MJI

Media Jurnal Informatika
Vol. 15, no.1, Juni 2023
http://jurnal.unsur.ac.id/mjinformatika

p-issn : 2088-2114
e-issn : 2477-2542

Random Nearest Neighbour Untuk Menyelesaikan *Russian TSP Instances*

Ekra Sanggala [1], Muhammad Ardhya Bisma [2]
Universitas Logistik & Bisnis Internasional [1,2]
ekrasanggala@mail.ru, bisma@ulbi.ac.id

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. Nearest Neighbour (NN) is a heuristic for solving the TSP. It chooses the nearest unvisited node until all nodes putted on the route. Sometimes there are many nearest unvisited node, so it will become problem for RNN to choose one of them. Random Algorithm can be a solution for solving this problem. Using Random Algorithm on NN, makes NN becomes Random Nearest Neighbour (RNN). Performance RNN on solving TSP must be tested on TSP Instance. Two important criterias on the test are solution route and CPU Time. Russian TSP Instances contains ten TSP Instances, on which RNN can be tested. The test result shows that RNN can improve the length of existing route rapidly.

Keywords:Travelling Salesman Problem, Random, Nearest Neighbour, Russian TSP Instances

Abstrak

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*. *Nearest Neighbour* (*NN*) merupakan salah satu algoritma yang bekerja berdasarkan *heuristic*. Dalam menyelesaikan *TSP*, cara kerja dari *NN* adalah memilih titik terdekat dari titik terakhir yang dikunjungi dan belum termasuk ke dalam rute, untuk dimasukkan ke dalam rute. Penentuan titik yang akan dikunjungi berikutnya, akan menjadi masalah jika terdapat 2 atau lebih pilihan titik, dikarenakan kesamaan jarak dari titik terakhir. Untuk menyelesaikan masalah tersebut algoritma *Random* dapat menjadi sebuah solusi.Dengan demikian algoritma ini dapat disebut dengan *Random Nearest Neighbour* (*RNN*). Kemampuan *RNN* dalam menyelesaikan *TSP* perlu diuji, agar dapat diketahui kehandalannya. Dua kriteria penting yang dinilai dalam pengujian ini adalah rute solusi yang dihasilkan dan waktu perhitungan (*CPU Time*). *Russian TSP Instances* merupakan *TSP Instances* yang dapat digunakan untuk menguji *RNN*. Hasil pengujian menunjukkan bahwa *RNN* dapat memperbaiki panjang rute yang secara cepat.

Kata kunci: Travelling Salesman Problem, Random, Nearest Neighbour, Russian TSP Instances

I. 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 untuk ditunggu [1]. Algoritma yang bekerja berdasarkan *heuristic* dan *metaheuristic* dapat menjadi sebuah solusi untuk menyelesaikan *NP-Hard Problem* dengan waktu perhitungan yang wajar untuk ditunggu, walaupun solusi yang dihasilkan belum tentu merupakan solusi yang terbaik [2].

Nearest Neighbour (*NN*) merupakan salah satu algoritma yang bekerja berdasarkan *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 penggunannya. Penentuan titik yang akan dikunjungi berikutnya, akan menjadi masalah jika terdapat 2 atau lebih pilihan titik, dikarenakan kesamaan jarak dari titik terakhir. Untuk menyelesaikan masalah tersebut, algoritma *Random* dapat menjadi sebuah solusi, tetapi dikarenakan ada kemungkinan jika memilih titik yang lain dapat menghasilkan rute yang lebih pendek, maka perlu dilakukan percobaan penyelesaian sebuah *TSP* dengan algoritma ini secara berulang-ulang sampai batas jumlah percobaan yang diinginkan. Dengan demikian algoritma ini dapat disebut dengan *Random Nearest Neighbour* (*RNN*).

Kemampuan *RNN* dalam menyelesaikan *TSP* perlu diuji, agar dapat diketahui kehandalannya. Dua kriteria penting yang dinilai dalam pengujian ini adalah rute solusi yang dihasilkan dan waktu perhitungan (*CPU Time*) [3]. Untuk menguji kemampuan *RNN* dalam menyelesaikan *TSP* maka diperlukan *TSP Instances* yang akan diselesaikan oleh *RNN*. *Russian TSP Instances*

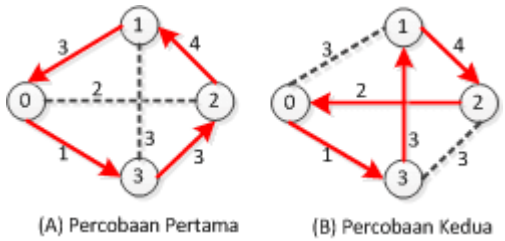
merupakan *TSP Instances* yang dapat digunakan untuk menguji sebuah algoritma [4]. Dengan demikian pada *paper* ini akan dibahas mengenai *Random Nearest Neighbour* dalam menyelesaikan *Russian TSP Instances*.

