JURNAL
TEKNOLOGI INFORMASI
DAN KOMUNIKASI
Vol. 4 No. 2, Desember 2015 : 76 - 87

# ANALISIS PERFORMANCE ATAS METODE ARITHMETIC CROSSOVER DALAM ALGORITMA GENETIKA

# *PERFORMANCE ANALYSIS OF THE METHOD ARITHMETIC CROSSOVER IN GENETIC ALGORITHM*

Erianto Ongko
Mahasiswa Program Studi Magister Teknik Informatika
Fakultas Ilmu Komputer dan Teknologi Informasi
Universitas Sumatera Utara
Jl. Universitas No. 2A Kampus USU Medan
*erianto_ongko@yahoo.co.id*

Diterima : 22 Juni 2015 Direvisi : 11 Agustus 2015 Disetujui: 2 Desember 2015

## ABSTRAK

Algoritma genetika sering digunakan pada masalah praktis yang berfokus pada pencarian parameter-parameter atau solusi yang optimal. Kelebihan algoritma genetika adalah kemampuan untuk mendapatkan global optima dalam pencarian solusi sehingga sering digunakan dalam optimasi. Salah satu mekanisme yang turut berperan di dalam algoritma genetika adalah proses crossover sebagian dari kromosom induk pertama dengan sebagian kromosom induk kedua lalu menghasilkan kromosom baru. Metode crossover yang akan dianalisis dalam penelitian ini adalah arithmetic crossover dengan studi permasalahan yang digunakan adalah permasalahan Travelling Salesman Problem (TSP). Kromosom offspring (kromosom anak) diperoleh dengan melakukan operasi aritmatika terhadap parent (induk). Algoritma genetika akan berhenti jika sejumlah generasi maksimum tercapai atau level fitness yang ditentukan telah terpenuhi. Tujuan dari penelitian ini adalah mendapatkan hasil analisis performance dari metode arithmetic crossover dengan masalah utama adalah mendapatkan gambaran mengenai kaitan antara jumlah gen di dalam suatu kromosom yang mengalami crossover dengan performance dari algoritma genetika. Hasil penelitian menunjukkan bahwa semakin banyak gen yang mengalami crossover akan meningkatkan performance dari algoritma genetika, yang ditunjukkan dalam bentuk whole arithmetic crossover memiliki performance yang lebih baik daripada simple arithmetic crossover dan simple arithmetic crossover memiliki performance yang lebih baik daripada single arithmetic crossover.

**Kata Kunci:** Analisis Performance, Arithmetic Crossover, Algoritma Genetika, Kromosom, Gen

## *ABSTRACT*

*Genetic algorithms are often used in practical problems that focuses on search parameters or the optimal solution. Excess genetic algorithm is its ability to obtain global optima in the search for a solution that is often used in the optimization. One of the mechanisms that play a role in the genetic algorithm is the crossover portion of the first parent chromosome with most second parent chromosome and produce new chromosomes. Crossover method which will be analyzed in this study is the arithmetic crossover used to study the problems is the problem of Traveling Salesman Problem (TSP). Offspring chromosome (child) is obtained by performing arithmetic operations of the parent. Genetic algorithm will stop when the maximum number of generations is reached or a specified level of fitness has been fulfilled. The purpose of this study is to get the performance analysis of the arithmetic crossover method with the main problem is to get an idea of the link between the number of genes in a*

*chromosome that is experiencing a crossover with the performance of the genetic algorithm. The results showed that the more genes that experiencing crossover will increase the performance of the genetic algorithm, which is shown in the form of whole arithmetic crossover has a better performance than simple arithmetic and simple arithmetic crossover crossover has better performance than a single arithmetic crossover.*

***Keywords:*** *Performance Analysis, Arithmetic Crossover, Genetic Algorithm, Chromosome, Gene*

## PENDAHULUAN

Pada tahun 1975, John Holland, mengemukakan penelitian mengenai komputasi berbasis evolusi yang dituangkan dalam bukunya yang berjudul "*Adaption in Natural and Artificial Systems*". Tujuannya membuat komputer dapat melakukan apa yang terdapat di alam. Sebagai seorang pakar komputer, Holland memfokuskan diri pada manipulasi *string* dalam bentuk *binary* bit. Holland mengemukakan algoritma tersebut sebagai suatu konsep abstrak dari evolusi alam. Tahapan algoritma genetika yang dikemukakan oleh Holland dapat direpresentasikan sebagai suatu tahapan berurutan sebagai suatu bentuk populasi dari kromosom buatan menjadi sebuah populasi baru.[1]

Algoritma genetika adalah algoritma pencarian heuristik yang didasari pada pemikiran mengenai seleksi alam yang terjadi pada proses evolusi dan operasi genetika.[2] Hal lain yang membuat algoritma genetika unggul adalah kemampuannya mendapatkan hasil pencarian *global optima* ketimbang terjebak dalam *local optima*.[3]

Dalam algoritma genetika ada sejumlah operasi yang mendukung keberhasilan algoritma genetika seperti pembangkitan populasi awal, perhitungan *fitness* tiap kromosom, seleksi kromosom, *crossover*, dan mutasi kromosom. Proses *crossover* merupakan proses pembentukan kromosom anak (*offspring*). *Crossover* bertujuan menambah keanekaragaman *string* dalam satu populasi dengan penyilangan antar *string* yang diperoleh dari reproduksi sebelumnya. Terdapat beberapa jenis *crossover* di antaranya adalah *crossover* 1 titik (*single point crossover*), 2 titik (*two point crossover*), dan *arithmetic crossover*.[4] Metode *arithmetic crossover* dapat dibagi menjadi 3 jenis, yaitu *single arithmetic crossover*, *simple arithmetic crossover*, dan *whole arithmetic crossover*.[2] Perbedaan metode *crossover* ini dalam menghasilkan kromosom terbaik akan mempengaruhi kinerja algoritma genetika. Algoritma genetika akan berhenti jika sejumlah generasi maksimum tercapai atau level *fitness* yang ditentukan telah terpenuhi.

