PENYELSAIAN *MULTI TRAVELING SALESMAN PROBLEM* DENGAN ALGORITMA GENETIKA

Ni Kadek Mayuliana[1§], Eka N. Kencana[2], Luh Putu Ida Harini[3]

[1] Program Studi Matematika – Fakultas MIPA – Universitas Udayana
Email: maycan146@gmail.com

[2] Program Studi Matematika – Fakultas MIPA – Universitas Udayana
Email: i.putu.enk@unud.ac.id

[3] Program Studi Matematika – Fakultas MIPA – Universitas Udayana
Email: ballidah@gmail.com

[§] Corresponding Author

Abstract

Genetic algorithm is a part of heuristic algorithm which can be applied to solve various computational problems. This work is directed to study the performance of the genetic algorithm (GA) to solve Multi Traveling Salesmen Problem (multi-TSP). GA is simulated to determine the shortest route for 5 to 10 salesmen who travelled 10 to 30 cities. The performance of this algorithm is studied based on the minimum distance and the processing time required for 10 repetitions for each of cities-salesmen combination. The result showed that the minimum distance and the processing time of the GA increase consistently whenever the number of cities to visit increase. In addition, different number of sales who visited certain number of cities proved significantly affect the running time of GA, but did not prove significantly affect the minimum distance.

Keywords: Genetic Algorithm, Multi Traveling Salesman Problem, simulation

1 LATAR BELAKANG

Multi Traveling Salesmen Problem (multi-TSP) sebagai perluasan TSP (Carter & Ragsdale, 2006), merupakan jenis permasalahan optimasi integer dari $m > 1$ *salesmen* yang mengunjungi $n > m$ kota sedemikian hingga masing-masing kota dikunjungi hanya sekali. Optimasi pada multi-TSP ditujukan agar total jarak rute yang ditempuh para *sales* minimal (Sedighpour et al, 2011). Secara matematis, permasalahan multi-TSP dapat dinyatakan melalui persamaan berikut:

$$Z = \min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} \cdot x_{ij} \qquad (1)$$

dengan kendala-kendala

1. $\sum_{i=1}^{n} x_{ij} = 1$, untuk $j = 2, \cdots, n$;
2. $\sum_{j=1}^{n} x_{ij} = 1$, untuk $i = 2, \cdots, n$;
3. $\sum_{i=1}^{n} x_{i,1} \leq m$;
4. $\sum_{i=1}^{n} x_{1,j} \leq m$; dan
5. $$x_{ij} = \begin{cases} 1, & \text{bila } \textit{salesman} \text{ bergerak dari } i \text{ ke } j; \\ 0, & \text{bila } \textit{salesman} \text{ tidak bergerak.} \end{cases}$$

Adanya kendala 1 dan 2 menjamin setiap *node* hanya dikunjungi sekali oleh seorang *salesman* dan kendala 3 serta 4 menjaga agar setiap *salesman* yang bergerak akan berangkat dan kembali dari dan ke *node* awal.

Salah satu teknik komputasi yang bisa digunakan untuk menyelesaikan permasalahan multi-TSP adalah Algoritma Genetika (AG). Carter & Ragsdale (2006) menyatakan AG merupakan salah satu teknik pada kelompok *heuristic algorithm* yang mengadopsi teori evolusi dalam penyelesaian permasalahan optimasi. Pada AG, lazin ditemui proses seleksi, proses rekombinasi, dan proses mutasi untuk memperoleh kromosom terbaik sebagai representasi solusi permasalahan komputasi yang dihadapi.

Permasalahan komputasi diselesaikan AG dengan memodelkan ke dalam rangkaian kromosom (*encoding*), dilanjutkan dengan proses seleksi dan rekombinasi kromosom melalui teknik mutasi dan perpindahan silang gen (*crossover*) untuk menemukan kromosom terbaik. Proses diakhiri dengan melakukan *decoding* dari kromosom terbaik yang diperoleh sebagai solusinya.

