GENETIC ALGORITHM UNTUK MEMPERBAIKI RUTE *TRAVELLING SALESMAN PROBLEM* YANG DIHASILKAN DARI *NEAREST NEIGHBOUR*

Ekra Sanggala 1), Tjutju Tarliah Dimyati 2), Yogi Yogaswara 3)
[1]D4 Logistik Bisnis, Politeknik POS Indonesia
email: ekrasanggala@mail.ru
[2]Program Magister Teknik Industri, Universitas Pasundan
email: tjutjutarliah@unpas.ac.id
[3]Program Magister Teknik Industri, Universitas Pasundan
email: yogiyoga@unpas.ac.id

Abstrak

Travelling Salesman Problem merupakan permasalahan penentuan rute terpendek yang diawali dari titik start untuk mengunjungi sekumpulan titik tepat sekali dan diakhiri dengan kembali ke titik start. Jika sebuah TSP mempunyai banyak titik, maka akan menjadi sebuah NP-Hard Problem. Algoritma yang bekerja berdasarkan heuristic dan metaheuristic dapat menjadi sebuah solusi utuk menyelesaikan NP-Hard Problem. Nearest Neighbour (NN) merupakan salah satu algoritma yang bekerja berdasarkan heuristic. Dikarenakan algoritma ini bekerja berdasarkan heuristic tentunya solusi yang dihasilkan belum tentu solusi yang terbaik, oleh karena itu diharapkan solusi yang dihasilkan masih dapat diperbaiki sehingga diperoleh solusi yang lebih baik. Genetic Algorithm (GA) merupakan sebuah metaheuristic yang dapat diaplikasikan pada berbagai permasalahan optimasi, termasuk TSP. Dalam tulisan ini akan dibahas mengenai GA untuk memperbaiki rute TSP yang dihasilkan dari NN. Pada pengujian terhadap 10 instance, dapat diketahui bahwa algoritma berdasarkan GA dapat memperbaiki rute TSP yang dihasilkan dari NN.

Kata Kunci: *Genetic Algorithm, Nearest Neighbour, Travelling Salesman Problem*

1. PENDAHULUAN

Travelling Salesman Problem (*TSP*) merupakan permasalahan penentuan rute terpendek yang diawali dari titik *start* untuk mengunjungi sekumpulan titik tepat sekali dan diakhiri dengan kembali ke titik *start*. Pada dunia nyata banyak permasalahan yang dapat didefinisikan sebagai *TSP*, diantaranya adalah rute perjalanan turis, rute bus sekolah, rute pengiriman barang dan lain-lain. Jika sebuah *TSP* mempunyai banyak titik, maka akan menjadi sebuah *NP-Hard Problem* dimana untuk memperoleh solusi terbaiknya diperlukan waktu perhitungan yang tidak wajar [1]. Algoritma yang bekerja berdasarkan *heuristic* dan *metaheuristic* dapat menjadi sebuah solusi untuk menyelesaikan *NP-Hard Problem* dengan waktu perhitungan yang wajar walaupun solusi yang dihasilkan belum tentu merupakan solusi yang terbaik [2].

Nearest Neighbour (*NN*) merupakan salah satu algoritma yang bekerja berdasarka *heuristic*. Dalam menyelesaikan *TSP,* cara kerja dari *NN* adalah memilih titik terdekat dari titik terakhir yang dikunjungi dan belum termasuk ke dalam rute, untuk dimasukan ke dalam rute [2]. Algoritma ini sering digunakan karena sangat mudah penggunaannya. Dikarenakan algoritma ini bekerja berdasarkan *heuristic* tentunya solusi yang dihasilkan belum tentu solusi yang terbaik, oleh karena itu diharapkan solusi yang dihasilkan masih dapat diperbaiki sehingga diperoleh solusi yang lebih baik.