Salah satu permasalahan yang dapat diselesaikan dengan menggunakan algoritma genetika adalah permasalahan *Travelling Salesman Problem* (TSP). TSP merupakan persoalan yang mempunyai konsep sederhana dan mudah dipahami. Pada TSP, optimasi yang diinginkan agar ditemukan rute perjalanan terpendek untuk melewati sejumlah kota dengan jalur tertentu, sehingga setiap kota hanya terlewati satu kali dan perjalanan diakhiri dengan kembali ke kota semula.

Lin *et al.*[5], menggunakan algoritma genetika untuk mencari jarak terpendek pada sistem ITS (*Intelligent Transportation System*) di Taiwan dengan menggunakan variasi jumlah gen dan kromosom. Dari penelitian tersebut diketahui bahwa semakin banyak gen dan kromosom, maka solusi optima akan lebih cepat diperoleh. Ada beberapa penelitian lain yang telah dilakukan berkenaan dengan algoritma genetika. Samuel *et al*.[6] membahas bagaimana algoritma genetik menyelesaikan TSP dengan menggunakan metode *order crossover* sebagai teknik rekombinasi dan metode *insertion mutation*

sebagai teknik mutasi yang digunakan pada algoritma genetik. Annies *et al*.[7] menunjukkan bahwa algoritma genetika dapat digunakan untuk menyelesaikan masalah optimasi yang kompleks seperti mencari rute optimum, menggunakan beberapa metode seleksi yaitu *roulette wheel*, *elitism,* dan gabungan antara metode *roulette wheel* dan *elitism*. Ada dua jenis *crossover* yang digunakan yaitu *one cut point crossover* dan *two cut point crossover*.

Nasution[8] membahas analisis penyelesaian TSP menggunakan *partially mapped crossover* dengan menentukan nilai *probabilitas crossover* 20%, 40%, 60%, 80% dan 99%. Deep & Mebrahtu[9] membuat variasi pada *partially mapped crossover* dengan menentukan letak kromosom dalam posisi acak. Kemudian, Al Kasasbeh, *et al*.[10] menambahkan sebuah prosedur baru pada algoritma genetika untuk menyelesaikan TSP yaitu dengan metode *shared neighbour*. Penelitian terbaru yang dilakukan oleh Picek *et al.*[2] yang membandingkan beberapa metode *crossover* di dalam menyelesaikan 24 permasalahan dengan menggunakan 16 metode *crossover*, yang menarik dari hasil penelitian adalah bahwa metode *whole arithmetic crossover* memiliki *performance* yang lebih baik daripada metode *simple arithmetic crossover*, *simple arithmetic crossover* memiliki *performance* yang lebih baik daripada *single arithmetic crossover*.

Penelitian yang dilakukan oleh Picek *et al.*[2] cukup menarik, karena secara luas membandingkan beberapa metode *crossover* yang ada sehingga memberikan sumbangsih yang cukup berarti di dalam perkembangan konsep algoritma genetika. Hal yang belum dibahas di dalam penelitian ini adalah apakah terdapat keterkaitan langsung antara jumlah *gen* yang mengalami *crossover* dan *performance* algoritma genetika. Hal ini dilakukan mengingat terdapat peningkatan *performance* bila dikaitkan dengan jumlah gen yang mengalami *crossover*. Hasil penelitian menunjukkan bahwa *whole arithmetic crossover* memiliki *performance* yang lebih baik dari *simple arithmetic crossover* dan *Simple arithmetic crossover* memiliki *performance* yang lebih baik daripada *single arithmetic crossover*.

Pada metode *whole arithmetic crossover*, seluruh gen pada kromosom *parent* mengalami *arithmetic crossover*, pada metode *simple arithmetic crossover* ditentukan sebuah bilangan acak, kemudian gen pada kromosom *parent* mulai dari titik bilangan acak sampai sepanjang jumlah kromosom akan mengalami *arithmetic crossover*, sedangkan pada metode *single arithmetic crossover* ditentukan sebuah bilangan acak dan hanya gen pada kromosom *parent* yang berada pada titik acak tersebut yang mengalami mutasi.[11]

Adapun masalah yang akan dikaji oleh peneliti adalah menganalisis keterkaitan antara jumlah gen di dalam suatu kromosom yang mengalami *crossover* dengan *performance* dari algoritma genetika dan diharapkan dapat memberikan gambaran mengenai upaya untuk meningkatkan *performance* dari algoritma genetika dengan menentukan jenis metode *arithmetic crossover* yang terbaik.

