SAINTEKS: JURNAL SAIN DAN TEKNIK

Volume 5 Nomor 1 Tahun 2023

E-ISSN : 2685-8304

Diterbitkan Oleh :
UNIVERSITAS INSAN CENDEKIA MANDIRI dh UNIVERSITAS BANDUNG RAYA
Fakultas Teknik
Jl. Pasir Kaliki No. 199 Bandung – Jawa Barat
http://ejournal.uicm-unbar.ac.id/index.php/sainteks

UICM – UNBAR
www.uicm.ac.id

Sainteks: Jurnal Sain dan Teknik
E-ISSN: 2685-8304
Volume 5 Nomor 1 Tahun 2023
http://ejournal.uicm-unbar.ac.id/index.php/sainteks/

Genetic Algorithm for Improving Route of Travelling Salesman Problem Generated by Savings Algorithm

Genetic Algorithm Untuk Memperbaiki Rute *Travelling Salesman Problem* Yang Dihasilkan Dari *Savings Algorithm*

Muhammad Ardhya Bisma*[1)], Ekra Sanggala[2)],
[1)] Universitas Logistik dan Bisnis Internasional. Jalan Sariasih No. 54, Bandung, 40151
Email: bisma@ulbi.ac.id
[2)]Universitas Logistik dan Bisnis Internasional. Jalan Sariasih No. 54, Bandung, 40151
Email: ekrasanggala@mail.ru
*) Corresponding Author

Abstract: *Travelling Salesman Problem (TSP) is the problem for finding the shortest route starting from start node then visiting number of nodes exactly once and finally go back to start node. If a TSP has a lot of nodes, it will be a NP-Hard Problem. Algorithms working based on heuristic and metaheuristic can be a solution for solving NP-Hard Problem. Savings Algorithm is a heuristic algorithm, so it's solution may be not the best solution, therefore there is a chance to improve it. Genetic Algorithm is a metaheuristic that can be applied on many optimization problems, including TSP. This paper will discuss about GA for improving route TSP generated by Savings Algorithm. On testing of 10 instances, showing that algorithm based on GA can improve route of TSP generated by Savings Algorithm.*
Keywords: Genetic Algorithm, Savings Algorithm, Travelling Salesman Problem

Abstrak: *Travelling Salesman Problem* berfokus pada penentuan rute paling pendek yang dimulai dari titik awal guna mengunjungi sekumpulan titik tujuan tepat satu kali yang kemudian diakhiri dengan kembali ke titik awal. TSP dengan banyak titik dapat diklasifikasikan sebagai *NP-Hard Problem* yang pada umumnya diselesaikan dengan *heuristic* dan *metaheuristic.* Algoritma yang termasuk dalam *heuristic* tidak selalu menghasilkan solusi terbaik. Dalam konteks ini, *Savings Algorithm* termasuk kedalam *heuristic, sedangkan Genetic Algorithm* (*GA*) termasuk kedalam *metaheuristic.* Hal ini mengindikasikan bahwa solusi yang dihasilkan oleh *Savings Algorithm* belum tentu solusi yang paling baik, sehingga masih terdapat kemungkinan untuk diperbaiki lebih lanjut dengan penggunaan *Genetic Algorithm* (*GA*). Penelitian ini melakukan pengujian dengan menggunakan sepuluh (10) *instance,* dan didapatkan bahwa *Genetic Algorithm* (*GA*) dapat memperbaiki rute TSP yang sebelumnya dihasilkan oleh *Savings Algorithm*.
Kata Kunci: *Genetic Algorithm*, *Savings Algorithm*, *Travelling Salesman Problem*

DOI: http://dx.doi.org/10.37577/sainteks.v%vi%i.542
Received: 01, 2023. Accepted: 02, 2023
Published: 03, 2023

PENDAHULUAN

Travelling Salesman Problem (TSP) berfokus pada penentuan rute paling pendek yang dimulai dari titik awal guna mengunjungi sekumpulan titik tujuan tepat satu kali yang kemudian diakhiri dengan kembali ke titik awal. Berbagai fenomena di dunia ini dapat didefinisikan dengan pendekatan TSP, seperti rute perjalanan wisatawan, rute transportasi publik, rute pendistribusian komoditi dan sebagainya. TSP dapat dimodelkan dalam bentuk *integer programming* seperti berikut ini (Lavrov, 2020):