Penelitian ini ditujukan untuk mengetahui kinerja AG dalam menyelesaikan permasalahan multi-TSP. Kinerja AG diukur dari waktu yang dibutuhkan untuk menemukan solusi jarak minimum yang ditempuh sejumlah *salesman* untuk mengunjungi sejumlah kota tujuan.

2 METODE PENELITIAN

Jenis & Sumber Data

Sebanyak m *salesmen*, $m \in \{5, \cdots, 10\}$ disimulasikan mengunjungi n kota, $n \in \{5, 10, \cdots, 30\}$. Untuk setiap n, matriks segi D berukuran $(n+1)^2$ dibangkitkan secara acak. Elemen-elemen D menunjukkan jarak antardua kota atau antara **depot** – titik awal atau titik akhir keberangkatan – dengan salah satu kota tujuan. Pada masing-masing kombinasi $n-m$ dilakukan 10 kali

ulangan.

Tahapan Analisis

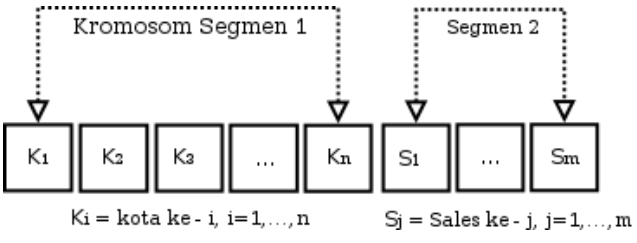
Tahapan analisis AG yang ingin diketahui kinerjanya dalam mencari solusi optimal persamaan (1) untuk masing-masing jumlah kota n yang disimulasikan, dilakukan melalui tahapan berikut:

1. Start
2. Untuk semua $m, m = 5, \cdots, 10$
3. Bangkitkan matriks jarak D
   (a) Ulangan $\leftarrow 1$
   (b) $\mathrm{D}_{Best} \leftarrow \infty$
   (c) While Ulangan ≤ 10 Do
   - I $\leftarrow 1$
   - Input MaxI, P_c, P_{mut}
   - While I $\leq$ MaxI Do
     - Bangkitkan populasi kromosom
     - Pilih kromosom terbaik
     - Hitung d = Panjang Rute
     - Jika $d < \mathrm{D}_{Best}$ maka $\mathrm{D}_{Best} = d$
     - I = I + 1
4. End

dengan MaxI merupakan maksimum iterasi pada suatu proses, P_c merupakan nilai probabilitas crossover, P_{mut} merupakan nilai probabilitas mutasi dan D_{Best} merupakan jarak minimum yang diperoleh setiap iterasi.

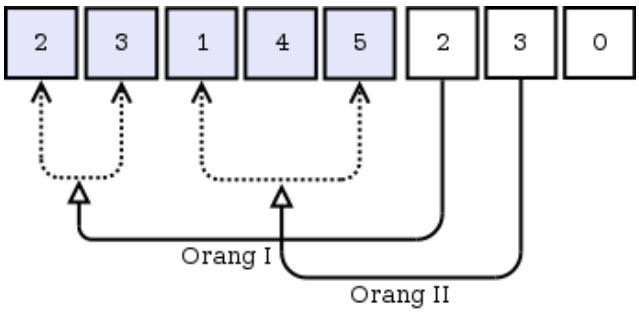
Kromosom dan Populasi

Mendefinisikan kromosom merupakan tahap awal dari aplikasi AG. Merujuk Haupt & Haupt (2004), kromosom tidak lain dari *array* nilai-nilai variabel yang akan dioptimisasikan. Sekelompok kromosom akan membentuk populasi. Pada penelitian ini ukuran populasi N_{pop} ditetapkan 20 kromosom, dengan representasi kromosom dinyatakan menggunakan *two-part chromosome technique* yang diusulkan oleh Carter & Ragsdale (2006) seperti terlihat pada Gambar 1:



Gambar 1. Representasi *Two-part Chromosomes*

Sebagai ilustrasi, pada multi-TSP dengan 5 kota tujuan dan 3 *salesmen*, maka representasi kromosom di mana hanya 2 *salesmen* yang bergerak dengan orang pertama mengunjungi kota 2 dan 3 serta orang kedua mengunjungi kota 1, 4, dan 5 bisa ditunjukkan pada Gambar 2. Algoritma untuk membangkitkan N_{pop} kromosom untuk masing-masing kombinasi $n - m$ diperlihatkan pada algoritma (1).