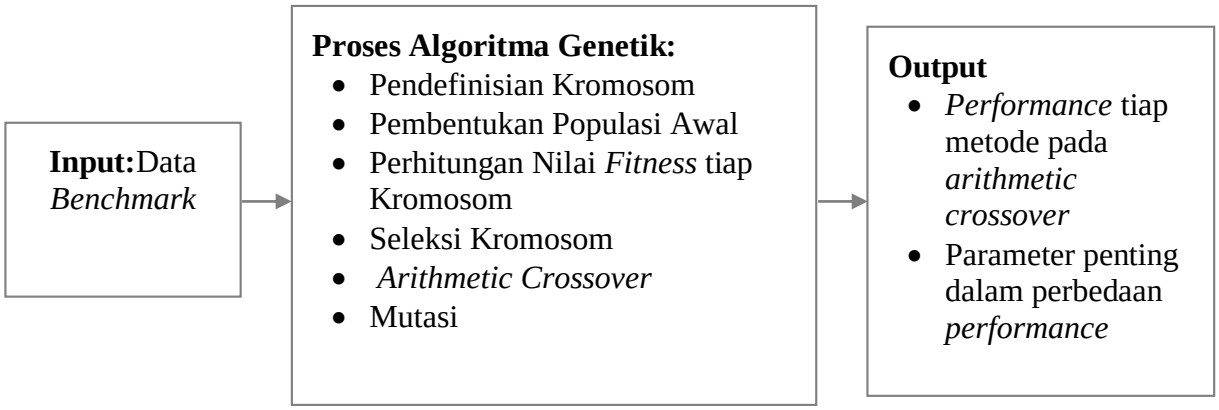
## METODE PENELITIAN

*Travelling Salesman Problem* termasuk ke dalam kelas permasalahan NP (*NonDeterministic Polynomial*) kategori sulit karena memiliki kompleksitas 0 (nol). Permasalahan utama dari TSP adalah bagaimana seorang *salesman* dapat mengatur rute perjalanannya untuk mengunjungi sejumlah kota yang diketahui jarak satu kota dengan kota lainnya sehingga jarak yang ditempuh merupakan jarak minimum di mana seorang *salesman* hanya dapat mengunjungi kota tersebut tepat satu kali.

Salah satu metode yang dapat digunakan di dalam menyelesaikan permasalahan TSP yaitu algoritma genetika. *Crossover* merupakan salah satu aspek penting di dalam algoritma genetika untuk menghasilkan *best fitness*. Terdapat beberapa metode *crossover* yang dapat digunakan. Salah satu metode *crossover* yang dapat digunakan adalah metode *arithmetic crossover*.

Pada penelitian ini dibahas mengenai *performance* dari tiap metode *arithmetic crossover* yang ada, yang terdiri atas *whole arithmetic crossover*, *simple arithmetic crossover*, dan *single arithmetic crossover*.

Data yang digunakan merupakan data *benchmark* yang diambil dari *Travelling Salesman Problem Library* (TSPLIB) dan TSPLIB merupakan *library* data dan permasalahan TSP diambil dari berbagai sumber dan bermacam tipe dari permasalahan TSP. Jenis data *file* TSP yang digunakan sebagai data uji adalah data TSP Simetri yang jarak antara titik I ke titik J sama dengan jarak titik J ke titik I. Adapun data yang digunakan yaitu data *berlin52.TSP*. Pada *file berlin52.TSP*, terdapat kota sebanyak 52 kota. Adapun prosedur kerja yang dilakukan oleh peneliti dari penelitian ini dapat dilihat secara keseluruhan pada Gambar 1.




**Gambar 1**. Metode Penelitian

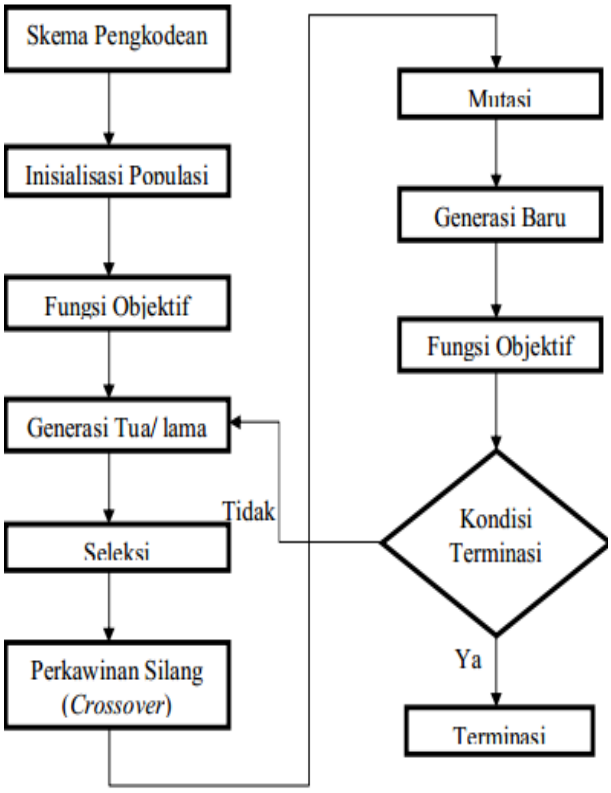
Pada Gambar 1, dapat dilihat bahwa proses penelitian dimulai dari penentuan *input* yang dalam hal ini digunakan data *benchmark* yang sudah ada, yang bersumber dari TSPLIB yaitu *Berlin52.TSP*. Tahapan penelitian yang dilakukan adalah dimulai dari pendefinisian kromosom, pembentukan populasi awal, perhitungan nilai *fitness* tiap kromosom, penyeleksian kromosom, *artihmetic crossover*, dan permutasian. Proses ini akan dilakukan hingga diperoleh *output* berupa *performance* tiap metode pada *artihmetic crossover* dan parameter penting yang menentukan perbedaan *performance*.