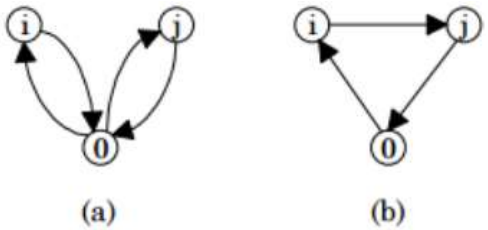
$$minimize \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$$

$$\sum_{\substack{1\le i\le n \\ i\neq j}} x_{ij} = 1 \qquad for\ each\ j = 1,2,\dots,n$$

$$\sum_{\substack{1\le k\le n \\ k\neq j}} x_{jk} = 1 \qquad for\ each\ j = 1,2,\dots,n$$

Ketika TSP memiliki banyak titik, maka TSP tersebut akan menjadi *NP-Hard Problem (nondeterministic polynomial time).* Sebagaimana yang diketahui, bahwa *NP-Hard Problem* memerlukan waktu yang lama guna mendapatkan penyelesaiannya (Applegate et al., 2006). Meskipun demikian, *NP-Hard Problem* dapat diselesaikan dalam jangka waktu yang wajar dengan menggunakan algortima *heuristic* serta *metaheuristic* meskipun solusi yang dihasilkan belum tentu menjadi solusi yang terbaik (Labadie et al. 2016).

Savings Algorithm merupakan salah satu *heuristic* yang populer dalam menyelesaikan TSP (Chiarandini, 2020). *Savings Algorithm* pertama kali diprakarsai oleh Clarke dan Wright pada tahun 1964 (Clarke, 1964). Adapun prinsip pokok dari *Savings Algorithm* adalah penghematan (*savings*) jarak tempuh yang diperoleh berdasarkan penggabungan dua rute yang berbeda. Ilustrasi penggabungan tersebut tercantum pada gambar 1, dimana titik 0 merupakan merupakan titik asal, dan titik i serta titik j adalah titik tujuannya (Lysgaard, 2007).



Gambar 1: Contoh Penggabungan 2 rute (Lysgaard, 2007).

Pada gambar 1(a) diketahui bahwa titik i dan titik j dicapai oleh rute yang berlainan, sedangkan pada gambar 1(b) didapatkan bahwa titik i dan titik j dicapai oleh rute yang sama. Dapat diketahui jarak tempuh D_a pada gambar 1(a) dapat diterjemahkan dengan persamaan sebagai berikut ini (Lysgaard, 2007):

$$D_a = d_{0i} + d_{i0} + d_{0j} + d_{j0}$$

Sedangkan untuk jarak tempuh D_b pada gambar 1(b) dapat diterjemahkan dengan persamaan ini: (Lysgaard, 2007):

$$D_b = d_{0i} + d_{ij} + d_{j0}$$

Nilai *Savings* diperoleh dengan mengurangkan jarak tempuh D_a dengan jarak tempuh D_b, sehingga persamaan nilai *Savings* S_{ij} menjadi seperti ini (Lysgaard, 2007):

$$S_{ij} = D_a - D_b = d_{i0} + d_{0j} - d_{ij}$$

Pada *Savings Algorithm*, langkah utama yang harus dilakukan adalah melakukan perhitungan nilai penghematan (*savings*) untuk setiap pasangan titik. Setelah setiap nilai penghematan *(savings)* untuk masing – masing pasangan sudah diperoleh, maka hal selanjutnya yang harus dilakukan adalah melakukan pengurutan dimulai dari nilai *Savings* terbesar hingga ke nilai *Savings* terkecil. Solusi berupa rute tersebut akan didapatkan dengan menyisipkan rangkaian

titik yang berkorespondensi dan memiliki nilai penghematan (*savings*) lebih tinggi (Lysgaard, 2007).