Genetic Algorithm (*GA*) merupakan sebuah *metaheuristic* yang dapat diaplikasikan pada berbagai permasalahan optimasi, termasuk *TSP*. Evolusi merupakan dasar dari *GA*. Dengan adanya banyak jenis spesies yang mampu beradaptasi dengan lingkungannya dan mampu bertahan hidup merupakan alasan yang baik untuk mengakui kekuatan dari evolusi. *GA* diawali dengan sekumpulan kandidat solusi yang mewakili sebuah solusi untuk masalah optimasi yang akan diselesaikan. Sekumpulan kandidat solusi ini dikenal sebagai populasi, sedangkan sebuah kandidat solusi disebut sebagai individu. Sebuah

kandidat solusi ini tersusun dari sekumpulan simbol, bisa berupa vektor atau *bit string* [3].

Dalam tulisan ini akan dibahas mengenai *GA* untuk memperbaiki rute *TSP* yang dihasilkan dari *NN*. Rute *TSP* yang dihasilkan dari *NN* akan menjadi salah satu kandidat solusi pada *GA*, yang kemudian akan mengalami proses evolusi. Dengan adanya proses evolusi ini, diharapkan dapat diperoleh solusi yang lebih baik lagi.

2. METODE PENELITIAN

Secara garis besar langkah-langkah dalam penelitian ini adalah sebagai berikut ini:

1. Menyiapkan *Instance*
   Instance diperlukan untuk menguji algoritma berdasarkan *GA* untuk memperbaiki rute *TSP* yang dihasilkan dari *NN*. *Instance* yang akan digunakan adalah sebagai berikut ini:
   a) berlin52
   b) eil51
   c) eil76
   d) eil101
   e) kroA100
   f) kroB100
   g) kroC100
   h) kroD100
   i) kroE100
   j) pr76

   Seluruh *instance* tersebut dapat diperoleh di *website* http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html.

2. Menyelesaikan *instance* dengan *NN*
   Seluruh *instance* yang digunakan diselesaikan oleh *NN*, kemudian rute yang dihasilkan akan menjadi salah satu kandidat solusi pada algoritma berdasarkan *GA*.

3. Mengembangkan Algoritma Berdasarkan *GA*
   Algoritma yang dikembangkan berdasarkan *GA* ini dalam langkah-langkahnya akan terdapat *initial solution, crossover*, *mutation* dan *selection*.

4. Menguji Algoritma Berdasarkan GA
   Algoritma yang telah dikembangkan pada langkah sebelumnya, diujikan pada *instance* agar dapat dilihat performansinya dalam memperbaiki rute yang dihasilkan dari *NN*.

3. HASIL DAN PEMBAHASAN

Instance yang digunakan merupakan *instance* yang sering digunakan oleh para peneliti untuk menguji algoritma yang dikembangkannya. Pada tabel 1 dapat dilihat data dari *instance* yang akan digunakan:

Tabel 1: *Instance*

No	*Instance*	Jml Titik	Author
1	berlin52	52	Groetschel
2	eil51	51	Christofides & Eilon
3	eil76	76	Christofides & Eilon
4	eil101	101	Christofides & Eilon
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

Pada tabel 2 dapat dilihat rute yang dihasilkan dari *NN* untuk setiap *instance*.