Gambar 2. Ilustrasi Kromosom pada AG

Algorithm 1: GenChr → Generate Chromosome

Input: Jumlah kota = n; Jumlah *sales* = m; Ukuran Populasi = N_{pop}
Output: Populasi kromosom $K = \{K_1, \cdots, K_{N_{pop}}\}$

```
K ← ∅; K_i ← ∅
for i ← 1 to N_pop do
    K_1 ← ∅ // Kromosom bagian I
    K_2 ← ∅ // Kromosom bagian II
    N_1 ← n // Salin jumlah kota n ke N_1
    for i ← 1 to m − 1 do
        Randomized R; R ∈ [0, N_1]
        K_2 ← K_2 ∪ R
        N_1 ← N_1 − R
    K_2 ← K_2 ∪ N_1
    i ← 0
    while i ≤ n do
        Randomized R; R ∈ [1, n]
        if R ∉ K_1 then
            K_1 ← K_1 ∪ R
            i = i + 1
    K_i ← K_1 ∪ K_2
    K ← K ∪ K_i
return K
```

Pembangkitan Matriks Jarak D

Elemen-elemen matriks jarak $D_{(n+1)(n+1)}$ untuk masing-masing n yang disimulasikan dibangkitkan secara acak. Matriks D dapat dinyatakan dalam bentuk:

$$\begin{pmatrix} 0 & d_{12} & \cdots & d_{1(n+1)} \\ d_{21} & 0 & \cdots & d_{2(n+1)} \\ \cdots & \cdots & \cdots & \cdots \\ d_{(n+1)1} & d_{(n+1)2} & \cdots & 0 \end{pmatrix}. \qquad (2)$$

Pada (2), diagonal utama dari D bernilai 0 mempertimbangkan jarak dari sebuah kota ke dirinya = 0. Misalkan untuk 5 kota tujuan secara acak dibangkitkan matriks D berukuran 6×6 dengan d_{1j} menyatakan jarak dari titik awal keberangkatan (depot) ke kota tujuan, sebagai berikut:

$$D_{6\times 6} = \begin{pmatrix} 0 & 2 & 3 & 4 & 7 & 1 \\ 2 & 0 & 9 & 5 & 7 & 8 \\ 3 & 9 & 0 & 7 & 6 & 4 \\ 4 & 5 & 7 & 0 & 2 & 8 \\ 7 & 7 & 6 & 2 & 0 & 1 \\ 1 & 8 & 4 & 8 & 1 & 0 \end{pmatrix}. \qquad (3)$$

Pada Gambar 2, rute yang ditempuh orang I dan II adalah D–2–3–D dan D–1–4–5–D, dengan D menyatakan depot atau titik awal dan akhir keberangkatan.

Menggunakan kendala 5 pada persamaan (1), rute kedua *sales* dapat dinyatakan dalam matriks kehadiran X berikut:

$$X_{6\times 6} = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \quad (4)$$

Menggunakan matriks kehadiran X tersebut, maka jarak dari rute yang ditempuh kedua *sales* sebesar

$$\begin{aligned} Jarak &= \sum_{i=1}^{i=6}\sum_{j=1}^{p=6} d_{ij}, \forall x_{ij} = 1 \\ &= 2+3+7+7+4+1+1 \\ &= 25. \end{aligned}$$