Adapun *Genetic Algorithm* (*GA*) pada dasarnya adalah proses dengan pendekatan *heuristic* yang dapat diterapkan guna menyelesaikan berbagai permasalahan terkait optimasi (Frutos, 2012). Pada dasarnya *Genetic Algorithm* memiliki mengacu pada ide yang didasarkan pada evolusi dan proses seleksi alam. Suatu "solusi" dari sebuah permasalahan dianalogikan sebagai spesies yang memiliki kemampuan untuk beradaptasi, menghasilkan keturunan dan berkembang hingga akhirnya mampu menjadi spesies yang optimal (Awad, 2018). Hal tersebut memberikan justifikasi yang relevan, bahwa dengan berbagai permasalahan optimasi dapat dilakukan dengan mengadopsi prinsip – prinsip evolusi. Konsep tersebut dituangkan ke dalam beberapa tahapan umum pada *Genetic Algorithm* sebagai berikut (Kramer, 2017):

1. *Setting Parameters*
   Pada langkah ini, ditentukan nilai-nilai *parameters* dan *Genetic Algorithm*. *Parameters* tersebut antara peluang terjadinya *Crossover*, peluang terjadinya *Mutation*, banyaknya individu yang akan terseleksi dan banyaknya generasi yang harus tercapai agar tercapai kondisi *Termination* (Talbi, 2009).

2. *Initialize Population*
   Merupakan penentuan individu-individu yang akan menjadi populasi awal atau populasi yang hidup di generasi 0 (Talbi, 2009).

3. *Crossover*
   Crossover merupakan perkawinan silang yang dilakukan oleh sepasang orang tua (Talbi, 2009).

4. *Mutation*
   Mutation merupakan perubahan susunan gen yang terjadi pada suatu individu (Talbi, 2009).

5. *Genotype-phenotype Mapping*
   Genotype adalah susunan gen pada individu yang belum menjadi solusi untuk menyelesaikan permasalahan. *Genotype* ini harus diterjemahkan menjadi *phenotype* agar dapat menjadi sebuah solusi (Talbi, 2009).

6. *Fitness*
   Fitness adalah sebuah nilai yang menyatakan seberapa baik kualitas suatu individu (Talbi, 2009).

7. *Selection*
   Selection adalah pemilihan sejumlah individu untuk digunakan pada proses selanjutnya (Talbi, 2009).

8. *Termination*
   Termination adalah suatu kondisi dimana proses evolusi dihentikan (Talbi, 2009).

Penelitian ini berfokus pada penerapan GA guna memperbaiki solusi rute TSP yang dihasilkan oleh *Savings Algorithm*. Dalam konteks ini, rute TSP yang dihasilkan oleh *Savings Algorithm* akan dijadikan sebagai bakal solusi yang mengalami "seleksi alam" dan "evolusi" sebagaimana yang direpresentasikan pada tahapan – tahapan GA. Dengan demikian diharapkan solusi rute yang diperoleh akan menjadi lebih baik.

METODOLOGI

Secara umum tahapan – tahapan yang dilakukan adalah sebagai berikut:

1. Pemilihan *instance*
   Instance digunakan sebagai kumpulan data awal yang akan menjadi acuan pengerjaan pada penelitian ini. Seluruh *instance* tersebut didapatkan melalui laman http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/. *Instance* ini pernah digunakan pada penelitian sebelumnya (Sanggala, 2021). Adapun penelitian ini menggunakan sepuluh (10) *instance* sebagai berikut:
   a) berlin52
   b) eil51
   c) eil76
   d) eil101
   e) kroA100
   f) kroB100
   g) kroC100
   h) kroD100
   i) kroE100
   j) pr76

   Kesepuluh *instance* yang akan digunakan, dibuat oleh *author* yang berbeda-beda, tetapi struktur penulisan pada file *text*-nya sudah sama sehingga semuanya dapat diperlakukan secara sama (Reinelt, 1995).

