

ANALISIS PERFORMA DARI *ONE-POINT*, *MULTI-POINT* DAN *ORDER CROSSOVER* DI ALGORITMA GENETIKA

Ahmad Miftah Fajrin[*1]
[1]Program Studi Teknik Informatika, Fakultas Teknik, Universitas Surabaya
e-mail: [*1]**ahmadmiftah@staff.ubaya.ac.id**

Abstrak

Algoritma Genetika (GA) adalah salah satu algoritma yang powerful untuk menyelesaikan masalah penjadwalan mata kuliah. Pada GA, terdapat operator *crossover* yang berperan aktif dalam pembuatan anak atau *offspring*. *Crossover* juga menjadi fondasi dalam menghasilkan solusi yang optimal. Kesalahan dalam pemilihan *crossover* membuat meningkatnya tingkat pelanggaran atau *fitness* terhadap *constraint*. Semakin tinggi nilai *Fitness* maka semakin buruk solusi yang dihasilkan. Pada penelitian ini, dilakukan analisis terdapat jenis *crossover* yang ada di GA yaitu *One-Point Crossover*, *Multi-Point Crossover* dan *Order Crossover*. Analisis yang dilakukan pada penelitian ini adalah dengan membandingkan nilai *fitness* dan waktu eksekusi antara jenis *crossover* tersebut. Hasil penelitian menunjukkan bahwa nilai *fitness* yang paling kecil dapat dihasilkan oleh *One-Point Crossover* pada 9 dataset. Untuk waktu eksekusi yang paling cepat dapat dihasilkan oleh *Multi-Point Crossover* pada 12 dataset.

Kata kunci; Algoritma Genetika, *Crossover*, Penjadwalan, Pelanggaran

Abstract

Genetic Algorithm (GA) is one of the most powerful algorithms to solve scheduling problem. In GA has crossover operator that plays an important role for making offspring. Crossover is also a foundation in producting an optimal solution. Failure to choose Crossover will increase a violation or fitness value to the constraint. The higher fitness value in solution will get the worst solution. In this research, three types of crossover will be analyzed, namely One-Point Crossover, Multi-Point Crossover and Order Crossover Mechanism. The analysis carried out in this research is comparing a fitness value and execution time between three crossovers. The result shows that the smallest fitness value can be generated by One-Point Crossover on 9 datasets. For the fastest execution time can be generated by Multi-Point Crossover on 12 datasets.

***Keywords**; Algoritma Genetika, Crossover, Timetabling, Violation*

1. PENDAHULUAN

Algoritma Genetika (GA) adalah algoritma yang meniru proses seleksi yang ada di alam. GA adalah salah satu algoritma *metaheuristic* yang dapat menyelesaikan permasalahan yang berkaitan dengan jenis masalah NP-Hard [1]. Dari beberapa algoritma *metaheuristic* yang ada seperti *Colony* dan *Ant*, GA adalah algoritma yang paling powerful dan optimal untuk menyelesaikan masalah penjadwalan [2].

Untuk menyelesaikan masalah penjadwalan, GA mempunyai beberapa

komponen operator yang penting yaitu c*rossover* dan *mutation*. Operator yang paling penting dalam proses di GA adalah *crossover* [3]. Ada banyak jenis *crossover* yang ada di GA seperti *One-Point Crossover*, *Multi-Point Crossover* dan *Order Crossover*. Jenis *crossover* tersebut adalah *crossover* yang sering digunakan. Pemilihan *crossover* menjadi sangat penting agar mendapatkan solusi yang paling optimal.