II. METODE PENELITIAN

Nearest Neighbour mungkin merupakan *heuristic* yang paling sederhana dan paling mudah. Prinsip dasar dari *Nearest Neighbour* adalah mengunjungi titik terdekat yang belum dikunjungi. Langkah-langkah dalam *Nearest Neighbour* kurang lebih seperti berikut ini [5]:

1. Tentukan titik *Start*.
2. Kunjungi titik terdekat yang belum dikunjungi.
3. Apa masih ada titik yang belum terkunjungi? Jika masih ada, kembali ke langkah kedua.
4. Kembali ke titik *Start*.

Dikarenakan algoritma *Random* akan digunakan untuk memilih 1 titik, jika terdapat 2 atau lebih pilihan titik, maka setiap percobaan penyelesaian sebuah *TSP* dapat menghasilkan rute yang berbeda-beda, oleh karena itu untuk menyelesaikan sebuah *TSP* perlu dilakukan berulang-ulang agar peluang mendapatkan rute yang lebih baik menjadi lebih besar. Penambahan algoritma *Random* pada *Nearest Neighbour*, menjadikan terbentuknya *Random Nearest Neighbour* (*RNN*). Ilustrasi cara kerja *RNN*, dapat dilihat pada Gambar 1.



Gambar 1: Ilustrasi Cara Kerja *RNN*

Secara garis besar, langkah-langkah pembentukan rute pada Gambar 1 adalah sebagai berikut ini:

(A) Percobaan Pertama
Titik *Start* adalah titik 0. Dari titik 0, titik terdekat yang belum terkunjungi adalah titik 3. Karena hanya ada 1 opsi pilihan titik, maka dipilihlah titik 3 untuk kunjungan berikutnya. Dari titik 3, titik terdekat yang belum terkunjungi adalah titik 1 dan 2. Karena ada 2 opsi pilihan titik, maka dipilih secara *random* antara titik 1 atau titik 2, terpilihlah titik 2 untuk kunjungan berikutnya. Dari titik 2, titik terdekat yang belum terkunjungi adalah titik 1. Karena hanya ada 1 opsi pilihan titik, maka dipilihlah titik 1 untuk kunjungan berikutnya. Karena semua titik sudah terkunjungi, maka dari titik 1 kembali ke titik 0. Dengan demikian rute yang terbentuk pada percobaan pertama ini adalah 0-3-2-1-0 dengan panjang rute 11 satuan panjang.

(B) Percobaan Kedua
Titik *Start* adalah titik 0. Dari titik 0, titik terdekat yang belum terkunjungi adalah titik 3. Karena hanya ada 1 opsi pilihan titik, maka dipilihlah titik 3 untuk kunjungan berikutnya. Dari titik 3, titik terdekat yang belum terkunjungi adalah titik 1 dan 2. Karena ada 2 opsi pilihan titik, maka dipilih secara *random* antara titik 1 atau titik 2, terpilihlah titik 1 untuk kunjungan berikutnya. Dari titik 1, titik terdekat yang belum terkunjungi adalah titik 2. Karena hanya ada 1 opsi pilihan titik, maka dipilihlah titik 2 untuk kunjungan berikutnya. Karena semua titik sudah terkunjungi, maka dari titik 2 kembali ke titik 0. Dengan demikian rute yang terbentuk pada percobaan pertama ini adalah 0-3-1-2-0 dengan panjang rute 10 satuan panjang.

Pada *RNN* yang digunakan pada *paper* ini, jumlah pengulangan tergantung pada kemampuan sebuah rute mampu bertahan sebagai yang terbaik selama sejumlah percobaan yang telah ditentukan. Sebagai contoh, ditentukan bahwa banyaknya percobaan yang harus dilalui oleh sebuah rute agar terpilih menjadi rute solusi adalah sebanyak 5 percobaan. Pada percobaan pertama diperoleh sebuah rute dengan panjang rute 1000 satuan panjang, maka rute tersebut untuk sementara terpilih menjadi rute terbaik. Rute tersebut harus mampu bertahan sebagai rute terbaik hingga percobaan kelima agar dapat terpilih menjadi rute solusi. Kemudian pada percobaan kedua diperoleh sebuah rute dengan panjang rute 1100 satuan panjang, pada percobaan ketiga diperoleh sebuah rute dengan panjang rute 1050 satuan panjang dan pada percobaan keempat diperoleh sebuah rute dengan panjang 950 satuan panjang, maka rute yang dihasilkan pada percobaan pertama hanya mampu bertahan sebagai rute terbaik sebanyak tiga percobaan, pada percobaan keempat, rute terbaik tergantikan oleh rute yang dihasilkan pada percobaan keempat, karena rute tersebut panjang rutenya lebih pendek daripada rute yang dihasilkan pada percobaan pertama. Rute yang dihasilkan pada percobaan keempat ini harus bertahan sebagai rute terbaik hingga percobaan kedelapan agar dapat terpilih menjadi rute solusi. Kemudan pada percobaan kelima diperoleh sebuah rute dengan panjang rute 1200 satuan panjang, pada percobaan keenam diperoleh sebuah rute dengan panjang 1100 satuan panjang, pada percobaan ketujuh diperoleh sebuah rute dengan panjang 950 satuan panjang dan pada percobaan kedelapan diperoleh sebuah rute dengan panjang 1000 satuan panjang., maka rute yang dihasilkan pada percobaan keempat mampu bertahan sebagai rute terbaik sebanyak lima percobaan, dengan demikian rute yang dihasilkan pada percobaan keempat terpilih sebagai rute solusi. Ilustrasi contoh diatas, dapat dilihat pada Tabel I.