Penghitungan Nilai *Fitness*

Fitness menunjukkan ukuran kelaikan dari sebuah kromosom. Pada penelitian ini, *fitness* kromosom ke-p dengan $p = 1, \cdots, N_{pop}$ dihitung menggunakan persamaan (5)

$$Fitness_p = \frac{1}{Jarak_p} \quad (5)$$

Sebagai ilustrasi, menggunakan persamaan (5), nilai *fitness* kromosom pada Gambar 2 $= \frac{1}{25} = 0.04$. Selanjutnya, masing-masing kromosom pada populasi awal berukuran 20 kromosom dihitung nilai *fitness*-nya dan peluang kumulatif (P_{cum}) dari kromosom ke-p dihitung menggunakan persamaan (6) untuk proses seleksi individu yang dilakukan pada tahap berikutnya.

$$P_{cum(p)} = \frac{\sum_{i=1}^{i=p} Fitness_i}{\sum_{i=1}^{i=20} Fitness_i} \quad (6)$$

Pemilihan Kromosom Elite

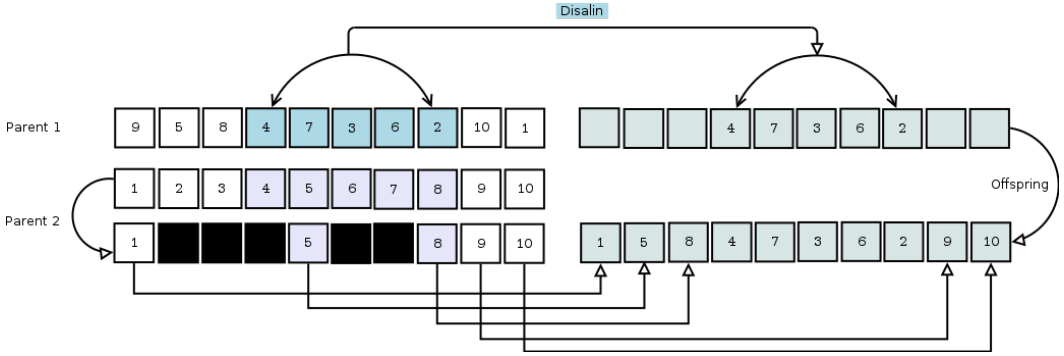
Sebelum dilakukan kawin silang (*crossover*) dan mutasi, sejumlah N_{elite} kromosom dipilih secara acak dari populasi awal. Proses pemilihan dilakukan menggunakan metode *roulette wheel*. Bila $P_{cum(i)} \leq R_i; i = 1, \cdots, N_{pop}$ dengan R_i menyatakan bilangan acak pada selang [0,1], maka kromosom ke-i terpilih sebagai anggota elite. Bila tidak, R_i lain dibangkitkan. Pada penelitian ini, proses diulangi sehingga jumlah kromosom yang terpilih $N_{elite} = N_{pop} = 20$.

Proses Kawin Silang dan Mutasi

Anggota kromosom populasi elite selanjutnya diseleksi untuk menentukan pasangan yang di-*crossover*-kan. Pada penelitian ini, teknik yang digunakan adalah *Order 1 Crossover* (OX1) mempertimbangkan kesederhanaan prosesnya. Jumlah kromosom yang mengalami kawin silang ditentukan sebesar N_{cross} bernilai genap dengan syarat $2 \leq N_{cross} \leq N_{elite}$. Masing-masing pasangan kromosom diprogram hanya menghasilkan satu kromosom anak (*offspring*).

Pemilihan pasangan dilakukan dengan menetapkan probabilitas kawin silang P_c dan membangkitkan bilangan acak R untuk kromosom ke-i dari populasi elite. Bila $R \leq P_c$, maka kromosom ke-i terpilih sebagai salah satu anggota pasangan. Anggota pasangan lainnya diperoleh dengan cara yang sama. Proses ini berhenti hingga seluruh kromosom pada populasi elite telah diperiksa. Gambar 3 memperlihatkan dua kromosom yang disilangkan dan *offspring* yang dihasilkan mengggunakan algoritma 2.