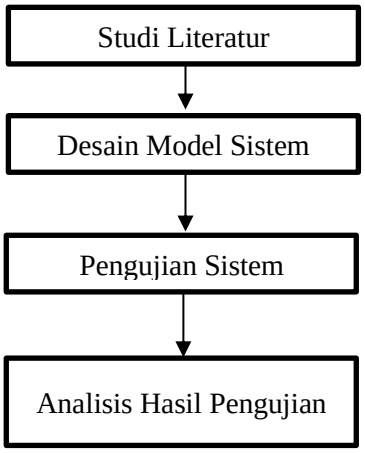
Pada kasus penjadwalan mata kuliah, kompleksitas untuk menyelesaikannya sangat tinggi [4]. Salah satunya adalah karena ada *constraint.* Pada kasus penjadwalan terdapat dua *constraint* yaitu *hard* dan *soft constraint*. *Hard constraint* berarti kriteria yang dibuat harus terpenuhi semuanya. Sedangkan *soft constraint* berarti kriteria yang sebisanya terpenuhi seluruhnya namun masih dibolehkan tidak terpernuhi.

Untuk menyelesaikan permasalahan tersebut, penelitian yang sudah dilakukan terdahulu menggunakan GA terbukti menghasilkan solusi penjadwalan yang optimal dalam hal pemenuhan *constraint* [5][6]. Penggunaan GA khususnya pada operator c*rossover* menurut beberapa penelitian sangat penting karena mempengaruhi proses eksplorasi [7] [8] [9]. Penggunaan *crossover* pada penjadwalan menentukan performa dari proses yang ada di GA [10]. Penelitian terkait penggunaan *One-Point* dilakukan untuk menyelesaikan optimasi [11] [12]. *Multi-Point Crossover* juga digunakan untuk penyelesaikan terkait masalah penjadwalan mata kuliah[13]. *Order Crossover* pada GA digunakan untuk menyelesaikan masalah penjadwalan [14]. Penelitian yang sudah dilakukan masing-masing menyimpulkan *crossover* pada penelitiannya yang paling baik dalam hal nilai pelanggaran atau *fitness* yang semakin menurun terhadap *constraint*. Semakin rendah nilai *fitness* maka semakin baik solusi yang dihasilkan

Berdasarkan masalah yang sudah dijelaskan, maka pada penelitian ini dilakukan analisis perfoma terkait dengan beberapa jenis *Crossover*. *One-Point Crossover*, *Multi-Point Crossover* dan *Order Crossover* dilakukan analisis terkait dengan performanya. Diharapkan pada analisis ini dapat didapatkan informasi bahwa manakah jenis *crossover* yang paling cocok untuk permasalahan penjadwalan mata kuliah**.**

2. METODE PENELITIAN

Desain penelitian dalam penelitian ini adalah studi literatur, desain model sistem, pengujian sistem dan analisis hasil pengujian. Lebih jelasnya dapat dilihat pada Gambar 1.



Gambar 1. Desain penelitian

2.1 Studi Literatur

Pada tahapan ini, dilakukan pencarian informasi dari literatur dan jurnal yang tersedia mengenai penjadwalan mata kuliah, dataset dan algoritma genetika.

2.2 Desain Model Sistem.

Pada tahapan ini, dilakukan desain sistem yang berfokus pada tiga jenis *crossover* di GA pada kasus penjadwalan mata kuliah.

2.2.1 Penjadwalan Mata Kuliah

Penjadwalan mata kuliah di universitas adalah salah satu permasalahan yang paling sering diteliti. Permasalahan penjadwalan adalah masalah *real world* yang kompleks. Penjadwalan mata kuliah ini sangat berkaitan erat dengan mata kuliah, ruang dan waktu. Semakin banyak jumlah mata kuliah maka semakin kompleks permasalahan. Masalah tersebut dapat digambarkan sebagai kumpulan mata kuliah MK = $\{MK_1, MK_2, \ldots, MK_n\}$ yang ditempatkan di Ruang R = $\{R_1, R_2, \ldots R_n\}$, Hari H = $\{H_1, H_2, \ldots H_n\}$ dan Waktu W = $\{W_1, W_2, \ldots W_n\}$.