Jml Percobaan Yang Harus Dilalui: 5

Percobaan	Panjang Rute	Panjang Rute Terbaik	Percobaan Yang Dilalui
1	1000	1000	1
2	1100	1000	2
3	1050	1000	3
4	950	950	1

Tabel I: Ilustrasi Pemilihan Rute Solusi

Jml Percobaan Yang Harus Dilalui: 5

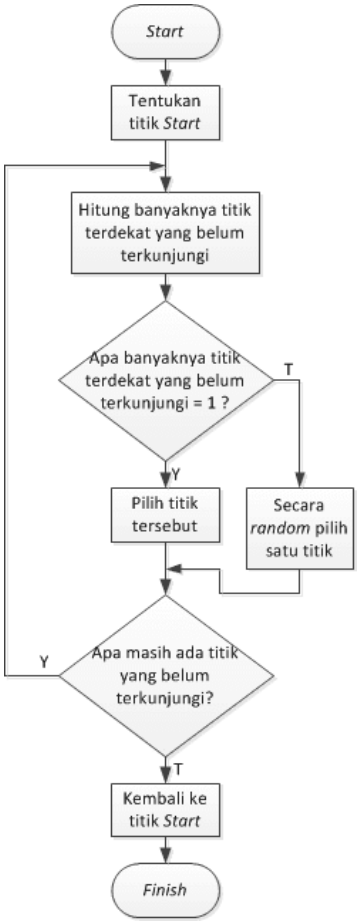
Percobaan	Panjang Rute	Panjang Rute Terbaik	Percobaan Yang Dilalui
5	1200	950	2
6	1100	950	3
7	950	950	4
8	1000	950	5

Tabel I: Ilustrasi Pemilihan Rute Solusi (lanjutan)

Dengan demikian langkah-langkah dalam *Random Nearest Neighbour* (*RNN*) dapat dijelaskan seperti berikut ini:

1. Tentukan banyaknya percobaan yang harus dilalui oleh sebuah rute terbaik agar terpilih menjadi rute solusi.
2. Tentukan titik *Start*.
3. Hitung banyaknya titik terdekat yang belum terkunjungi.
4. Jika terdapat 1 titik terdekat yang belum terkunjungi, pilih titik tersebut. Jika terdapat 2 atau lebih titik terdekat yang belum terkunjungi, pilih satu titik secara *random*.
5. Apa masih ada titik yang belum terkunjungi? Jika masih ada, kembali ke langkah ketiga.
6. Kembali ke titik *Start*.

Langkah-langkah dalam *RNN* dapat digambarkan dengan *flowchart* seperti pada Gambar 2.



Gambar 2: *Flowchart* dari *RNN*

Flowchart pada gambar 2 dapat diterjemahkan menjadi *pseudocode* seperti berikut ini:

Algoritma: *Random Nearest Neigbour*

Output : *return* titik-titik yang dikunjungi & *route length*

Initialize

Tentukan titik Start

Repeat

//R: banyaknya titik terdekat yang belum terkunjungi

Hitung R

If *R=1*

Kunjungi titik tersebut

else

Secara random kunjungi satu titik

end if

Until *semua titik terkunjungi*

Kembali ke titik Start

End

Russian TSP Instances merupakan sepuluh *TSP* yang dibuat berdasarkan sejarah *Russia* dan tempat-tempat di *Russia*. Sepuluh *TSP* yang terdapat di *Russian TSP Instances* adalah sebagai berikut ini [4]:

1. AK-47-TSP
   Memperingati senjata AK-47 yang ditemukan oleh *Mikhail Kalashnikov*.
2. Gagarin-108-TSP
   Memperingati *Yuri Gagarin* sebagai manusia pertama yang melakukan perjalanan ke luar angkasa dan mengorbit selama 108 menit.
3. Mendeleev-101-TSP
   Memperingati *Dmitri Mendeleev* sebagai penemu sistem periodik. Unsur dengan nomor atom 101 diberi nama *Mendelevium* untuk mengenang *Mendeleev*.
4. Petersburg-182-TSP
   Terdiri dari 182 area berpenduduk di wilayah *Saint Petersburg*.
5. Popov-250-TSP
   Memperingati *Alexander Popov* sebagai penemu radio. Pada tanggal 24 Maret 1896, dia berhasil mentransmisikan sinyal radio sejauh 250 meter.
6. Russia-10-Nodes-TSP
   Terdiri dari 10 kota berpenduduk terbanyak di *Russia*.
7. Russia-20-Nodes-TSP
   Terdiri dari 20 kota berpenduduk terbanyak di *Russia*.
8. Siege-of-Leningrad-872-TSP
   Memperingati 872 hari blokade *Leningrad* saat perang dunia kedua.

9. World-Cup-Stadium-12-TSP
   Memperingati *World Cup 2018* di *Russia*. Sebanyak 12 stadion digunakan pada *World Cup* tersebut.
10. Yashin-270-TSP
    Memperingati *Lev Yashin* sebagai kiper terhebat dalam sejarah sepakbola. Dia berhasil mendapatkan lebih dari 270 *clean sheets* dalam karirnya.

Data koordinat dan jumlah penduduk untuk membuat *Russian TSP Instances* bersumber dari https://www.geonames.org [6]. Pada *Russian TSP Instances*, untuk menghitung jarak antar titik digunakan *Haversine Formula*, yang hasilnya dibulatkan keatas sampai nilai satuan terdekat.

Haversine Formula merupakan persamaan yang digunakan untuk menghitung jarak antara dua titik di Bumi, bentuk persamaannya adalah sebagai berikut ini [7]:

$$a = sin^2\left(\frac{\Delta\phi}{2}\right) + \cos\phi_1 . \cos\phi_2 \ . sin^2\left(\frac{\Delta\lambda}{2}\right)$$

$$c = 2 \, . atan2\left[\sqrt{a}, \sqrt{(1-a)}\,\right]$$

$$d = R \, . c$$

Φ1 : *Latitude* titik pertama.
Φ2 : *Latitude* titik kedua.
λ1 : *Longitude* titik pertama.
λ2 : *Longitude* titik kedua.
ΔΦ: Φ1 - Φ2
Δλ : λ1 - λ2
R : Radius Bumi (6371 Km).

Untuk membantu menyelesaikan seluruh perhitungan digunakan perangkat lunak *GNU Octave*. *GNU Octave* merupakan alat multifungsi untuk melakukan analisis numerik. Berbagai alat yang tersedia di dalam *GNU Octave* adalah sebagai berikut ini [8]:

1. Sekumpulan *function* untuk menyelesaikan berbagai masalah.
2. Bahasa pemrograman yang dapat digunakan untuk mengembangkan kemampuan *GNU Octave*.
3. Berbagai fasilitas untuk melakukan *plotting*.

Nama *GNU Octave* diambil dari seorang ahli kimia yang bernama Octave Levenspiel. *GNU Octave* ini merupakan proyek resmi dari *GNU* dan lisensi *source code* berada di bawah *GNU General Public License* (*GPL*), sehingga siapa pun boleh menggunakannya untuk berbagai tujuan [9].

III. HASIL PENELITIAN

Agar *Russian TSP Instances* dapat digunakan maka disimpan dalam *file* bertipe *text* yang disusun secara terstruktur sehingga dapat dibaca dengan baik oleh *GNU Octave*.

Instance Gagarin-108-TSP akan digunakan sebagai contoh pengerjaan. Berikut ini merupakan contoh penggunaan *Haversine Formula* dalam menghitung jarak antara dua titik di Bumi:

Klushino (Lat: 55,66933; Long: 35,04579)
Moscow (Lat: 55,75222; Long: 37,61556)

$$a = sin^2\left(\frac{\Delta\phi}{2}\right) + \cos\phi_1 . \cos\phi_2 \ . sin^2\left(\frac{\Delta\lambda}{2}\right)$$

$$a = 0{,}000160$$

$$c = 2 \, . atan2\left[\sqrt{0{,}000160}, \sqrt{0{,}999840}\,\right]$$
$$c = 0{,}025308$$