Kromosom *offspring* yang dihasilkan selanjutnya digabungkan ke dalam kelompok populasi elite. Bila masih terdapat kromosom induk yang belum dipasangkan, maka proses *crossover* diulangi, dan kromosom *offspring* yang dihasilkan juga digabungkan ke dalam kelompok elite sehingga di akhir proses ukuran populasi bertambah menjadi $N_{elite} + \frac{N_{cross}}{2}$.

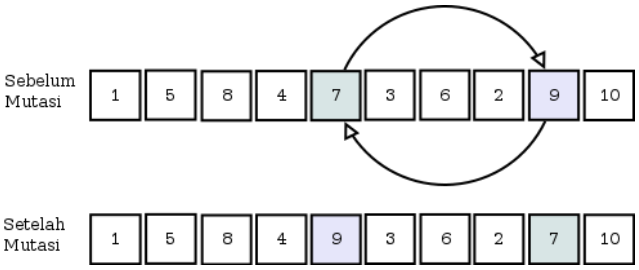


Gambar 3. Tahapan pada *Order 1 Crossover*

Proses terakhir yang dilakukan terhadap kromosom pada kelompok elite adalah proses mutasi, yang menurut Kumar et al. (2012) akan mencegah AG terperangkap pada penemuan solusi optimal yang bersifat lokal. Kromosom ke-i bermutasi bila nilai acak R_i yang dibangkitkan lebih kecil dari probabilitas mutasi (P_{mut}) yang pada penelitian ini ditetapkan sebesar 0.3 seperti disarankan oleh Tsujimura et al. (2001).

Penelitian ini mengaplikasikan teknik mutasi sisip (*insertion mutation*) pada kromosom yang terpilih. Teknik ini membangkitkan dua bilangan bulat acak – R_1 dan R_2 dengan syarat $1 \leq R_1 < R_2 \leq n$. Selanjutnya, alel pada posisi gen R_1 dan R_2 saling dipertukarkan.

Sebagai ilustrasi mutasi sisip, misalkan R_1 dan R_2 yang dibangkitkan untuk memutasi alel dari kromosom *offspring* pada gambar (3) adalah **5** dan **9**. Maka mutasi sisip akan mengubah nilai gen *offspring* menjadi:



Gambar 4. Tahapan pada Mutasi Sisip

Algorithm 2: CrossChr → Ordered Crossover

Input: Probabilitas *Crossover*=P_c, Pop. Orangtua C_i
Output: Elite baru $C_{(i)} = \{C_1, \cdots, C_{N_{elite}}\}$

$C_{mating} \leftarrow \emptyset$; *Repeat* $\leftarrow$ *true*; $j \leftarrow 0$
while *Repeat* **do**
  for $i \leftarrow 1$ ***to*** N_{pop} **do**
    Randomized R // Bangkitkan bil. acak
    if $R < P_c$ **then**
      $j = j + 1$
      $C_{mating(j)} \leftarrow C_{mating(j)} \cup C_i$
  if $j\%2 = 0$ **then**
    Repeat $\leftarrow$ *false*
for $i \leftarrow 1$ ***to*** $(j-1)$ **do**
  $O_{(i)} \leftarrow \emptyset$ // Kromosom *offspring* ke-i
  Randomized R_1, R_2; $R_1, R_2 \in [1, length(K_1)]$
  if $R_2 < R_1$ **then**
    swap (R_1, R_2) // Tukar R_1 dengan R_2
  /* Salin alel P_1 ke alel *offspring* */
  for $k \leftarrow 1$ *to* $length(K_i)$ **do**
    if $k \in [R_1, R_2]$ **then**
      /* alel ke-k disalin ke *offspring* */
      $O_{(i,k)} = C_{(i,k)}$
    else
      $O_{(i,k)} = 0$
    $O_{(i)} \leftarrow O_{(i)} \cup O_{(i,k)}$
  /* Salin alel P_2 ke alel *offspring* */
  for $k \leftarrow R_2 + 1$ *to* $length(K_i)$ **do**
    if $C_{(i+1,k)} \notin O_i$ **then**
      $O_{(i,k)} = C_{(i+1,k)}$
      $k \leftarrow k + 1$
  for $k \leftarrow 1$ *to* $(R_1 + 1)$ **do**
    if $C_{(i+1,k)} \notin O_i$ **then**
      $O_{(i,k)} = C_{(i+1,k)}$
      $k \leftarrow k + 1$
  $O_{(i)} \leftarrow O_{(i)}$
  $C_{(i)} \leftarrow C_{(i)} \cup O_{(i)}$
  return $C_{(i)}$