Pada penjadwalan, terdapat *constraints* yang harus diselesaikan yaitu *hard* dan *soft constraint.*

Untuk *hard constraint* sebagai berikut:

1. *Lectures:* Semua *lectures of course* harus dijadwalkan di *timeslot* (*day*, *period*) yang tersedia.
2. *Room occupancy:* Dua *lectures* tidak boleh ditempatkan di ruang dan periode yang sama.
3. *Conflicts*: *Lectures of course* yang dalam satu grup kurikulum atau diajarkan oleh dosen yang sama harus dijadwalkan di periode yang berbeda.
4. *Availability: Course* mempunyai spesifik period yang tidak bisa ditempatkan di periode tertentu

Untuk *soft constraint* sebagai berikut :

1. *Room Capacity:* Jumlah mahasiswa yang mengambil mata kuliah harus kurang atau sama dengan jumlah kapasitas ruangan. Setiap mahasiswa yang tidak mendapatkan kursi maka dikenakan pinalti sebesar satu point.
2. *Room stability* : Semua *lectures of course* dalam satu *course* seharusnya ditempatkan di ruangan yang sama. Jika ada *lectures of course* yang berbeda ruangan maka dikenakan pinalti sebanyak satu point.
3. *Minimum Working Days* : *Lectures* dari setiap *course* harus disebarkan ke dalam jumlah minimum hari yang diberikan. Setiap kekurangan hari akan dikenakan pinalti sebanyak lima point.
4. *Curriculum Compactness* **:** *Lectures* dalam satu kurikulum seharusnya beriringan satu sama lain. Setiap *lectures* yang tidak beriringan dikenakan pinalti sebanyak dua point.

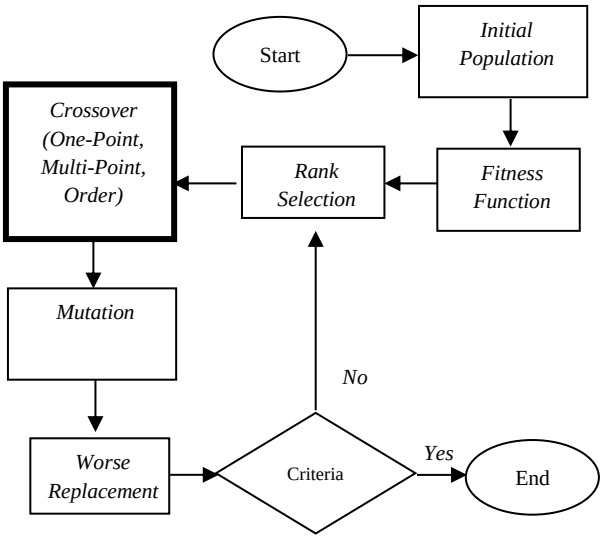
2.2.2 Dataset

Dataset pada penelitian ini menggunakan dataset dari *International Timetabling Competition* (ITC-2007). CB-CTT adalah dataset yang mempunyai 21 jenis dataset yang berbeda. Masing-masing dataset mempunyai keunikan tersendiri. Dari yang sederhana sampai kompleks terdapat pada dataset tersebut. Tujuan dari dataset CB-CTT yang dilombakan adalah untuk menemukan konfigurasi yang terbaik agar pelanggaran yang terjadi pada *constraint* semakin kecil. Pada data CB-CTT terdapat atribut yang melekat yaitu :

1. *Courses*: **:** Setiap kursus (*course)* terdiri dari jumlah pengajaran (#*lectures*) yang memiliki jumlah mahasiswa (#students) yang sudah ditetapkan untuk dijadwalkan di periode yang tertentu.
2. *Rooms* : Setiap ruang juga memiliki kapasitas kursi.
3. *Days* : Jumlah hari yang tersedia
4. *Period Per Day* : Jumlah periode dalam satu hari
5. *Curricula* : Suatu grup mata pelajaran.
6. *MinWorkingDays:* Jumlah minimal hari *course* yang harus ada

2.2.3 Algoritma Genetika

Algoritma Genetika mempunyai beberapa proses diantaranya *initial population, fitness function, selection, crossover, mutation* dan *worst replacement.* Untuk alur dari Algoritma Genetika dapat dilihat pada Gambar 2



Gambar 2. Algoritma Genetika

a. Initial Population

GA akan membuat sekumpulan populasi awal yang menggunakan teknik pengkodean kromosom. Kromosom bisa dianggap sebagai sekumpulan individu yang ada di alam dan siap untuk diseleksi. Panjang kromosom bergantung pada panjangnya urutan gen. Untuk teknik pengkodean kromosom dapat dilihat pada Gambar 3.