$$d = 6371 \, . 0{,}025308$$
$$d = 161{,}234916 \approx 162 \, km$$

Pada *paper* ini banyaknya percobaan yang harus dilalui oleh rute terbaik agar terpilih menjadi rute solusi adalah 100 percobaan. Titik *start* pada *Instance Gagarin-108-TSP* adalah titik 1. Dari titik 1, titik terdekat adalah titik 95, maka titik 95 terpilih untuk dikunjungi. Dari titik 95, titik terdekat adalah titik 83, maka titik 83 terpilih untuk dikunjungi. Dari titik 83, titik terdekat adalah titik 101, maka titik 101 terpilih untuk dikunjungi. Dari titik 101, titik terdekat adalah titik 2, maka titik 2 terpilih untuk dikunjungi. Dari titik 2, titik terdekat adalah titik 81 dan titik 94, secara *random* terpilih titik 81 untuk dikunjungi. Cara ini terus berlanjut hingga seluruh titik terkunjungi, dan kembali titik *start*. Pada percobaan pertama ini dihasilkan sebuah rute dengan panjang rute 36738 km. Rute yang dihasilkan dari percobaan pertama ini, hanya mampu bertahan sebagai rute terbaik selama 5 percobaan, karena dari percobaan keenam dihasilkan sebuah rute dengan panjang rute 36730 km. Ternyata rute yang dihasilkan dari percobaan keenam ini berhasil bertahan sebagai rute terbaik selama 100 percobaan atau sampai dengan percobaan ke-105, maka rute yang dihasilkan dari percobaan keenam ini terpilih sebagai rute solusi.

Perhitungan yang dilakukan oleh *GNU Octave* untuk seluruh percobaan dari setiap *instance* menghasilkan 30 *files* bertipe *text* yang menyimpan secara detail hasil seluruh percobaan dari setiap *instance*. Dikarenakan keterbatasan tempat pada *paper* ini, maka seluruh hasil percobaan dari setiap *instance* tidak dapat ditampilkan. Bagi para pembaca yang membutuhkan *file-file* tersebut dapat langsung menghubungi penulis melalui *e-mail*. Untuk rute solusi dapat dilihat pada bagian lampiran.

Pada Tabel II disajikan panjang rute solusi dan *CPU Time* untuk setiap *instance*.

Instance	Banyaknya Percobaan	Panjang Rute Solusi (km)	*CPU Time (Detik)*
AK-47-TSP	100	27803	3,18242
Gagarin-108-TSP	105	36730	8,86086
Mendeleev-101-TSP	100	38067	8,12765
Petersburg-182-TSP	150	1465	32,76021
Popov-250-TSP	102	57144	48,15751

Tabel II: Panjang Rute Solusi dan *CPU Time*

Instance	Banyaknya Percobaan	Panjang Rute Solusi (km)	*CPU Time (Detik)*
Russia-10-Nodes-TSP	100	8671	1,57576
Russia-20-Nodes-TSP	100	12334	2,23082
Siege-of-Leningrad-872-TSP	189	3969	2360,51353
World-Cup-Stadium-12-TSP	100	8656	1,29481
Yashin-270-TSP	191	46112	106,31468

Tabel II: Panjang Rute Solusi dan *CPU Time*

IV. PEMBAHASAN

Penyajian hasil perhitungan secara garis besar perlu dilakukan sebelum dilakukan analisis terhadap hasil perhitungan yang diperoleh. Pada Tabel III disajikan tahapan-tahapan terpilihnya rute solusi untuk setiap *instance*.

AK-47-TSP

Panjang Rute Terbaik (Km)	Percobaan	Banyaknya Percobaan Yang Dilalui
27803	1	100

Gagarin-108-TSP

Panjang Rute Terbaik (Km)	Percobaan	Banyaknya Percobaan Yang Dilalui
36738	1	5
36730	6	100

Mendeleev-101-TSP

Panjang Rute Terbaik (Km)	Percobaan	Banyaknya Percobaan Yang Dilalui
38067	1	100

Petersburg-182-TSP

Panjang Rute Terbaik (Km)	Percobaan	Banyaknya Percobaan Yang Dilalui
1744	1	1
1620	2	3
1584	5	5
1567	10	5
1527	15	31
1469	46	5
1465	51	100

Popov-250-TSP

Panjang Rute Terbaik (Km)	Percobaan	Banyaknya Percobaan Yang Dilalui
57236	1	1
57174	2	1
57144	3	100

Tabel III: Tahapan Terpilihnya Rute Solusi

Russia-10-Nodes-TSP

Panjang Rute Terbaik (Km)	Percobaan	Banyaknya Percobaan Yang Dilalui
8671	1	100

Russia-20-Nodes-TSP

Panjang Rute Terbaik (Km)	Percobaan	Banyaknya Percobaan Yang Dilalui
12334	1	100

Siege-of-Leningrad-872-TSP

Panjang Rute Terbaik (Km)	Percobaan	Banyaknya Percobaan Yang Dilalui
4338	1	1
4204	2	1
4121	3	8
4119	11	16
4104	27	24
4092	51	8
4084	59	3
4067	52	25
4033	87	3
3969	90	100