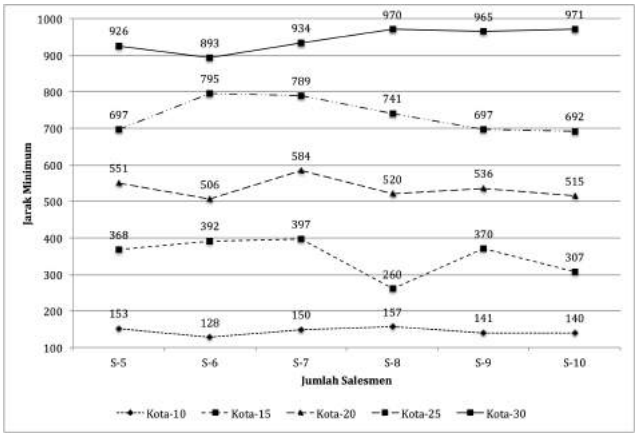
3 HASIL SIMULASI

Untuk mengetahui kinerja AG dalam menemukan solusi optimal multi-TSP, dua indikator kinerja diamati yaitu (a) total jarak, dan (b) *running time* dari program Matlab yang dirancang.

Simulasi dirancang dengan proses *crossover* dan mutasi dilakukan pada kromosom **segmen 1** dengan tujuan jumlah kota yang dikunjungi *salesman* tidak berubah. Masing-masing kombinasi $n - m$ direplikasi 10 kali dengan jumlah iterasi untuk memperoleh jarak minimum ditetapkan sebesar 200. Nilai-nilai P_{cross} dan P_{mut} yang digunakan masing-masing sebesar 0.5 dan 0.3.

Optimasi Jarak Minimum

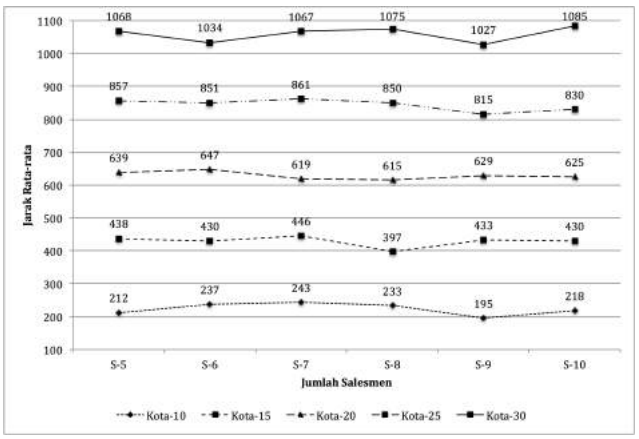
Jarak minimum yang diperoleh untuk masing-masing kombinasi $n - m$ dengan konfigurasi simulasi seperti uraian sebelumnya diperlihatkan pada Gambar 5:



Gambar 5. Plot Jumlah *Sales* terhadap Jarak Minimum

Gambar 5 memperlihatkan semakin banyak jumlah kota n yang harus dikunjungi semakin besar jarak minimum yang diperoleh. Pada $n = 10$, AG memberikan hasil 128 satuan sebagai jarak minimum terkecil dari rute yang ditempuh saat $m = 6$, dan terbesar pada saat $m = 8$ dengan nilai 157. Pada $n = 30$, jarak minimum terkecil diperoleh saat $m = 6$ dan terbesar pada saat $m = 10$.