Mata Kuliah	*Hari*	*Waktu*	*Ruangan*
MK-1	H-1	W-1	R-1
MK-2	H-2	W-2	R-2
…	…	…	…
MK-n	H-d	W-t	R-r

Gambar 3. Pengkodean kromosom

n : jumlah total mata kuliah
d : jumlah total hari
t : jumlah total waktu
r : jumlah total ruangan.

b. Fitness Function

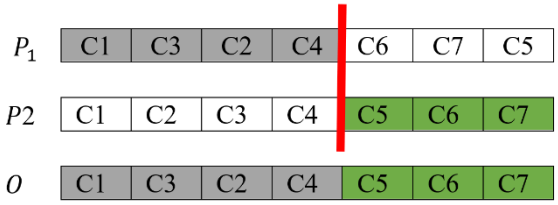
Fungsi *fitness* digunakan sebagai alat bantu untuk melihat dari hasil solusi apakah baik atau buruk dalam sebuah permasalahan. Dalam hal penjadwalan, kualitas dari sebuah *chromosome* diukur dari banyaknya *constraint* yang dipenuhi atau jumlah *constraint* yang dilanggar. Setiap *chromosome* akan dicek validitas dengan memeriksa *hard* dan *soft constraint.* Pada *Hard constraint*, semua kasus harus tanpa terjadi pelanggaran dan pada *soft constraint* boleh terjadi pelanggaran tetapi pelanggarannya sekecil mungkin. Setiap *soft constraint* yang dilanggar akan diberikan nilai pinalti.

c. Rank Selection

Rank Selection (RS) adalah teknik yang memakai konsep eksplorasi yang dapat mencegah terjadinya konvergensi *premature*. *Rank Selection* akan mengurutkan populasi berdasarkan nilai *fitness* dan memberikannya peringkat [15]. Nilai *fitness* yang terbanyak akan berada di urutan pertama dan sedangkan nilai *fitness* terkecil berada di urutan paling akhir

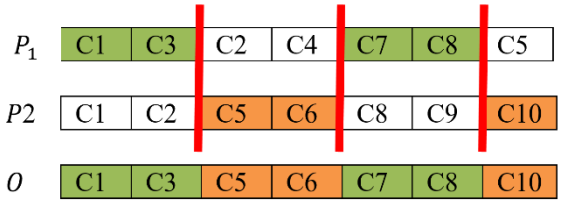
d. Crossover

Crossover adalah proses perkawinan dari individu yang sudah terpilih pada waktu *rank selection.* Ada beberapa *Crossover* yang popular yaitu *One-Point Crossover, Multi-Point Crossover* dan *Order Crossover*. Gambar 4 adalah gambar untuk proses *One-Point Crossover*.



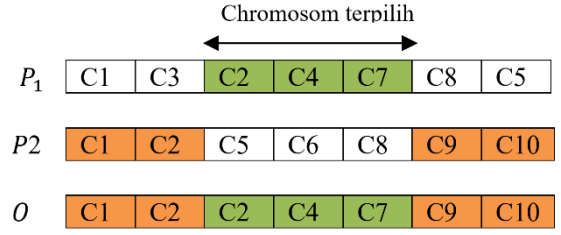
Gambar 4. *One-Point Crossover*

One-point Crossover akan mencari satu potongan di antara urutan *chromosomes*. Pada Gambar 4 terjadi pemotongan di urutan di *chromosome* keempat. Jadi *parent* pertama dipotong di posisi empat kemudian parent kedua akan dipotong dari sisa *parent* pertama. Untuk *Multi-point Crossover* dapat dilihat di Gambar 5.



Gambar 5. *Multi-point Crossover*

Multi-point Crossover adalah improvisasi dari *One-point Crossover* yang beberapa potongan di antara urutan *chromosomes*. Pada Gambar 4 terjadi pemotongan di urutan kedua, keempat dan keenam. Proses pemilihan pemotongan yang akan dihasilkan dari *parent* pertama atau kedua dilakukan secara *random*. Pemotongan tersebut menghasilkan *offspring* yang bisa dilihat di Gambar 5. Untuk proses *crossover* lainnya yaitu *Order Crossover* dapat dilihat di Gambar 6.