## HASIL DAN PEMBAHASAN

### Analisis Proses Algoritma Genetika

Algoritma genetik memberikan suatu pilihan bagi penentuan nilai parameter dengan meniru cara reproduksi genetik, pembentukan kromosom baru serta penyeleksian alami seperti yang terjadi pada makhluk hidup. Fase awal dari algoritma genetika adalah inisialisasi populasi yang menyatakan alternatif solusi. Elemen dari populasi dideskripsikan dalam bentuk deretan bit *string* yang berisi bit 0 atau 1 yang disebut sebagai kromosom. Kemudian langkah selanjutnya adalah menghitung nilai *fitness* berdasarkan gen yang ada pada kromosom dalam tiap

populasi. Berdasarkan atas nilai *fitness* dari tiap koromosom, maka tahapan selanjutnya adalah tahapan seleksi yang berfungsi untuk memilih kromosom yang terpilih sebagai *parent* yang akan menjalani *crossover*. Proses *crossover* yang berjalan dengan beberapa variasi operator *crossover* berperan penting di dalam membentuk kromosom anak (*offspring*) yang juga berperan penting untuk menambah keanekaragaman *string* di dalam suatu populasi. Kromosom selanjutya akan masuk ke dalam tahap mutasi yang berfungsi untuk memastikan keanekaragaman (*diversity*) dari kromosom dalam suatu populasi tetap terjaga, untuk menghindari terjadinya konvergensi prematur yang berujung pada terjadinya solusi *local optima*.[4] . Proses yang ada di dalam Algoritma Genetik dapat diilustrasikan dalam diagram pada Gambar 2.




Sumber: Goldberg, 1989

**Gambar 2**. Gambar Diagram Alir Algoritma

**Analisis Pembangkitan Populasi Awal**

Pada penelitian ini data yang digunakan adalah *Berlin52.TSP*. Di mana terdapat 52 kota yang ada dan seorang *salesman* harus mengunjungi 52 kota tersebut dengan jarak yang terpendek. Jumlah populasi yang dibangkitkan adalah sebanyak 10, di mana di dalam penelitian ini tiap kromosom memiliki 51 gen yang mewakili kota yang harus dikunjungi.

**Analisis Proses Penghitungan Nilai *Fitness***

Untuk dapat menghitung nilai *fitness* maka kita harus menghitung nilai fungsi objek terlebih dahulu. Fungsi objektif di dalam permasalahan ini adalah merupakan total jarak perjalanan yang dilalui oleh seorang *salesman*. Adapun jarak antara 1 kota dengan kota lainnya dapat dihitung dengan menggunakan perhitungan *euclidean distance*. Adapun persamaan untuk *euclidean distance* dapat dilihat pada Persamaan 1.[13]

$$d_{(i,j)} = \sqrt{(x_i - x_j) + (y_i - y_j)^2} \quad \text{.....(1)}$$

Dimana:
$x_i$ = Koordinat x kota i
$x_j$ = Koordinat x kota j
$y_i$ = Koordinat y kota i
$y_j$ = Koordinat y kota j

Setelah menghitung nilai fungsi objektif, maka nilai *fitness* dapat dihitung dengan menggunakan Persamaan 2.[1]

Fitness = 1 / (1+Fungsi_Objektif)....(2)

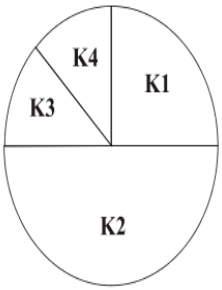
**Analisis Proses Seleksi**

Proses seleksi yang digunakan di dalam penelitian ini adalah *roulette wheel selection*. *Roulette wheel selection* adalah metode seleksi yang paling sederhana. Pada metode ini semua kromosom (individu) di dalam suatu populasi adalah ditempatkan pada *roulette wheel* sesuai dengan nilai *fitness* mereka. Besarnya ukuran tiap segmen di dalam *roulette* adalah sebanding

dengan nilai *fitness* dari tiap individu. Semakin besar nilai *fitness* maka semakin besar pula ukuran segmen di dalam *roulette wheel*, kemudian *roulette wheel* diputar. Individu yang sesuai dengan segmen pada *roulette wheel* ketika berhenti yang akan dipilih.[14]

Metode *roulette wheel selection* dapat dilihat pada Gambar 3.

| Kromosom | Nilai fitness |
|---|---|
| K1 | 1 |
| K2 | 2 |
| K3 | 0.5 |
| K4 | 0.5 |
| Jumlah | 4 |




Sumber: Kumar, 2012

**Gambar 3**. Metode *Roulette Wheel Selection*

### Analisis Proses *Crossover*

Metode *crossover* yang digunakan di dalam penelitian ini adalah metode *arihtmetic crossover* yang terdiri-dari: *single arithmetic crossover*, *simple arithmetic crossover*, dan *whole arithmetic crossover*.