Tabel 2: Rute yang Dihasilkan dari *NN*

No	*Instance*	Rute	Panjang Rute
1	berlin52	1-22-49-32-36-35-34-39-40-38-37-48-24-5-15-6-4-25-46-44-16-50-20-23-31-18-3-19-45-41-8-10-9-43-33-51-12-28-27-26-47-13-14-52-11-29-30-21-17-42-7-2-1	8980,9183
2	eil51	1-32-11-38-5-49-9-50-16-2-29-21-34-30-10-39-33-45-15-44-37-17-4-18-47-12-46-51-27-48-6-14-25-13-41-19-42-40-24-23-7-26-8-31-28-3-20-35-36-22-43-1	513,6100
3	eil76	1-73-62-28-74-30-2-68-75-76-67-34-46-8-35-7-53-14-19-54-13-27-52-45-29-48-47-21-36-71-60-70-20-37-5-15-57-4-26-12-40-17-51-6-33-63-16-3-44-32-9-39-72-58-10-38-65-11-66-59-31-25-50-18-24-49-23-56-41-42-43-22-61-69-64-55-1	711,9933
4	eil101	1-69-27-101-53-58-40-21-73-72-74-22-75-56-39-23-67-25-55-54-80-68-77-3-79-33-81-9-51-20-30-70-31-88-7-82-48-47-36-49-19-11-62-10-90-32-63-64-46-8-45-17-84-5-60-83-18-52-89-6-94-95-97-92-59-99-96-93-85-91-100-37-98-61-16-44-14-42-87-2-57-15-43-38-86-13-26-12-76-50-28-4-24-29-78-34-35-71-66-65-41-1	825,2423
5	kroA100	1-63-6-49-90-10-84-72-21-74-59-17-15-11-32-91-98-23-45-47-93-28-67-58-61-25-81-69-73-50-44-2-54-40-64-68-85-39-30-96-78-52-5-37-33-76-13-95-82-48-100-71-41-14-3-46-29-34-83-55-12-27-86-35-20-57-7-9-87-51-77-60-62-43-92-8-42-89-31-80-56-97-75-19-53-79-18-24-38-36-99-94-22-16-88-70-66-65-4-26-1	26856,3886

Tabel 2: Rute Yang Dihasilkan dari *NN* (Lanjutan)

No	*Instance*	Rute	Panjang Rute
6	kroB100	1-95-98-12-71-27-61-35-94-57-34-7-84-58-52-54-43-89-87-60-74-66-4-83-6-15-33-82-64-14-42-2-13-78-17-45-36-96-92-19-44-41-18-24-77-16-50-55-22-23-88-37-72-65-79-81-47-56-100-26-69-62-5-67-40-39-70-53-73-85-93-11-3-28-91-97-99-8-29-32-59-76-90-21-86-49-30-75-80-20-38-46-25-9-68-10-63-31-48-51-1	29155,0437
7	kroC100	1-53-85-27-15-13-79-64-20-55-42-67-31-47-6-54-75-22-8-17-25-90-34-58-98-88-28-39-38-71-56-5-43-86-50-72-83-62-2-35-68-30-89-41-59-3-73-69-74-57-36-100-33-45-81-97-10-92-99-19-14-4-60-93-49-18-29-37-61-26-78-9-82-7-32-24-46-12-40-65-80-77-23-70-76-91-94-95-21-66-44-63-16-51-48-84-11-52-96-87-1	26327,3621
8	kroD100	1-50-34-81-38-66-8-52-46-61-74-7-75-67-36-29-58-22-42-23-11-90-17-94-32-97-98-49-70-89-27-24-88-79-13-41-84-15-63-54-91-77-43-71-4-44-93-60-53-5-3-83-40-6-59-85-18-25-20-9-10-26-87-55-92-19-37-51-100-2-48-30-35-62-64-16-96-86-65-14-68-33-45-99-31-78-80-76-72-57-69-73-39-82-47-21-56-95-28-12-1	26950,4612
9	kroE100	1-74-21-7-76-12-32-58-63-88-35-40-99-45-54-25-97-80-57-10-60-50-49-56-15-20-23-64-95-36-59-98-51-69-47-83-73-16-61-71-89-81-79-26-46-3-14-52-44-72-68-38-33-48-90-42-5-8-65-31-66-30-93-91-55-6-34-37-85-96-77-70-4-2-11-82-92-27-67-41-75-87-9-13-39-43-86-19-94-18-24-29-100-78-53-22-17-84-62-28-1	27587,1875
10	pr76	1-2-23-22-21-25-24-46-45-44-48-47-69-68-67-50-49-52-53-54-42-43-28-29-30-31-19-20-5-6-7-8-9-10-11-12-13-14-15-16-17-18-37-36-35-34-40-41-60-59-58-57-63-64-62-61-55-56-51-66-65-71-72-73-39-38-32-33-27-26-4-3-75-76-74-70-1	153461,9225