Gambar 6. *Order Crossover*

Penjelasan dari Gambar 6 sebagai berikut:

1. Memilih posisi awal gen secara *random* dari *parent* 1.
2. Urutan gen yang sudah dipilih dari *parent* 1 akan dimasukkan ke *block offspring.*

3. *Block offspring* yang masih kosong akan diisi secara berurutan dari kiri ke kanan yang diperoleh dari *parent* 2.

e. Mutation

Mutation dibuat untuk menanggulangi ketika hasil dari *crossover* belum menghasilkan hasil yang maksimal. *Mutation* berguna untuk mengubah bagian kecil yang ada *offspring* untuk memenuhi *requirements* yang diinginkan. Diharapkan pada proses *mutation* ini *offspring* sudah siap untuk mengganti individu yang memiliki nilai *fitness* terbesar.

f. *Worst Replacement*.

Proses penggantian individu yang memiliki nilai *fitness* paling banyak akan digantikan oleh *offspring.* Pada proses ini akan dilakukan terus menurus sampai pada suatu populasi berisi individu yang terbaik.

3. HASIL DAN PEMBAHASAN

Pengujian yang dilakukan ini adalah menganalisa performa mengenai nilai *fitness* dan waktu eksekusi di setiap *crossover.* Nilai *fitness* dihitung dari banyaknya jumlah pelanggaran pada *soft constraint. Hard constraint* tidak hitung dikarenakan harus terpenuhi tanpa terjadi pelanggaran. Semakin kecil nilai *fitness* maka semakin baik solusi penjadwalan yang dihasilkan karena semakin sedikit *soft constraint* yang dilanggar. Sedangkan untuk waktu eksekusi dihitung selisih waktu mulai proses algoritma dengan selesai proses algoritma. Semakin kecil selisih waktu maka semakin cepat proses dari algoritma. Dataset yang akan digunakan untuk pengujian berjumlah 21 yang memiliki keunikan setiap dataset.

Pada penilitian ini untuk *tools* yang digunakan menggunakan bahasa pemrograman PHP. Untuk menghitung nilai *fitness*, jika *chromosoms* yang berisi mata kuliah, hari, waktu dan ruangan melanggar ketentuan *soft constraints* akan dikenakan pinalti. Berikut adalah contoh perhitungan nilai *fitness* untuk *soft constraint* yang melanggar pada *room capacity* dengan ditunjukkan Gambar 7 untuk *chromosom* 1.

c1	1	2	rB	c1	0	0	rB
c1	0	1	rB	c2	2	1	rB
c2	2	5	rB	c4	1	0	rB
c4	2	0	rB	c4	0	2	rB
c5	1	4	rB	c5	1	1	rB

Gambar 7. *Chromosom* 1

Mata kuliah c1 mempunyai jumlah 3 kelas. Kemudian pada kelas pertama di c1 dialokasikan di hari ke-1, waktu ke-2 dan ruangan rB dengan kapasitas 200 dan jumlah mahasiswa yang mengambilnya berjumlah 30. Persamaan untuk *Room Capacity* dapat dilihat pada persamaan (1)

$$\forall\, X_{i,j} = C_k \in X$$

$$f_1(X) = \begin{cases} std_k - cap_j, if\ std_k > cap_j \\ 0, \text{ otherwise.} \end{cases} \quad (1)$$

$X_{i,j}$: *Course* yang ditempatkan di *period* t_i dan *room* r_j
std_i: Mahasiswa yang mengikuti *course* c_i.
cap_j: Kapasitas *room* r_j

Nilai *fitness* untuk *chromosom* 1 untuk *room capacity* adalah sebanyak 0 karena $std > cap$. Begitu seterusnya untuk *constraint* lainnya dan *chromosom* lainnya.