2. Penerapan *Savings Algorithm* pada *instance* dan Pengembangan Algoritma Menggunakan *Genetic Algorithm*
   Instance yang dipilih diselesaikan terlebih dahulu dengan menggunakan *Savings Algorithm*, kemudian rute yang dihasilkan berdasarkan *Savings Algorithm* tersebut dijadikan bakal solusi untuk selanjutnya diproses dengan menggunakan tahapan – tahapan pada *Genetic Algorithm (crossover, mutation* dan *selection)*.

3. Pengujian atas Algoritma yang dikembangkan
   Algoritma yang didapatkan pada tahapan sebelumnya diujikan kembali pada *instance*. Hal ini dilakukan untuk membandingkan dan memverifikasikan performa dari rute yang dihasilkan dengan *Saving Algorithm* dengan rute baru yang dihasilkan berdasarkan *Saving Algorithm* yang kemudian dilanjutkan dengan pendekatan *Genetic Algorithm*.

HASIL PENELITIAN DAN PEMBAHASAN

Instance yang digunakan merupakan *instance* yang umum digunakan oleh para peneliti lain guna menguji hasil pengembangan algoritma yang didapatkan. Adapun data dari *instance* tersebut dapat dilihat pada Tabel 1.

Tabel 1: *Instance* yang Digunakan

No	*Instance*	Jumlah Titik	*Author*
1	berlin52	52	Groetschel
2	eil51	51	Christofides & Elion
3	eil76	76	Christofides & Elion
4	eil101	101	Christofides & Elion
5	kroA100	100	Krolak, Felts & Nelson
6	kroB100	100	Krolak, Felts & Nelson
7	kroC100	100	Krolak, Felts & Nelson
8	kroD100	100	Krolak, Felts & Nelson
9	kroE100	100	Krolak, Felts & Nelson
10	pr76	76	Padberg & Rinaldi

Masing – masing *instance* tersebut kemudian diproses dengan menggunakan *Savings Algorithm* dan menghasilkan rute yang sebagaimana tercantum pada tabel 2.

Tabel 2: Rute yang Dihasilkan dari *Savings Algorithm*

No	*Instance*	Rute	Panjang Rute
1	berlin52	1-35-36-39-34-44-50-20-23-31-21-18-3-17-42-7-2-30-29-16-46-37-40-38-48-24-5-15-6-4-25-28-26-47-14-52-13-27-11-51-12-33-43-10-9-8-41-19-45-32-49-22-1	8381,9068
2	eil51	1-8-26-31-28-3-20-36-35-29-16-38-5-12-47-18-4-17-37-15-49-9-50-21-34-30-10-39-33-45-44-42-19-40-41-13-25-14-24-43-7-23-6-48-27-51-46-11-2-22-32-1	466,0787
3	eil76	1-33-73-62-2-74-28-61-21-69-36-37-71-60-70-20-15-57-13-54-19-53-14-59-66-65-11-38-10-31-55-25-9-39-72-58-7-35-8-46-34-52-27-45-29-5-47-48-30-4-67-26-12-40-32-50-18-24-49-44-3-17-76-75-68-6-51-16-63-23-56-41-42-64-22-43-1	590,6276
4	eil101	1-50-70-30-51-33-81-77-76-28-27-52-89-101-53-58-40-26-12-80-68-3-79-78-9-20-66-71-65-35-34-29-24-54-55-25-4-21-73-72-74-75-56-39-67-23-22-41-57-42-15-43-14-38-44-86-16-91-100-37-98-85-61-17-84-45-46-36-49-64-11-19-47-48-8-83-5-93-59-92-97-87-2-95-99-96-94-13-6-60-18-82-7-62-63-90-32-10-88-31-69-1	735,6019
5	kroA100	1-92-63-6-49-10-72-36-84-90-19-75-8-42-89-31-80-56-97-4-65-26-66-79-53-88-16-70-22-94-18-24-38-99-21-74-59-11-17-15-45-23-77-60-62-61-51-87-57-9-7-20-35-86-27-12-55-83-34-25-81-69-73-68-85-39-30-29-46-43-3-14-41-71-100-48-52-78-96-5-37-33-76-13-95-82-50-44-2-54-40-64-58-67-28-93-98-91-32-47-1	25292,1927
6	kroB100	1-21-90-12-98-32-59-29-8-99-76-97-91-28-11-93-85-73-53-40-39-70-67-5-62-26-100-56-72-37-65-79-81-47-88-23-22-54-52-89-43-50-55-77-24-18-96-19-44-92-41-17-36-45-78-13-63-31-48-51-82-42-2-16-14-64-33-15-6-83-4-87-60-74-66-94-57-7-84-58-34-27-61-35-71-9-25-46-20-80-75-30-69-49-86-38-68-10-3-95-1	25444,7490
7	kroC100	1-53-40-12-46-29-24-32-61-26-7-78-82-9-37-18-49-93-4-60-14-19-99-92-10-36-57-45-66-16-51-63-44-48-84-11-52-96-87-97-81-33-100-74-69-3-73-59-41-89-21-23-91-70-76-94-95-50-72-39-71-86-5-43-56-38-28-88-98-58-34-90-25-17-8-22-83-62-75-54-6-2-35-68-30-77-80-65-31-47-67-55-42-64-20-79-13-15-27-85-1	22916,1169
8	kroD100	1-27-13-79-88-24-41-84-73-69-15-63-39-71-4-93-44-60-53-5-62-35-37-19-51-30-100-2-48-92-55-87-26-10-21-56-95-28-12-47-9-20-25-82-18-6-40-83-3-59-85-64-77-43-91-54-16-96-86-14-68-33-45-99-31-57-72-78-65-80-76-7-75-67-36-29-58-22-42-23-74-61-46-52-8-66-38-89-70-49-98-97-32-94-17-90-11-34-81-50-1	23761,2388