World-Cup-Stadium-12-TSP

Panjang Rute Terbaik (Km)	Percobaan	Banyaknya Percobaan Yang Dilalui
8656	1	100

Yashin-270-TSP

Panjang Rute Terbaik (Km)	Percobaan	Banyaknya Percobaan Yang Dilalui
46938	1	5
46913	6	2
46739	8	1
46239	9	10
46188	19	15
46114	34	58
46112	92	100

Tabel III: Tahapan Terpilihnya Rute Solusi (lanjutan)

Pada Tabel III dapat dilihat bahwa *instance* AK-47-TSP, Mendeleev-101-TSP, Russia-10-Nodes-TSP, Russia-20-Nodes-TSP dan World-Cup-Stadium-12-TSP tidak pernah mengalami perbaikan panjang rute. Hal ini wajar terjadi karena pada *instance* tersebut *RNN* tidak pernah mengalami keharusan memilih satu titik dari beberapa opsi titik pilihan. Sementara *instance* lainnya mengalami perbaikan panjang rute karena beberapa kali *RNN* harus memilih satu titik dari beberapa opsi titik pilihan. Hal ini menunjukkan bahwa *RNN* mempunyai

kemampuan untuk melakukan perbaikan terhadap rute yang sudah ada.

V. KESIMPULAN

Random Nearest Neighbour dalam menyelesaikan *Travelling Salesman Problem* mampu untuk memperbaiki rute yang sudah ada dengan waktu perhitungan yang sangat cepat. Agar dapat diketahui lebih lanjut mengenai kehandalan *RNN* dalam menyelesaikan *TSP*, maka perlu dilakukan pengujian lebih lanjut pada berbagai *instance* yang lain.

Perlu juga dilakukan perbandingan antara *RNN* dengan algoritma lainnya, seperti *Savings Algorithm*, *Nearest Insertion*, *Lin-Kernighan Algorithm*, *Christofides Algorithm* dan lain-lain, dengan demikian dapat diketahui kehandalan *RNN* dibandingkan dengan algoritma lainnya.

Dikarenakan *RNN*, bekerja berdasarkan *heuristic* tentunya solusi yang dihasilkan belum tentu solusi yang terbaik. Untuk meningkatkan kehandalan *RNN* dapat dilakukan dengan melakukan *hybrid* antara *RNN* dengan berbagai *metaheuristic*, seperti *Genetic Algorithm, Ant Colony, Particle Swarm Optimization* dan lain-lain.

VI. REFERENSI

[1] Zefeng Wu, *Comparative Study of Solving Traveling Salesman Problem with Genetic Algorithm, Ant Colony Algorithm and Particle Swarm Optimization*, International Conference on Robotics, Systems and Vehicle Technology, Xiamen, 2020.

[2] R. Elshaer & H. Awad, *A Taxonomic Review of Metaheuristic Algorithms for Solving The Vehicle Routing Problem and Its Variants*, Computers & Industrial Engineering, 2020.

[3] J. Zhang, *Comparison of Various Algorithm Based on TSP Solving*, Journal of Physics, China, 2021.

[4] E. Sanggala & M.A. Bisma, *Perbandingan Savings Algorithm dengan Nearest Neighbour dalam Menyelesaikan Russian TSP Instances*, Jurnal Media Teknik & Sistem Industri, Volume 7, No 1, Maret 2023.

[5] C. Nilsson, *Heuristics for The Traveling Salesman Problem*, Linkoping University.

[6] https://download.geonames.org/export

[7] B. Idrizi, *Necessity for Geometric Corrections of Distances in Web and Mobile Maps*, International Conference on Cartography and GIS, Bulgaria, 2020.

[8] V. S. Ramos, *SIHR: a MATLAB/GNU Octave Toolbox for Single Image Highlight Removal*, The Journal of Open Source Software, 2020.

[9] S. Nagar, *Introduction to Octave for Engineers and Scientists*, Apress, Birmingham, 2018.

[10] K. Karagul & G. Gunduz, *A Novel Heuristic for The Travelling Salesman Problem: maxS*, Dokuz Eylul University Faculty of Engineering Journal of Science and Engineering, Turkey, 2022.