Pada *single arithmetic crossover,* pindah silang terjadi pada salah satu gen yang posisinya ditentukan dengan cara membangkitkan suatu bilangan acak. Pada posisi gen yang ditentukan, nilai gen akan ditentukan melalui operasi aritmatika terhadap nilai gen dari *parent* menurut persamaan 3.[11] Adapun operasi aritmatika pada *single arithmetic crossover* dapat dilihat pada Persamaan 3 dan Tabel 1.

$$\text{Child} = \left\langle x_1,\ldots,x_k,\alpha.y_k+(1-\alpha).x_k,\ldots,x_n \right\rangle \quad \ldots\ldots(3)$$

**Tabel 1**. *Single Arithmetic Crossover*

| | |
|---|---|
| Kromosom *Parent* 1 | 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 |
| Kromosom *Parent* 2 | 0.2 0.3 |
| Bilangan Acak | 8 |
| A | 0.5 |
| Kromosom *Offspring* 1 | 0.1 0.2 0.3 0.4 0.5 0.6 0.7 **0.5** 0.9 |
| Kromosom *Offspring* 2 | 0.3 0.2 0.3 0.2 0.3 0.2 0.3 **0.5** 0.3 |

Pada *simple arithmetic* crossover, Tentukan bilangan *random* sebagai titik potong antara 0 sampai sepanjang kromosom pada masing-masing parent. Untuk gen pada kromosom *offspring* untuk batas sebelum titik potong disalin dari gen pada kromosom *parent*. Untuk gen setelah titik potong, gen yang ada dibentuk dari operasi aritmatika pada gen dari kromosom *parent* dengan persamaan seperti pada persamaan 4.[2] Illustrasi dari proses *simple arithmetic crossover* dapat dilihat pada Tabel 2.

$$\text{Child}=\left\langle x_1,\ldots,x_k,\alpha.y_{k+1}+(1-\alpha).x_{k+1},\ldots,\alpha.y_n+(1-\alpha).x_n \right\rangle \quad \ldots\ldots(4)$$

**Tabel 2**. *Simple Arithmetic Crossover*

| | |
|---|---|
| Kromosom *Parent* 1 | 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 |
| Kromosom *Parent* 2 | 0.3 0.2 0.3 0.2 0.3 0.2 0.3 0.2 0.3 |
| Bilangan Acak | 6 |
| α | 0.5 |
| Kromosom *Offspring* 1 | 0.1 0.2 0.3 0.4 0.5 0.6 **0.5 0.5 0.6** |
| Kromosom *Offspring* 2 | 0.3 0.2 0.3 0.2 0.3 0.2 **0.5 0.5 0.6** |

Pada *whole arithmetic crossover*, gen pada kromosom *offspring* diperoleh dari hasil operasi aritmatika gen pada kromosom *parent*, di mana proses aritmatika yang dilakukan sesuai dengan persamaan 5.[11] Illustrasi dari proses *whole arithmetic crossover* dapat dilihat pada Tabel 3.

$$\text{Child}=\left\langle \alpha.\bar{x}+(1-\alpha).\bar{y} \right\rangle \quad \ldots\ldots\ldots\ldots(5)$$

**Tabel 3.** *Whole Arithmetic Crossover*

| | |
|---|---|
| Kromosom *Parent* 1 | 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 |
| Kromosom *Parent* 2 | 0.3 0.2 0.3 0.2 0.3 0.2 0.3 0.2 0.3 |
| α | 0.5 |
| Kromosom *Offspring* | **0.2 0.2 0.3 0.3 0.4 0.4 0.5 0.5 0.6** |

**Analisis Proses Mutasi**

Mutasi adalah proses untuk mengubah gen di dalam sebuah kromosom. Mutasi dilakukan setelah proses *crossover* dilakukan. Mutasi mengubah *offspring* baru dengan mengubah *bit* 1 menjadi 0 atau *bit* 0 menjadi 1. Mutasi dapat terjadi pada setiap posisi *bit* di dalam *string* dengan beberapa probabilitas yang umumnya sangat kecil. Mutasi adalah dimaksudkan untuk mencegah hasil pencarian mengarah pada keadaan *local optima* di dalam sebuah area pencarian.[15]

**Analisis Proses Pengujian**

Pada penelitian ini akan ditampilkan hasil penilaian performansi untuk tiap metode *arithmetic crossover* di dalam menyelesaikan permasalahan TSP. Pengukuran performansi akan dilakukan terhadap metode *whole arithmetic crossover*, *simple arithmetic crossover*, dan *single arithmetic crossover*. Nilai performansi akan dinyatakan di dalam bentuk nilai *average best distance* yang merupakan nilai rata-rata untuk jarak terpendek untuk tiap pengujian dan juga *average best fitness* yang merupakan nilai rata-rata untuk *best fitness* untuk tiap pengujian. Nilai *best fitness* diperoleh dari hasil pembagian 1 dengan nilai *best distance* sehingga semakin kecil nilai *best distance* akan semakin besar pula nilai *best fitness*, dengan demikian semakin besar nilai *average best fitness* berarti semakin baik pula performansi dari suatu metode *arithmetic crossover*. Hasil pengujian yang dilakukan oleh peneliti akan disampaikan dalam bentuk tabel.

**Hasil Pengujian untuk 100 Generasi dengan Nilai PC=0.25**

Pengujian dilakukan sebanyak 100 generasi dengan nilai *probability crossover* 0.25 dan nilai *mutation rate* sebesar 0.1 serta nilai α sebesar 0.5 untuk melihat nilai *best fitness* dari masing-masing metode *arithmetic crossover* dengan mengambil nilai rata-rata pada masing-masing metode *arithmetic crossover*. Hasil Pengujian untuk 100 Generasi dengan nilai PC = 0.25 dapat dilihat pada Tabel 4.