Secara deskriptif, tidak terlihat adanya pola kausal dari bertambahnya *salesmen* terhadap jarak minimum untuk jumlah kota m yang harus dikunjungi. Untuk mengkonfirmasi temuan tersebut, maka uji statistika dilakukan. Nilai rataan $(\bar{x})$ jarak yang dihitung dari 10 ulangan untuk setiap kombinasi $n - m$ digunakan dalam uji ini. Gambar 6 menunjukkan plot jumlah *salesmen* dengan rataan jarak rute.



Gambar 6. Plot Jumlah *Sales* terhadap Rataan Jarak

Hasil uji *Analysis of Variance* (ANOVA) juga menunjukkan bertambahnya *salesmen* pada n yang sama tidak berpengaruh signifikan terhadap jarak rute. Uji ANOVA untuk masing-masing jarak ditunjukkan pada Tabel 1.

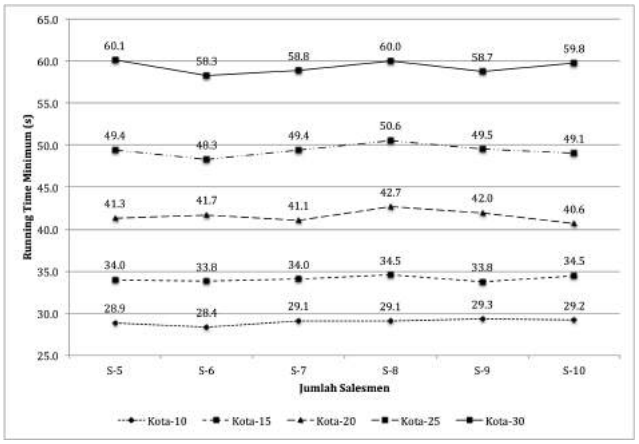
Tabel 1. ANOVA Pengaruh Jumlah *Sales* vs. Jarak Rata-rata

Jumlah	Nilai F	Nilai p	Keterangan
10 Kota	1.339	0.262	Tidak signifikan
15 Kota	1.143	0.349	Tidak signifikan
20 Kota	0.463	0.802	Tidak signifikan
25 Kota	0.764	0.579	Tidak signifikan
30 Kota	1.198	0.323	Tidak signifikan

Sumber: Data primer (2016), dianalisis.

Optimasi *Running Time*

Indikator kedua yang dianalisis dalam kinerja AG pada permasalahan multi-TSP adalah waktu (*running time*) yang diperlukan untuk menemukan solusi optimal pada masing-masing kombinasi $n - m$. Deskripsi waktu minimum diperlihatkan pada Gambar 7.



Gambar 7. Plot Jumlah *Sales* terhadap Waktu Minimum

Seperti halnya dengan indikator jarak rute, semakin besar jumlah kota (n) yang harus dikunjungi maka *running time* yang diperlukan untuk menemukan solusi optimum juga meningkat. Meski demikian, secara deskriptif terlihat bertambahnya *salesmen* tidak menjamin bertambahnya waktu eksekusi program.

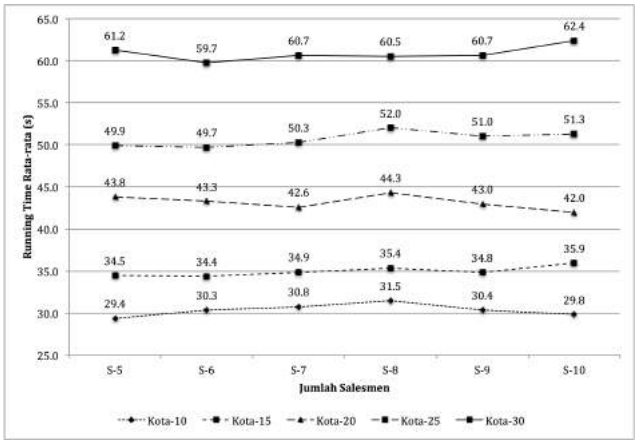
Untuk mengetahui apakah terdapat perbedaan nyata secara statistika terhadap *running time* algoritma pada n tertentu untuk m yang berbeda, uji ANOVA pada rata-rata waktu *running time* dilakukan dengan hasil ditunjukkan pada Tabel 2.