Lampiran 1

Rute Solusi yang Dihasilkan dari *Random Nearest Neighbour*

No	*Instance*	Rute
1	AK-47-TSP	1-23-4-39-38-29-15-25-8-31-5-11-46-12-34-21-14-41-42-9-20-18-7-16-33-6-45-22-2-40-30-32-17-47-44-10-19-13-36-37-28-35-3-43-26-27-24-1
2	Gagarin-108-TSP	1-95-83-101-2-94-81-55-40-30-65-45-72-22-93-63-64-48-97-98-3-99-35-56-100-60-44-58-47-88-17-32-69-33-67-20-18-7-107-102-104-16-59-13-90-10-71-19-82-57-103-50-74-62-89-37-36-28-75-12-106-11-5-49-14-21-34-85-9-70-42-6-86-41-87-52-105-76-61-43-53-78-46-54-31-8-4-23-96-92-29-39-38-15-77-80-25-51-66-91-26-27-73-24-84-108-79-68-1
3	Mendeleev-101-TSP	1-31-5-86-14-96-91-21-34-75-70-18-7-65-20-9-80-42-6-85-90-45-22-79-2-74-40-69-30-32-64-17-59-13-54-10-19-53-55-37-51-36-16-33-47-44-63-58-78-83-88-93-3-35-94-89-84-48-95-41-12-46-11-60-28-81-76-8-4-71-23-29-39-38-15-66-25-61-26-27-24-56-57-62-67-82-87-92-77-72-97-50-52-101-100-98-99-43-73-68-49-1
4	Petersburg-182-TSP	1-169-8-139-46-23-38-116-2-9-21-13-81-37-75-10-20-73-174-151-162-147-173-125-172-140-144-57-99-163-178-68-33-80-175-138-91-180-154-121-77-59-71-18-17-4-6-31-16-25-28-52-70-69-15-22-35-34-97-78-143-122-153-141-51-131-114-19-128-115-39-132-72-92-96-84-93-61-27-14-165-45-40-5-26-50-168-166-43-32-157-113-90-12-74-158-94-111-179-117-55-102-101-109-85-105-135-11-124-49-112-82-47-65-36-108-106-86-176-181-182-170-58-136-149-56-150-118-98-103-160-48-83-7-127-66-67-129-41-130-24-30-171-161-89-62-148-156-88-142-3-29-79-110-137-60-54-100-42-104-44-63-164-123-126-64-119-87-133-120-145-107-155-76-146-167-53-152-134-177-95-159-1
5	Popov-250-TSP	1-234-221-14-206-49-222-207-191-5-11-176-106-162-75-12-190-205-21-34-85-175-161-18-7-107-20-174-67-160-33-146-16-104-102-147-28-133-78-46-148-54-177-31-192-178-8-163-208-193-38-39-92-29-165-149-96-23-164-179-4-180-194-15-77-166-80-25-151-51-134-135-152-66-136-91-137-138-73-26-27-124-123-24-122-125-139-153-195-209-223-181-167-237-109-250-84-249-248-247-111-245-61-238-239-241-105-52-242-230-229-228-63-64-216-72-22-215-93-214-200-201-45-202-203-86-188-6-173-187-65-186-94-81-2-83-101-95-185-48-199-213-212-198-183-197-97-211-226-225-35-56-98-3-99-224-100-210-196-182-60-169-155-170-184-55-156-171-40-157-143-142-58-141-47-88-127-128-129-130-131-145-69-144-32-17-53-44-140-126-172-30-158-159-42-189-70-204-218-41-219-232-220-233-87-244-243-231-217-227-76-240-9-132-59-13-118-120-36-37-121-110-89-119-62-74-117-116-115-103-50-114-113-10-71-90-19-82-112-57-43-168-154-108-235-68-79-246-236-150-1
6	Russia-10-Nodes-TSP	1-5-8-6-10-4-7-3-9-2-1
7	Russia-20-Nodes-TSP	1-5-8-19-17-6-15-12-9-18-16-2-20-13-4-10-11-7-3-14-1
8	Siege-of-Leningrad-872-TSP	1-626-632-624-618-616-615-617-620-619-628-629-621-623-627-634-639-641-646-648-645-638-571-570-565-635-567-562-6-563-564-560-559-5-613-614-625-631-169-643-649-651-652-650-642-647-580-584-585-582-578-574-576-573-579-4-577-575-572-637-644-581-636-633-653-654-655-657-658-659-3-660-713-715-716-719-662-723-724-663-664-593-62-89-148-156-661-88-142-656-588-586-587-583-569-568-566-497-561-31-492-548-546-549-550-551-547-545-544-543-606-607-608-609-692-693-694-165-45-612-697-139-46-23-622-8-630-699-640-17-703-38-116-705-2-706-9-21-13-81-698-10-700-75-37-14-40-610-611-552-553-554-555-556-558-557-491-25-489-490-488-28-487-486-483-52-481-480-479-478-542-15-541-540-22-35-538-536-535-534-533-532-603-531-97-602-601-600-78-599-598-529-460-530-463-464-465-399-101-402-404-403-466-468-470-471-474-473-472-69-476-482-70-477-475-412-85-416-419-113-423-424-157-422-421-32-