**Tabel 4**. Hasil Pengujian untuk 100 Generasi dengan Nilai PC = 0.25

| Metode *Arithmetic Crossover* | *Average Best Fitness* | *Average Best Distance* |
|---|---|---|
| *Whole Arithmetic* | 0.00004435 | 22605.9780 |
| *Simple Arithmetic* | 0.00004194 | 23893.0054 |
| *Single Arithmetic* | 0.00004105 | 24410.1405 |

Pada Tabel 4, terlihat bahwa untuk pengujian dengan menggunakan PC sebesar 0.25 untuk 100 generasi, metode *Whole Arithmetic* merupakan metode yang memiliki *average best fitness* terbaik, posisi kedua ditempati oleh *simple arithmetic*, dan posisi ketiga ditempati oleh *single arithmetic*. Sehingga terlihat bahwa semakin banyak gen yang mengalami *crossover* akan memberikan hasil *fitness* yang semakin baik.

**Hasil Pengujian untuk 100 Generasi dengan Nilai PC=0.5**

Pengujian dilakukan sebanyak 100 generasi dengan nilai *probability crossover* 0.5 dan nilai *mutation rate* sebesar 0.1 serta nilai α sebesar 0.5 untuk melihat nilai *best fitness* dari masing-masing metode *arithmetic crossover* dengan mengambil nilai rata-rata pada masing-masing metode *arithmetic crossover*. Hasil Pengujian untuk

100 Generasi dengan nilai PC = 0.5 dapat dilihat pada Tabel 5.

**Tabel 5**. Hasil Pengujian untuk 100 Generasi dengan Nilai PC = 0.5

| Metode *Arithmetic Crossover* | *Average Best Fitness* | *Average Best Distance* |
|---|---|---|
| *Whole Arithmetic* | 0.00004664 | 21460.4936 |
| *Simple Arithmetic* | 0.00004302 | 23300.7059 |
| *Single Arithmetic* | 0.00004165 | 24103.0330 |

Berdasarkan Tabel 5 diperoleh hasil bahwa hasil *fitness* dari metode *whole arithmetic crossover* pada nilai PC sebesar 0.5 untuk 100 generasi adalah lebih baik dari metode *simple arithmetic crossover* dan *simple arithmetic crossover* memiliki nilai *fitness* yang lebih baik daripada *single arithmetic crossover*.

**Hasil Pengujian untuk 300 Generasi dengan Nilai PC=0.25**

Pengujian dilakukan sebanyak 300 generasi dengan nilai *probability crossover* 0.25 dan nilai *mutation rate* sebesar 0.1 serta nilai α sebesar 0.5 untuk melihat nilai *best fitness* dari masing-masing metode *arithmetic crossover* dengan mengambil nilai rata-rata pada masing-masing metode *arithmetic crossove*r. Hasil Pengujian untuk 300 Generasi dengan nilai PC = 0.25 dapat dilihat pada Tabel 6.

**Tabel 6**. Hasil Pengujian untuk 300 Generasi dengan Nilai PC = 0.25

| Metode *Arithmetic Crossover* | *Average Best Fitness* | *Average Best Distance* |
|---|---|---|
| *Whole Arithmetic* | 0.00004624 | 21657.9058 |
| *Simple Arithmetic* | 0.0000456 | 21937.7414 |
| *Single Arithmetic* | 0.00004487 | 22337.9496 |

Berdasarkan Tabel 6, diperoleh hasil bahwa hasil *fitness* dari metode *whole arithmetic crossover* pada nilai PC sebesar 0.25 untuk 300 generasi adalah lebih baik dari metode *simple arithmetic crossover* dan *simple arithmetic crossover* memiliki nilai *fitness* yang lebih baik daripada *single arithmetic crossover*.

**Hasil Pengujian untuk 300 Generasi dengan Nilai PC=0.5**

Pengujian dilakukan sebanyak 300 generasi dengan nilai *probability crossover* 0.5 dan nilai *mutation rate* sebesar 0.1 serta nilai α sebesar 0.5 untuk melihat nilai *best fitness* dari masing-masing metode *arithmetic crossover* dengan mengambil nilai rata-rata pada masing-masing metode *arithmetic crossover*. Hasil Pengujian untuk 300 Generasi dengan nilai PC = 0.5 dapat dilihat pada Tabel 7.

**Tabel 7**. Hasil Pengujian untuk 300 Generasi dengan Nilai PC = 0.5

| Metode *Arithmetic Crossover* | Average *Best Fitness* | Average *Best Distance* |
|---|---|---|
| *Whole Arithmetic* | 0.000046303 | 21599.3185 |
| *Simple Arithmetic* | 0.00004576 | 21885.5975 |
| *Single Arithmetic* | 0.00004407 | 22730.1067 |

Berdasarkan Tabel 7, diperoleh hasil bahwa hasil *fitness* dari metode *whole arithmetic crossover* pada nilai PC sebesar 0.5 untuk 300 generasi adalah lebih baik dari metode *simple arithmetic crossover* dan *simple arithmetic crossover* memiliki nilai *fitness* yang lebih baik daripada *single arithmetic crossover*.

**Hasil Pengujian untuk 500 Generasi dengan Nilai PC=0.25**

Pengujian dilakukan sebanyak 500 generasi dengan nilai *probability crossover* 0.25 dan nilai *mutation rate* sebesar 0.1 serta nilai α sebesar 0.5 untuk melihat nilai *best fitness* dari masing-masing metode *arithmetic crossover* dengan mengambil nilai rata-rata pada masing-masing metode *arithmetic crossover*. Hasil Pengujian untuk 500 Generasi dengan nilai PC = 0.25 dapat dilihat pada Tabel 8.