Pada Tabel 1 adalah perbandingan untuk nilai *best fitness* dari *One-Point Crossover, Multi-Point Crossover* dan *Order Crossover*. Dari tabel tersebut terlihat bahwa *One-Point Crossover* memiliki nilai *fitness* yang paling kecil pada dataset 2, 4, 8, 12, 16, 17, 18. Untuk *Multi-Point Crossover* memiliki data nilai *fitness* yang paling kecil pada dataset nomer 3, 6, 7, 15, 20. Sedangkan untuk *Order Crossover* mempunyai nilai *fitness* paling kecil pada dataset nomer 5, 9, 10, 13 dan 14. Dapat disimpulkan bahwa *One-Point Crossover* memiliki jumlah paling banyak berdasarkan pada nilai *fitness.* Sedangkan *Multi-Point Crossover* dan *Order Crossover* mempunyai jumlah yang sama dalam hal nilai *fitness.* Ini berarti bahwa *One-Point Crossver* adalah crossover yang paling optimal dalam mendapatkan individu dengan nilai *fitness* yang paling kecil.

Tabel 1. Perbandingan nilai *best fitness* untuk *fitness* antara *One-Point Crossover, Multi-Point Crossover* dan *Order Crossover*

Dataset	*Crossover*		
	One-Point (s)	*Multi-Point*	*Order*
1	0	0	0
2	**75**	93	83
3	127	**113**	124
4	**117**	127	125
5	419	424	**405**
6	133	**120**	131
7	125	**123**	126
8	**129**	134	147
9	167	162	**161**
10	106	114	**97**
11	0	0	0
12	**396**	402	406
13	154	158	**145**
14	128	128	**127**
15	119	**111**	123
16	**129**	130	130
17	**139**	150	142
18	**144**	150	155
19	**119**	123	121
20	143	**133**	153
21	156	162	**150**

Pada Tabel 2 adalah perbandingan untuk nilai rata-rata untuk *best fitness* dari *One-Point Crossover, Multi-Point Crossover* dan *Order Crossover*. Rata-rata diambil dari 10 kali percobaan yang sudah dilakukan untuk masing-masing *crossover*. Terlihat pada *One-Point Crossover* memiliki rata-rata nilai *fitness* paling kecil pada dataset nomer 1, 2, 6, 8, 9, 11, 13, 15 dan 17. Untuk *Multi-Point* memiliki rata-rata nilai *fitness* paling kecil pada dataset nomer 5, 7, 11, 12, 14, 16, 19 dan 20. Sedangkan untuk *Order Crossover* paling kecil pada dataset nomer 3, 4, 10, 18 dan 21. Dapat disimpulkan bahwa *One-Point Crossover* memiliki jumlah paling banyak dalam hal jumlah nilai rata-rata *fitness* sejumlah 9. Sedangkan untuk *Multi-Point Crossover* sejumlah 8 dan *Order Crossover* sejumlah 5. Ini menandakan bahwa *One-Point Crossover* adalah jenis *crossover* yang paling optimal dilihat dari banyaknya rata-rata nilai *fitness* dibandingkan dengan *crossover* lainnya.

Tabel 2. Perbandingan rata-rata untuk nilai *fitness* antara *One-Point Crossover, Multi-Point Crossover* dan *Order Crossover* dalam 10 kali percobaan

Dataset	Crossover		
	One-Point	Multi-Point	Order
1	**0**	0.6	1.2
2	**103.2**	110.1	104.5
3	139.9	139.6	**135.7**
4	140.1	138	**136.9**
5	495.5	**459.7**	518.2
6	**153.3**	155.1	155.6
7	154.5	**144.0**	151.7
8	**151.6**	158.5	160.8
9	**186.2**	190.7	186.5
10	129.7	129.0	**123.0**
11	**0**	**0**	0.1
12	441.5	**437.0**	438.8
13	**166.9**	168.7	174.3
14	140.8	**138.7**	142.7
15	**133.6**	137.0	136.8
16	148.4	**147.2**	149.9
17	**157.1**	165.5	166.6
18	180.3	171.5	**171.2**
19	130	**128.9**	136.4
20	159.3	**157.4**	164.3
21	187	174.8	**164.3**