Algoritma untuk menyelesaikan *TSP* yang dikembangkan berdasarkan *GA* mempunyai beberapa *parameter* sebagai berikut ini:

1. *Mutated Individual Size*
   Parameter ini menyatakan berapa banyak individu yang dilahirkan dari hasil *mutation* individu di *population*. Pada penelitian ini, nilai untuk *parameter* ini sama dengan jumlah individu dalam *population*, yaitu 30 individu.

2. *Max Mutated Individual Pair Gen*
   Parameter Max Mutated Individual Pair Gen merupakan jumlah maksimal pasangan gen yang akan mengalami *mutation*. Pada penelitian ini, nilai untuk *parameter* ini sama dengan 45% dari banyaknya titik pada *instance*. Jika nilai yang dihasilkan bukan merupakan bilangan bulat, maka akan dibulatkan ke bawah.

3. *One Point Crossover Couple Size*
   Parameter ini menyatakan berapa banyak pasangan yang akan melakukan *crossover*. Pada penelitian ini, nilai untuk *parameter* ini adalah 50% dari jumlah individu dalam *population*. Jika nilai yang dihasilkan bukan merupakan bilangan bulat, maka akan dibulatkan ke atas. Dikarenakan jumlah individu dalam *population* adalah 30 individu, maka nilai *parameter* ini adalah 15 pasangan.

4. *Max Mutated One Point Crossover Individual Pair Gen*
   Pada algoritma yang dikembangkan ini, individu-individu yang dihasilkan dari *crossover* akan melahirkan individu dari hasil *mutation*. Pada penelitian ini, nilai untuk *parameter* ini sama dengan 45% dari banyaknya titik pada *instance*. Jika nilai yang dihasilkan bukan merupakan bilangan bulat, maka akan dibulatkan ke bawah.

5. *Survivor Size*
   Sejumlah individu dari suatu generasi akan mengalami proses *selection*, yaitu proses memilih sejumlah individu yang akan hidup pada generasi selanjutnya. Pada proses pemilihan ini seluruh individu akan diurutkan mulai dari individu dengan rute terpendek. *Parameter Survivor Size* akan menentukan seberapa banyak individu yang akan terpilih. Pada penelitian ini, nilai untuk *parameter* ini sama dengan $\frac{5}{6}$ dari jumlah individu dalam *population*. Jika nilai yang dihasilkan bukan merupakan bilangan bulat, maka akan dibulatkan ke atas.

6. *Random Individual Size*
   Karena nilai *parameter Survivor Size* lebih kecil dari jumlah individu dalam *population* maka kekurangan jumlah individu ini akan diisi oleh individu dari hasil *random*. Banyaknya individu dari hasil *random* ini sama dengan jumlah individu dalam *population* dikurangi nilai *parameter Survivor Size*. Nilai yang dihasilkan disimpan dalam *parameter Random Individual Size*.

7. *Max Repeat Shortest Length*
   Parameter *Max Repeat Shortest Length* merupakan jumlah maksimal generasi yang harus dilalui oleh suatu individu terbaik. Jika suatu individu mampu selalu menjadi yang terbaik dan mampu mencapai nilai *Max Repeat Shortest Length*, maka algoritma selesai dan individu tersebut terpilih sebagai rute terpendek yang mampu diperoleh. Pada penelitian ini, nilai untuk *parameter* ini sama dengan 250 generasi.