**Tabel 8**. Hasil Pengujian untuk 500 Generasi dengan Nilai PC = 0.25

| Metode *Arithmetic Crossover* | *Average Best Fitness* | *Average Best Distance* |
|---|---|---|
| *Whole Arithmetic* | 0.0000472 | 21191.0488 |
| *Simple Arithmetic* | 0.0000467 | 21427.2217 |
| *Single Arithmetic* | 0.00004459 | 22488.0355 |

Berdasarkan Tabel 8, diperoleh hasil bahwa hasil *fitness* dari metode *whole arithmetic crossover* pada nilai PC sebesar 0.25 untuk 500 generasi adalah lebih baik dari metode *simple arithmetic crossover* dan *simple arithmetic crossover* memiliki nilai *fitness* yang lebih baik daripada *single arithmetic crossover*.

**Hasil Pengujian untuk 500 Generasi dengan Nilai PC=0.5**

Pengujian dilakukan sebanyak 500 generasi dengan nilai *probability crossover* 0.5 dan nilai *mutation rate* sebesar 0.1 serta nilai α sebesar 0.5 untuk melihat nilai *best fitness* dari masing-masing metode *arithmetic crossover* dengan mengambil nilai rata-rata pada masing-masing metode *arithmetic crossover*. Hasil Pengujian untuk 500 Generasi dengan nilai PC = 0.5 dapat dilihat pada Tabel 9.

**Tabel 9**. Hasil Pengujian untuk 500 Generasi dengan Nilai PC = 0.5

| Metode *Arithmetic Crossover* | *Average Best Fitness* | *Average Best Distance* |
|---|---|---|
| *Whole Arithmetic* | 0.00005430 | 20303.3926 |
| *Simple Arithmetic* | 0.00004716 | 21232.9354 |
| *Single Arithmetic* | 0.00004574 | 21897.273 |

Berdasarkan Tabel 9, diperoleh hasil bahwa hasil *fitness* dari metode *whole arithmetic crossover* pada nilai PC sebesar 0.5 untuk 500 generasi adalah lebih baik dari metode *simple arithmetic crossover* dan *simple arithmetic crossover* memiliki nilai *fitness* yang lebih baik daripada *single arithmetic crossover*.

**Pembahasan**

Berdasarkan hasil pengujian yang dilakukan oleh peneliti seperti yang terllihat pada Tabel 4, 5, 6, 7, 8, dan 9 dapat dilihat bahwa terdapat peningkatan nilai *fitness* yang diperoleh seiring dengan pertambahan jumlah generasi yang ada. Secara umum, nilai *fitness* pada masing-masing metode *arithmetic crossover* semakin tinggi seiring dengan pertambahan jumlah generasi. Nilai *fitness* untuk 300 generasi akan lebih baik daripada nilai *fitness* untuk 100 generasi, demikian juga nilai *fitness* untuk 500 generasi akan lebih baik daripada nilai *fitness* untuk 300 generasi.

Hasil penelitian juga menunjukkan bahwa Semakin banyak gen yang terlibat dalam proses *crossover* akan meningkatkan keanekaragaman gen dalam populasi, yang dapat meningkatkan kinerja algoritma genetika. Hal ini telihat dari nilai *fitness*

untuk *whole arithmetic crossover* yang lebih baik daripada *simple arithmetic crossover* dan *simple arithmetic crossover* yang memiliki nilai *fitness* yang lebih baik daripada *single arithmetic crossover*.

Berdasarkan hasil analisis yang dilakukan oleh peneliti, diperoleh bahwa proses yang terjadi pada algoritma genetika pada hakikatnya adalah mengikuti proses evolusi biologis. Variasi atau keanekaragaman gen akan meningkatkan *performance* dari algoritma genetika, karena pada hakikatnya terjadi peningkatan kualitas gen yang baik. Di sisi lain, dapat dipahami bahwa pertambahan jumlah generasi memberikan kesempatan yang lebih besar untuk meningkatkan *performance* algoritma genetika, karena pada hakikatnya proses di dalam algoritma genetika akan terus berulang sampai dicapai *performance* yang diharapkan.

Hasil penelitian yang dilakukan oleh peneliti diharapkan dapat memberikan sumbangsih pemikiran kepada Kemkominfo, untuk mendapatkan pemilihan metode *crossover* yang terbaik di dalam pemanfaatan algoritma genetika. Pemanfaatan algoritma genetika, cukup potensial bagi Kemkominfo, karena pada prinsipnya algoritma genetika dapat digunakan untuk mencari solusi yang optimal.

## SIMPULAN

Berdasarkan hasil penelitian yang dilakukan oleh peneliti maka beberapa kesimpulan yang dapat ditarik oleh peneliti adalah sebagai berikut.