Tabel 2: Rute yang Dihasilkan dari *Savings Algorithm* (Lanjutan)

No	*Instance*	Rute	Panjang Rute
9	kroE100	1-83-73-16-47-3-46-26-79-61-71-81-89-52-14-44-72-68-38-33-22-39-86-43-19-94-18-24-29-100-78-53-77-85-96-11-2-4-70-37-34-6-55-91-93-92-82-67-27-41-87-75-13-66-30-90-42-5-48-31-9-57-80-10-60-50-97-65-8-25-54-45-99-17-84-49-20-15-56-23-28-36-95-64-62-58-59-98-32-63-35-40-88-12-76-7-21-51-69-74-1	24479,0259
10	pr76	1-23-22-21-24-46-45-25-3-4-20-19-31-30-26-27-29-32-33-28-43-42-54-53-44-48-47-69-68-70-67-49-50-52-55-56-51-66-65-71-72-73-64-63-62-61-57-58-59-60-41-40-39-38-34-35-36-37-18-17-16-15-13-14-74-12-11-10-9-5-6-7-8-2-75-76-1	117072,8770

Setelah mendapatkan rute dari *Savings Algorithm*, hal selanjutnya yang dilakukan adalah menjadikan rute – rute tersebut sebagai bakal solusi dengan algoritma berdasarkan *Genetic Algorithm*. Adapun parameter – parameter yang digunakan dalam prosesnya adalah sebagai berikut:

1. *Mutated Individual Size*
   Parameter ini memberikan jumlah individu yang terbentuk dari hasil mutasi tanpa *crossover/*perkawinan. Dalam penelitian ini, nilai yang ditetapkan untuk *parameter* ini adalah sebesar 30 individu, yang mana sama dengan nilai populasinya.

2. *Maximum Mutated Individual Pair Gen*
   Parameter ini digunakan untuk menentukan jumlah maksimal pasangan gen yang akan mengalami mutasi. Adapun nilai *Maximum Mutated Individual Pair Gen* yang ditetapkan dalam penelitian adalah 45% dari banyaknya titik pada *instance*. Dalam prosesnya, dilakukan pembulatan ke bawah jika nilai yang dihasilkan berupa pecahan.

3. *One Point Crossover Couple Size*
   One Point Crossover Couple Size mendefinisikan jumlah pasangan yang akan melakukan *crossover* (perkawinan). Nilai parameter ini ditetapkan sebesar 50%. Dalam prosesnya, dilakukan pembulatan ke bawah jika nilai yang dihasilkan berupa pecahan.