Pada Tabel 3 adalah perbandingan nilai rata-rata untuk waktu eksekusi dari *One-Point Crossover, Multi-Point Crossover* dan *Order Crossover*. Terlihat jumlah rata-rata waktu eksekusi paling kecil dari *One-Point Crossover* dibandingkan jenis lainnya adalah pada dataset nomer 3, 6, 8, 9. Untuk *Multi-Point crossover*, paling kecil pada dataset 1, 11 sampai 21. Untuk *Order Crossover* ada di dataset 2, 4, 5, 7. Dapat disimpulkan bahwa *Multi-Point Crossover* sangat superior dalam hal kecepatan waktu eksekusi daripada jenis *Crossover* lainnya. Ini dibuktikan dengan dataset 1, 11 ,12, 13, 14, 15, 16, 17, 18, 19, 20, 21 yang mempunyai waktu eksekusi paling cepat dibandingkan dengan *crossover* lainnya yang diujikan yaitu *One-Point Crossover* dan *Order Crossover*

Tabel 3. Perbandingan rata-rata untuk waktu eksekusi (detik) antara *One-Point Crossover, Multi-Point Crossover* dan *Order Crossover* dalam 10 kali percobaan

Dataset	*Crossover*		
	One-Point	*Multi-Point*	*Order*
1	26	**24.2**	26.5
2	92	98.7	**84.8**
3	**79.8**	95.4	80.9
4	71.1	94.9	**70.7**
5	55.6	66.2	**34.8**
6	**94.8**	134.9	112.5
7	186.3	175.6	**142.6**
8	**77.9**	97.6	166.2
9	**77.4**	89.3	78.9
10	102	103.2	118.1
11	12.8	**7.9**	14.5
12	116.2	**96.2**	109.3
13	83	**74.9**	89.6
14	83.6	**72**	83.1
15	87.9	**69.1**	86.2
16	130.8	**95.6**	132.5
17	109.8	**89.8**	110.4
18	53.1	**41.2**	53.1
19	97	**91.3**	98.5
20	150.3	**130.2**	154.2
21	123.9	**106.4**	121

4. KESIMPULAN

Kesimpulan yang didapat dari hasil pengujian yang sudah dilakukan adalah penggunaan semua jenis *crossover* dapat digunakan untuk menyelesaikan permasalahan penjadwalan mata kuliah. Untuk jenis *crossover* yang paling optimal dalam hal sedikit terjadi pelanggaran yaitu *One-Point Crossover*. Karena nilai rata-rata yang paling kecil didapatkan pada semua dataset pada *crossover* tersebut sejumlah 9 dataset. Sedangkan untuk waktu eksekusi yang paling cepat adalah *Multi-Point Crossover*. *Multi-Point Crossover* memiliki waktu eksekusi paling cepat di 12 datasets.

Pemilihan jenis *crossover* untuk menyelesaikan masalah penjadwalan dapat memilih *One-Point Crossover* jika berfokus pada nilai pelanggaran atau *fitness.* Sedangkan *Multi-Point Crossover* sebaiknya digunakan jika berfokus pada kecepatan waktu eksekusi.

5. SARAN

Dalam penilitian yang sudah dilakukan, peneliti menyarankan untuk dilakukan penelitian lanjutan mengenai perbandingan *crossover* di studi kasus yang berbeda dalam hal lingkup optimasi. Peneliti juga menyarankan untuk melakukan *improvement* pada *One-Point Crossover* menggunakan konsep *Multi Parent* yang diharapkan dapat lebih banyak menurunkan nilai *fitness* atau tingkat pelanggaran yang terjadi terhadap *constraint* di Algoritma Genetika.

DAFTAR PUSTAKA

[1] G. Panchal and D. Panchal, "Solving NP hard Problems using Genetic Algorithm," vol. 6, no. 2, pp. 1824–1827, 2015.

[2] V. Rohini and A. M. Natarajan, "Comparison of genetic algorithm with Particle Swarm Optimisation, ant colony optimisation and Tabu search based on university course scheduling system," *Indian J. Sci. Technol.*, vol. 9, no. 21, 2016, doi: 10.17485/ijst/2016/v9i21/85379.