1. Hasil penelitian menunjukkan bahwa Semakin banyak gen yang terlibat dalam proses *crossover* akan meningkatkan keanekaragaman gen dalam populasi, yang dapat meningkatkan kinerja algoritma genetika. Peningkatan kinerja algoritma genetika dapat lihat dari nilai *fitness* untuk pengujian pada metode *arithmatic crossover*. Dimana terdapat 3 metode yang diuji performansinya yaitu *whole arithmatic crossover*, *simple arithmatic crossover*, dan *single arithmatic crossover*. Secara umum, jumlah gen yang mengalami *crossover* pada *whole arithmatic crossover* akan lebih besar daripada *simple arithmatic crossover* dan *simple arithmatic crossover* akan memiliki gen yang mengalami *crossover* yang lebih banyak daripada *single arithmatic crossover*. Berdasarkan hasil pengujian *whole arithmetic crossover* memiliki nilai *average best fitness* yang lebih baik daripada *simple arithmetic crossover* dan *simple arithmetic crossover* yang memiliki nilai *average best fitness* yang lebih baik daripada *single arithmetic crossover*. Hal ini terjadi pada pengujian dengan menggunakan nilai *probability crossover* sebesar 0.25 dan 0.5 pada pengujian dengan menggunakan 100, 300, dan 500 generasi.
2. Nilai *average best fitness* pada masing-masing metode *arithmetic crossover* semakin tinggi seiring dengan pertambahan jumlah generasi. Nilai *average best fitness* untuk 300 generasi akan lebih baik daripada nilai *average best fitness* untuk 100 generasi, demikian juga nilai *average best fitness* untuk 500 generasi akan lebih baik daripada nilai *average best fitness* untuk 300 generasi. Secara umum peningkatan nilai *fitness* seiring dengan pertambahan jumlah generasi terjadi karena dengan jumlah generasi yang semakin banyak maka peluang untuk menghasilkan generasi baru yang memiliki nilai *fitness* yang lebih baik juga akan menjadi semakin besar. Hasil pengujian menunjukkan bahwa meskipun nilai *best fitness* yang diambil dari pengujian sebanyak 10 kali untuk jumlah generasi sebanyak 100, 300, dan 500 tidak dicapai pada generasi maksimal (Generasi ke-n), seperti

misalnya pengujian dengan nilai PC sebesar 0.5 dan jumlah generasi sebanyak 500 nilai *best fitness* justru dicapai pada generasi ke-26. Namun, secara *average best fitness* diperoleh bahwa nilai *average best fitness* dari pengujian dengan 500 generasi akan lebih baik daripadai 300 generasi dan *average best fitness* dari pengujian dengan 300 generasi akan lebih baik daripada 100 generasi.

**DAFTAR PUSTAKA**

[1]Negnevitsky, Michael. 2005. *Artificial Intteligence-A Guide to Intelligent Systems*. Addison Wesley: Edinburg

[2]Picek, Stjepan, Jakobovic, Domagoj and Gloub, Marin. 2013. On the Recombination Operator in The Real-Code Genetic Algorithms, *2013 IEEE Congress on Evolutionary Computation*, pp. 3103-3110

[3]Russell, Stuart And Norvig, Peter. 2009. *Artificial Intelligence: A Modern Approach*. 3rd Edition. Pearson Education Limited: London

[4]Konar, Amit. 2005. *Computational Intelligence Principles, Techniques, and Applications*. Springer: Calcutta, India

[5]Lin, Chu Hsing, Yu, Jui Ling, Liu, Jung Chun, Lai, Wei Shen and Ho, Chia Han. 2009. Genetic Algorithm For Shortest Driving Time in Intelligent Transportation System. *Internation Journal of Hybrid Information Technology***2**(1): 21-30

[6]Samuel, Lukas, Toni, A. dan Willi, Y.. 2005. Penerapan Algoritma Genetika Untuk Salesman Problem Dengan Menggunakan Metode Order Crossover dan Insertion Mutation. *Prosiding Seminar Nasional Aplikasi Teknologi Informasi 2005 (SNATI 2005)*, pp. I-1 – I-5

[7]Annies, Hannawati, Thing, Eleazar. 2002. Pencarian Rute Optimum menggunakan algoritma genetika. *Jurnal Teknik Elektro Fakultas Teknologi Industri, Jurusan Teknik Elektro, Universitas Kristen Petra***2**(2)*:* 78-83

[8]Nasution, K. 2012. Analisis Pemilihan *Partially Mapped Crossover* Algoritma Genetika pada Penyelelesaian *Travelling Salesman Problem*. Tesis. Universitas Sumatera Utara

[9]Deep, Kusum & Mebrahtu, Hadush. 2012. Variant of partially mapped crossover for the Travelling Salesman problems*. International Journal of Combinatorial Optimization Problems and Informatics***3**(1): 47-69

[10]Al Kasassbeh, M., Alabadleh, A., & Al-Ramadeen, T. 2012. Shared Crossover Method for Solving Traveling Salesman Problem*. International Journal of Intelligent Control and Systems(IJICS)***1**(6)*:*153-158

[11]Eiben, A.E. & Smith, J.E. 2007. *Introduction to Evolutionary Computing Genetic Algorithms*. Springer: New York

[12]Goldberg, David E. 1989. *Genetic Algorithms*. Pearson Education: London

[13]Loohach, Richa dan Garg, Kanwal. 2012. Efffect of Distance Functions on K-Means Clustering Algorithm. *International Journal of Computer Application***49**(6): 7-9

[14]Kumar, Rakesh and Jyotishree. 2012. Blending Roulette Wheel Selection & Rank Selection in Genetic Algorithms, *International Journal of Machine Learning and Computing***2**(4): 365-370

[15]Shaikh, Misba and Panchal, Mahesh. 2012. Solving Asymmetric Travelling Salesman Problem Using Memetic

Algorithm, *International Journal of Emerging Technology and Advanced Engineering* **2**(11): 634-639