4. *Maximum Mutated One Point Crossover Individual Pair Gen*
   Parameter ini menentukan jumlah individu yang akan bermutasi dari hasil *crossover* sebelumnya. Nilai parameter yang ditetapkan adalah 45% dari titik instance. Dalam prosesnya, dilakukan pembulatan ke bawah jika nilai yang dihasilkan berupa pecah

5. *Survivor Size*
   Parameter ini menentukan jumlah individu yang akan terus hidup pada generasi selanjutnya. Nilai yang ditetapkan adalah sebesar $\frac{5}{6}$ dari jumlah individu dalam populasi. Dalam prosesnya, dilakukan pembulatan ke atas jika nilai yang dihasilkan berupa pecahan. Sebagai contoh jika populasi yang ditetapkan adalah 30 individu, maka jumlah individu yang berhasil hidup pada generasi selanjutnya adalah sebesar $\frac{5}{6}\ x\ 30 =$ 25 individu.

6. *Random Individual Size*
   Parameter ini menentukan jumlah individu yang akan muncul secara acak ketika nilai populasi berada di bawah parameter yang ditetapkan sebelumnya. Hal ini berhubungan dengan parameter sebelumnya. Sebagai contoh pada perhitungan poin (5) didapatkan nilai populasi sebesar 25 individu. Nilai tersebut masih berada di bawah nilai populasi yang ditetapkan, yakni 30 individu. Dengan demikian, akan muncul 5 individu dengan susunan gen yang acak agar memenuhi nilai populasi sebesar 30 individu.

7. *Maximum Repeat Shortest Length*
   Parameter ini pada dasarnya mengindikasikan iterasi perhitungan yang dilakukan. Secara spesifik, pemeter ini menentukan jumlah maksimal kelanjutan generasi yang harus dilewati oleh individu/solusi terbaik yang telah terpilih pada generasi sebelumnya. Nilai yang ditentukan dalam penelitian ini adalah 1000 generasi. Hal ini berarti individu terbaik tersebut harus mampu *survive*/tetap menjadi yang terbaik selama 1000 generasi berikutnya.

Dalam pelaksanaannya, algoritma yang sudah dikembangkan ini dituliskan ulang ke dalam bahasa pemrograman pada GNU Octave. Hal ini dilakukan agar proses komputasi dari TSP ini dapat dilakukan dengan menggunakan komputer. *GNU Octave* sendiri merupakan *tools* untuk membantu proses komputasi numerik (Hansen, 2011). Adapun hasil rute yang didapatkan dari pengembangan algoritma tersebut dapat dilihat pada tabel 3 berikut:

Tabel 3: Rute yang Dihasilkan dari Algoritma yang Dikembangkan Berdasarkan *GA*

No	*Instance*	Rute	Panjang Rute	Hasil Proses
1	berlin52	1-35-36-39-34-44-50-20-23-21-31-18-3-17-42-7-2-30-29-16-46-37-40-38-48-24-5-15-6-4-25-12-26-47-14-52-13-27-28-11-51-33-43-10-9-8-41-19-45-32-49-22-1	8196,3654	One Point CrossOver Mutation
2	eil51	1-8-26-31-28-3-20-36-35-29-16-38-5-12-47-18-4-17-37-15-49-9-50-21-34-30-10-39-33-45-44-42-19-40-41-13-25-14-24-43-7-23-48-6-27-51-46-32-11-2-22-1	459,2792	One Point CrossOver Mutation
3	eil76	1-33-73-62-2-74-28-61-21-69-36-37-71-60-70-20-15-57-13-54-19-53-14-59-66-11-65-38-10-31-55-25-9-39-72-58-7-35-8-46-34-52-27-45-29-5-47-48-30-4-67-26-12-40-32-50-18-24-49-3-44-17-76-75-68-6-51-16-63-23-56-41-42-64-22-43-1	586,121	One Point CrossOver Mutation
4	eil101	1-50-70-30-51-33-81-77-76-28-27-52-89-101-53-58-40-26-12-80-68-3-79-78-9-20-66-71-65-35-34-29-24-54-55-25-4-21-73-72-74-75-56-39-67-23-22-41-57-42-15-43-14-44-38-86-16-91-100-37-98-85-61-84-17-45-46-36-49-64-11-19-47-48-8-83-5-93-59-92-97-87-2-13-94-95-99-96-6-60-18-82-7-62-63-90-32-10-88-31-69-1	725,9273	Individual Mutation