[3] K. Kumar, "Genetic Algorithm: Review and Application," *Int. J. Tech. Res.*, vol. 2, no. 3, 2013, [Online]. Available: http://www.csjournals.com/IJITKM/PDF 3-1/55.pdf.

[4] T. Song, S. Liu, X. Tang, X. Peng, and M. Chen, "An iterated local search algorithm for the University Course Timetabling Problem," *Appl. Soft Comput. J.*, vol. 68, pp. 597–608, 2018, doi: 10.1016/j.asoc.2018.04.034.

[5] S. Jaengchuea and D. Lohpetch, "A hybrid genetic algorithm with local search and tabu search approaches for solving the post enrolment based course timetabling problem: Outperforming guided search genetic algorithm," *Proc. - 2015 7th Int. Conf. Inf. Technol. Electr. Eng. Envisioning Trend Comput. Inf. Eng. ICITEE 2015*, pp. 29–34, 2015, doi: 10.1109/ICITEED.2015.7408907.

[6] M. W. S. Almeida, J. P. S. Medeiros, and P. R. Oliveira, "Solving the

academic timetable problem thinking on student needs," *Proc. - 2015 IEEE 14th Int. Conf. Mach. Learn. Appl. ICMLA 2015*, pp. 673–676, 2016, doi: 10.1109/ICMLA.2015.184.

[7] J. Liu and W. Li, "Greedy Permuting Method for Genetic Algorithm on Traveling Salesman Problem," *Proc. 2018 IEEE 8th Int. Conf. Electron. Inf. Emerg. Commun. ICEIEC 2018*, pp. 47–51, 2018, doi: 10.1109/ICEIEC.2018.8473531.

[8] Q. Zhu *et al.*, "A novel adaptive hybrid crossover operator for multiobjective evolutionary algorithm," *Inf. Sci. (Ny).*, vol. 345, pp. 177–198, 2016, doi: 10.1016/j.ins.2016.01.046.

[9] T. Informatika, "Pemilihan Crossover Pada Algoritma Genetika Untuk Program Aplikasi Pengenalan," *J. Ilmu Komput. dan Sist. Inf.*, pp. 69–72.

[10] A. M. Fajrin and C. Fatichah, "Multi-parent order crossover mechanism of genetic algorithm for minimizing violation of soft constraint on course timetabling problem," *Regist. J. Ilm. Teknol. Sist. Inf.*, vol. 6, no. 1, 2020, doi: 10.26594/register.v6i1.1663.

[11] S. Abdullah and H. Turabieh, "On the use of multi neighbourhood structures within a Tabu-based memetic approach to university timetabling problems," *Inf. Sci. (Ny).*, vol. 191, pp. 146–168, 2012, doi: 10.1016/j.ins.2011.12.018.

[12] F. A. Zainuddin, F. A. Samad, and D. Tunggal, "A Review of Crossover Methods and Problem Representation of Genetic Algorithm in Recent Engineering Applications Faculty of Mechanical Engineering , Universiti Teknikal Malaysia Melaka , Hang Tuah Jaya , 76100 2 . Principles of Genetic Algorithm," vol. 29, no. 6, pp. 759–769, 2020.

[13] G. A. Akbar and F. Shigeru, "Solving University Course Timetabling Problem Using Multi-Depth Genetic Algorithm," vol. 01001, 2020.

[14] W. Chinnasri, S. Krootjohn, and N. Sureerattanan, "Performance comparison of Genetic Algorithm's crossover operators on University Course Timetabling Problem," *Comput. Technol. Inf. Manag. (ICCM), 2012 8th Int. Conf.*, vol. 2, pp. 781–786, 2012, doi: 10.4156/ijact.vol4.issue20.8.

[15] R. Kumar and Jyotishree, "Blending Roulette Wheel Selection & Rank Selection in Genetic Algorithms," *Int. J. Mach. Learn. Comput.*, vol. 4, no. October, pp. 365–370, 2012, doi: 10.7763/IJMLC.2012.V2.146.