Algoritma yang telah dikembangkan diterjemahkan ke dalam bahasa pemrograman, sehingga penyelesaian *TSP* dapat dilakukan dengan bantuan komputer. Penulisan program dilakukan di *GNU Octave*. *GNU Octave* merupakan alat multifungsi untuk melakukan analisis numerik [4].

Pada tabel 3 dapat dilihat rute yang dihasilkan dari algoritma yang dikembangkan untuk setiap *instance*.

Tabel 3: Rute yang Dihasilkan dari Algoritma yang Dikembangkan berdasarkan GA

No	*Instance*	Rute	Panjang Rute
1	berlin52	1-22-32-49-36-35-34-39-40-38-37-48-24-5-15-6-4-25-46-44-16-50-20-23-31-18-3-45-19-41-8-9-10-43-33-51-12-28-27-26-47-13-14-52-11-29-30-21-17-42-7-2-1	8802,5465
2	eil51	1-32-11-38-5-49-9-50-16-2-29-21-34-30-10-39-33-45-15-44-37-17-4-18-47-12-46-51-27-6-23-14-25-13-41-19-42-40-24-43-7-26-8-31-28-3-20-35-36-22-48-1	488,0761
3	eil76	1-73-62-28-74-30-2-68-75-76-67-34-46-8-35-7-53-14-19-54-13-52-27-45-29-48-47-21-36-71-60-70-20-37-5-15-57-4-26-12-40-17-51-6-33-63-16-50-44-32-9-39-72-58-10-38-65-59-66-11-31-25-55-18-24-49-23-56-41-64-42-22-61-69-3-43-1	683,3523
4	eil101	1-69-27-101-53-58-40-21-73-72-74-22-75-41-39-23-67-25-55-54-80-68-77-3-79-33-81-9-51-20-30-70-31-88-7-82-48-47-36-49-19-11-62-10-90-32-63-64-46-8-45-17-84-5-60-83-18-52-89-6-94-95-97-92-59-99-96-93-85-91-100-37-98-61-16-44-14-42-87-2-57-15-43-38-86-13-26-12-76-50-28-4-56-29-78-34-35-71-66-65-24-1	811,0424
5	kroA100	1-63-6-49-90-10-84-72-21-74-59-17-15-11-32-91-98-23-45-47-93-28-67-58-61-25-81-69-73-50-44-2-54-40-64-68-85-39-30-96-78-52-5-37-33-76-13-95-82-48-100-71-41-14-3-46-29-43-83-55-12-27-86-35-34-57-7-9-87-51-77-60-62-20-92-8-42-89-31-80-56-97-75-19-53-79-18-24-38-36-99-94-22-88-16-70-66-4-65-26-1	26209,6408