Tabel 3: Rute yang Dihasilkan dari Algoritma yang Dikembangkan Berdasarkan *GA* (Lanjutan)

No	*Instance*	Rute	Panjang Rute	Hasil Proses
5	kroA100	1-92-63-6-49-10-72-36-84-90-19-75-8-42-89-31-80-56-97-4-65-26-66-53-79-88-16-70-22-94-18-24-38-99-21-74-59-11-17-15-45-23-77-60-62-61-51-87-57-9-7-20-35-86-27-12-55-83-34-25-81-69-73-68-85-39-30-29-46-43-3-14-41-71-100-48-52-78-96-5-37-33-76-13-95-82-50-44-2-54-40-64-58-67-28-93-98-91-32-47-1	25104,0960	One Point CrossOver Mutation
6	kroB100	1-21-90-12-98-32-59-29-8-99-76-97-91-28-11-93-85-73-53-40-39-70-67-5-62-26-100-56-72-37-65-79-81-47-23-22-88-54-52-89-43-50-55-77-24-18-96-19-92-44-41-17-36-45-78-13-63-31-48-51-82-42-2-16-14-64-33-15-6-4-83-87-60-74-66-94-57-7-84-58-34-27-61-35-71-9-25-46-20-38-80-30-69-75-86-49-68-10-3-95-1	24983,7871	One Point CrossOver Mutation
7	kroC100	1-53-40-12-46-29-24-32-61-26-7-82-78-9-37-18-49-93-4-60-14-99-19-92-10-36-57-45-66-16-51-63-44-48-84-11-52-96-87-97-81-33-100-74-69-3-73-59-41-89-21-23-70-76-91-94-95-50-72-39-71-86-5-43-56-38-28-88-98-58-34-90-25-17-8-22-83-62-75-6-54-2-35-68-30-77-80-65-31-47-67-55-42-64-20-79-13-15-27-85-1	22541,8994	One Point CrossOver Mutation
8	kroD100	1-27-13-79-88-24-41-84-73-15-69-63-39-71-4-93-44-60-53-5-62-35-37-19-51-30-100-2-48-92-55-87-26-10-21-56-95-28-12-47-9-20-25-82-18-6-40-83-3-59-85-64-77-43-91-54-16-96-86-14-68-33-45-99-31-57-72-78-65-80-76-7-75-67-36-29-58-22-42-23-74-61-46-52-8-66-38-89-70-49-98-97-32-94-17-90-11-34-81-50-1	23593,7454	Individual Mutation
9	kroE100	1-83-73-16-47-3-46-26-79-61-71-81-89-52-14-44-72-68-38-33-22-39-86-43-19-94-18-24-29-100-78-53-77-85-96-11-2-4-70-37-34-6-55-91-93-92-82-67-27-87-41-75-13-66-30-90-42-5-48-31-9-57-80-10-60-50-97-65-8-25-54-45-99-17-84-49-20-15-56-23-28-36-95-64-62-58-59-98-32-63-35-40-88-12-76-7-21-51-69-74-1	24428,2740	Individual Mutation
10	pr76	1-23-22-21-24-46-45-25-3-4-20-19-31-30-26-27-29-32-33-28-43-42-54-53-44-48-47-69-68-70-67-50-49-52-55-56-51-66-65-71-72-73-64-63-62-61-57-58-59-60-41-40-39-38-34-35-36-37-18-17-16-15-13-14-74-12-11-9-10-5-6-7-8-2-75-76-1	116305,6134	One Point CrossOver Mutation

Setelah dilakukan perbandingan, maka didapatkan bahwa performa solusi rute yang dihasilkan dari algoritma yang dikembangkan dengan mengacu pada *Genetic Algorithm* mampu memperbaiki solusi yang sebelumnya telah dihasilkan oleh *Savings Algorithm* sebagaimana tercantum pada tabel 4 berikut:

Tabel 4: Perbandingan Performa Algoritma

No	*Instance*	Panjang Rute dari *Savings Algorithm*	Panjang Rute dari Algoritma Berdasarkan *GA*	Selisih	Tingkat Perbaikan Rute
1	berlin52	8381,9068	8196,3654	185,5414	2,21%
2	eil51	466,0787	459,2792	6,7995	1,46%
3	eil76	590,6276	586,121	4,5066	0,76%
4	eil101	735,6019	725,9273	9,6746	1,32%
5	kroA100	25292,1927	25104,0960	188,0967	0,74%
6	kroB100	25444,7490	24983,7871	460,9619	1,81%
7	kroC100	22916,1169	22541,8994	374,2175	1,63%
8	kroD100	23761,2388	23593,7454	167,4934	0,70%
9	kroE100	24479,0259	24428,2740	50,7519	0,21%
10	pr76	117072,8770	116305,6134	767,2636	0,66%

SIMPULAN

Pendekatan *heuristic* seperti *Savings Algorithm* tidak selalu memberikan solusi yang optimal. Peluang perbaikan atas solusi dari pendekatan *heuristic* tersebut dapat diperoleh menggunakan pendekatan *metaheuristic* seperti *Genetic Algorithm.* Dalam penelitian ini, algoritma yang dikembangkan dengan mengacu pada *Genetic Algorithm* mampu memperbaiki solusi rute yang dihasilkan dari *Savings Algorithm* dengan batasan 10 *instance*. Meskipun begitu, algoritma yang dikembangkan ini perlu terus diujikan pada jenis *instance* lain guna mengukur konsistensi atas performanya.

DAFTAR PUSTAKA

Awad, H., Elshaer, R., Abdelmo'ez, A. & Nawara, G. (2018). An Effective Genetic Algorithm for Capacitated Vehicle Routing Problem, *International Conference on Industrial Engineering and Operations Management Bandung, Indonesia, March 6-8, 2018.*

Applegate, D.L., Bixby, R.E., Chvatal, V., & Cook, W.J. (2006). *The Traveling Salesman Problem A Computational Study*, Princeton University Press – New Jersey

Bhagya, T.G. (2021). Algoritma Genetik Pada Penjadwalan Transportasi Kapal Laut (Studi Kasus PT. Pelni). *Journal of Industrial and Manufacture Engineering*, 5(2), 81-92.

Chiarandini, M. (2020). *Construction Heuristics for Traveling Salesman Problem*, University of Southern – Denmark

Clarke, G., & Wright, J.W. (1964). Scheduling of Vehicles from A Central Depot to A Number of Delivery Points, *Operations Research*

Frutos, M., & Tohme, F. (2012). A New Approach to The Optimization of The CVRP through Genetic Algorithm, *American Journal of Operations Research*

Hansen, J.S. (2011). *GNU Octave Beginner's Guide*, PACKT – Birmingham

Kramer, O. (2017). *Genetic Algorithm Essentials*, Springer – Switzerland

Labadie, N., Prins, C., & Prodhon, C. (2016). *Metaheuristics for Vehicle Routing Problem*, ISTE – London

Lavrov, M. (2020). *The Traveling Salesman Problem*, University of Illinois at Urbana-Champaign - USA

Lysgaard, J. (2007). *Clarke & Wright's Savings Algorithm*, Department of Management Science and Logistics – The Aarhus School of Business – Denmark

Reinelt, G. (1995). *TSPLIB 95*, Institut fur Angewandte Mathematik – Universitat Heidelberg – Germany

Sanggala, E., Dimyati, T.T., & Yogaswara, Y. (2021). Genetic Algorithm Untuk Memperbaiki Rute Travelling Salesman Problem Yang Dihasilkan Dari Nearest Neighbour, *Jurnal Logistik Bisnis – Politeknik POS Indonesia*

Talbi, E. (2009). *Metaheuristics from Design to Implementation*, John Wiley & Sons – USA

http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/, diakses 2 Januari 2023