Tabel 2. ANOVA Pengaruh Jumlah *Sales* vs *Running Time*

Jumlah	Nilai F	Nilai p	Keterangan
10 Kota	4.677	0.001	Signifikan
15 Kota	6.573	0.000	Signifikan
20 Kota	6.676	0.000	Signifikan
25 Kota	4.953	0.001	Signifikan
30 Kota	7.890	0.000	Signifikan

Sumber: Data Primer (2016)

Merujuk hasil ANOVA pada Tabel 2, maka untuk 5 jumlah kota yang disimulasikan terbukti penambahan jumlah *sales* berpengaruh secara signifikan pada *running time* dari AG. Plot antara jumlah *sales* dengan *running time* rata-rata dari 10 ulangan diperlihatkan pada Gambar 8.



Gambar 8. Plot Jumlah *Sales* terhadap Waktu Rata-rata

4 SIMPULAN DAN SARAN

Simpulan

Penelitian yang ditujukan untuk mengetahui kinerja AG dalam menyolusikan permasalahan multi-TSP menyimpulkan:

1. Panjang rute terpendek yang diperoleh untuk n kota yang sama tidak dipengaruhi secara nyata oleh adanya penambahan tenaga *salesmen* dari nilai awal 5 orang hingga menjadi 10 orang. Hasil yang sama juga diperoleh bila yang dijadikan indikator adalah nilai rataan panjang rute terpendek yang diperoleh dari 10 ulangan simulasi;

2. Terdapat perbedaan yang signifikan pada perubahan *running time* dengan bertambahnya *salesmen*, meski tidak dapat disimpulkan bahwa pengaruhnya bersifat positif.

Saran

Penelitian ini merupakan penelitian awal tentang kinerja AG dalam mencari solusi dari multi-TSP melalui pengaturan lingkungan simulasi yang sederhana. Disarankan ada penelitian lanjutan untuk memodifikasi teknik crossover dan atau teknik mutasi yang digunakan dalam memilih populasi elite. Teknik *partially mapped crossover* (PMX) atau *modified order crossover* yang diusulkan oleh Sehrawat & Singh (2011) layak untuk diujicoba.

REFERENSI

Carter, A. E., C. T. Ragsdale. "A new approach to solving the multiple traveling salesperson problem using genetic algorithms", European Journal of Operational Research. No. 175, pp. 246-257, 2006.

Haupt, R. L., S. E. Haupt. "Practical Genetic Algorithms", 2nd. Ed. John Wiley & Sons, Inc. Canada. 2004.

N. Kumar, Karambir, R. Kumar, A Comparative Analysis of PMX, CX and OX Crossover operators for solving Travelling Salesman Problem, International Journal of Latest Research in Science and Technology. Vol. 1, No. 2, pp. 98-101, 2012.

Rostami, A. S, F. Mohanna, H. Keshavarz, A. A. R. Hosseinabadi. "Solving Multiple Traveling Salesman Problem using the Gravitational Emulation Local Search Algorithm", Applied Mathematics & Information Sciences. Vol. 9, No. 2, pp.699-709, 2015.

Sehrawat, M., S. Singh, "Modified Order Crossover (OX) Operator", International Journal on Computer Science and Engineering (IJCSE), Vol. 3 No. 5 May, pp. 2019-2023, 2011.

Sedighpour, M., M. Yousefikhoshbakht, N. M. Darani. "An Effective Genetic Algorithm for Solving the Multiple Traveling Salesman Problem", Journal of Optimization in Industrial Engineering. No. 8, pp. 73-79, 2011.

S. Z. Selim, M. A. Ismail, K-means-type algorithm: generalized convergence theorem and characterization of local optimality, IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 6, No. 1, pp. 81-87, 1984.

Tsujimura, Y., Y. Mafune, M. Gen, "Effects of Symbiotic Evolution in Genetic Algorithms for Job-Shop Scheduling", Proceedings of the 34th Hawaii International Conference on System Sciences, pp. 1-7, 2001.