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

No	*Instance*	Rute	Panjang Rute
6	kroB100	1-95-98-12-71-27-61-35-94-57-34-7-84-58-54-52-43-89-87-60-74-66-4-83-6-15-33-82-64-14-42-2-13-78-17-45-36-96-92-19-44-41-18-24-77-16-50-55-22-23-88-37-72-65-79-81-47-56-100-26-69-62-5-67-40-39-70-53-73-85-93-11-3-28-91-97-99-8-29-32-59-76-90-21-86-49-30-75-80-20-38-9-25-46-10-68-63-31-48-51-1	29054,5222
7	kroC100	1-53-85-27-15-13-79-64-20-55-42-67-31-47-6-54-75-22-8-17-25-90-34-58-98-88-28-39-38-71-56-5-43-86-50-72-83-62-2-35-68-30-89-41-59-3-73-69-74-57-36-100-33-97-81-45-10-92-19-99-14-4-60-93-49-18-29-37-61-26-78-9-82-7-32-24-46-12-40-65-80-77-23-70-76-91-94-95-21-66-44-63-16-51-48-84-11-52-87-96-1	26035,9418
8	kroD100	1-50-34-81-38-66-8-52-46-61-74-7-75-67-36-29-58-22-42-23-11-90-17-94-32-97-98-49-70-89-27-24-88-79-13-41-84-15-63-54-91-77-43-71-4-44-93-60-53-5-3-83-40-6-59-85-18-25-20-9-10-26-87-55-92-19-37-51-100-2-48-30-35-62-64-16-96-86-65-14-68-33-45-99-31-78-80-57-72-76-69-73-39-82-47-21-95-56-28-12-1	26781,7577
9	kroE100	1-74-21-7-76-12-32-58-63-88-35-40-99-45-54-25-97-80-57-10-60-50-49-56-20-15-23-64-95-36-59-98-51-69-83-47-73-16-61-71-89-81-79-26-46-3-14-52-44-72-68-38-33-48-90-42-5-8-65-31-66-30-93-91-55-6-34-37-85-96-77-70-4-11-2-82-92-67-27-41-75-87-9-13-39-43-86-19-94-18-24-29-100-78-53-22-17-84-62-28-1	27480,9197
10	pr76	1-23-22-21-24-25-45-46-68-44-48-47-69-70-67-50-49-52-53-54-42-43-28-29-30-31-19-20-5-6-7-8-9-10-11-12-13-14-15-16-17-18-37-36-35-34-40-41-60-59-58-57-63-64-62-61-55-56-51-66-65-71-72-73-39-38-32-33-27-26-4-3-74-76-75-2-1	136593,1526

Pada tabel 4 dapat dilihat performansi algoritma yang dikembangkan berdasarkan *GA* dalam memperbaiki rute yang dihasilkan dari *NN*.

Tabel 4: Performansi Algoritma yang Dikembangkan Berdasarkan *GA*

No	*Instance*	Panjang Rute dari *NN*	Panjang Rute dari Algoritma Berdasarkan *GA*	Selisih	Tingkat Perbaikan Rute
1	berlin52	8980,9183	8802,5465	178,3718	1,99%
2	eil51	513,6100	488,0761	25,5339	4,97%
3	eil76	711,9933	683,3523	28,6410	4,02%
4	eil101	825,2423	811,0424	14,1999	1,72%
5	kroA100	26856,3886	26209,6408	646,7478	2,41%
6	kroB100	29155,0437	29054,5222	100,5215	0,34%
7	kroC100	26327,3621	26035,9418	291,4203	1,11%
8	kroD100	26950,4612	26781,7577	168,7035	0,63%
9	kroE100	27587,1875	27480,9197	106,2678	0,39%
10	pr76	153461,9225	136593,1526	16868,7699	10,99%

Pada tabel 4 terlihat bahwa untuk 10 *instance* yang digunakan, algoritma yang dikembangkan berdasarkan *GA* mampu memperbaiki seluruh rute yang dihasilkan dari *NN*

4. KESIMPULAN

Rute *TSP* yang dihasilkan dari algoritma dengan pendekatan *heuristic* belum tentu merupakan rute yang terbaik, sehingga diharapkan rute tersebut dapat diperbaiki agar diperoleh rute yang lebih baik. Perbaikan rute tersebut dapat dilakukan dengan menggunakan algoritma yang dikembangkan dengan pendekatan *metaheuristic*, salah satunya adalah *GA*.

Algoritma yang dikembangkan berdasarkan *GA* pada penelitian ini, telah berhasil memperbaiki rute yang dihasilkan dari *NN* pada 10 *instance*. Tentunya perlu juga diujikan pada *instance* lainnya agar dapat lebih diketahui lagi kualitas performansinya.

5. REFERENSI

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

2. Nacime Labadie, Christian Prins, and Caroline Prodhon, *Metaheuristics for Vehicle Routing Problems*, ISTE, London, 2016.

3. Oliver Kramer, *Genetic Algorithm Essentials,* Springer, Switzerland, 2017.

4. Jesper Schmidt Hansen, *GNU Octave Beginner's Guide*, PACKT, Birmingham, 2011.

5. http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html