420-484-105-485-43-425-426-429-431-430-432-135-434-11-124-439-49-437-436-438-435-48-374-373-160-372-375-376-377-371-369-90-367-427-428-365-362-359-74-361-12-364-363-358-360-286-357-284-281-280-279-278-347-351-349-348-352-94-356-158-355-354-353-350-346-345-405-339-401-400-398-333-332-330-331-328-326-102-323-327-329-325-183-255-259-55-262-263-265-266-117-269-270-271-272-274-273-343-111-276-344-342-341-340-338-337-336-334-335-397-396-395-393-391-394-462-461-392-324-256-258-260-261-264-268-267-201-204-203-205-207-206-210-209-103-211-214-217-213-218-219-221-220-222-224-228-226-230-236-240-241-244-247-250-252-253-254-251-318-319-320-86-317-106-321-322-316-314-313-310-308-307-305-304-306-311-312-246-248-315-242-239-237-235-231-232-229-234-238-243-245-249-176-227-225-223-170-302-182-181-233-301-300-298-58-297-299-296-295-294-292-287-283-150-118-208-212-215-216-98-56-293-136-291-289-149-290-288-285-282-277-275-179-195-196-191-189-187-257-190-192-193-194-197-199-202-200-185-198-188-366-370-368-418-417-415-414-411-413-409-410-109-408-406-407-469-467-537-604-539-677-678-676-675-673-674-34-672-671-669-143-670-122-153-141-752-51-131-679-114-19-128-115-685-681-39-682-132-72-92-96-683-84-684-680-686-178-68-33-172-140-57-696-144-691-163-99-125-702-701-704-707-710-712-711-151-174-162-147-173-20-73-709-708-59-71-18-27-164-687-93-688-61-689-605-16-168-166-493-494-495-496-498-50-502-499-500-26-501-504-30-24-508-509-129-510-41-130-505-506-441-7-446-447-448-444-442-445-443-378-112-440-433-503-589-590-507-171-161-591-592-594-595-516-520-519-523-524-42-100-54-666-60-734-733-730-596-665-727-110-726-79-29-722-721-725-720-717-718-714-77-121-175-787-138-91-180-789-177-134-152-780-779-777-774-771-775-770-773-772-769-778-53-783-786-788-784-781-785-790-791-864-859-856-854-852-76-146-768-767-839-835-844-846-850-848-155-843-845-847-855-853-849-841-840-836-837-145-831-830-107-829-833-832-838-842-834-828-823-820-818-819-816-817-814-813-810-806-805-807-811-812-808-742-815-133-751-749-747-748-753-755-757-758-87-754-756-126-750-64-743-741-740-119-744-746-745-739-738-736-737-668-667-123-760-762-763-765-766-764-690-695-80-782-154-796-794-792-95-793-795-797-798-799-801-802-803-800-872-867-866-869-868-862-870-871-860-858-861-857-851-863-865-159-186-827-824-809-821-825-822-120-826-759-761-776-167-804-728-729-731-732-137-735-597-526-525-63-457-456-458-459-528-184-527-522-521-104-518-517-44-515-514-513-512-127-67-511-66-452-47-453-454-455-385-384-65-451-450-449-82-381-382-383-36-108-386-387-388-389-390-309-380-303-379-83-1
9	World-Cup-Stadium-12-TSP	1-3-7-10-8-9-5-6-4-11-2-12-1
10	Yashin-270-TSP	1-174-235-209-138-213-112-139-93-131-127-167-226-160-267-196-124-200-248-236-120-145-180-130-142-221-222-268-205-155-137-159-165-136-241-141-109-164-250-193-168-208-154-178-171-169-177-147-263-122-204-129-232-82-225-123-247-100-166-80-110-158-191-54-39-152-29-132-269-252-170-134-175-201-187-94-261-47-92-21-71-231-44-128-64-161-242-186-85-5-41-157-69-214-8-19-151-17-182-6-106-237-32-66-68-230-31-176-16-87-245-52-46-220-57-43-59-199-99-96-240-233-146-97-2-125-98-153-251-198-34-55-254-224-266-75-62-63-40-238-211-217-246-216-163-20-33-84-144-234-190-11-74-121-27-244-101-103-15-156-58-12-114-89-215-117-9-184-257-70-243-18-135-102-264-49-148-173-150-260-140-270-73-118-61-88-162-36-258-195-35-192-56-81-219-194-77-45-256-10-259-115-105-149-228-4-108-265-53-30-179-181-67-78-183-227-262-203-86-210-113-13-48-218-7-3-239-253-255-37-185-38-188-197-91-28-207-116-14-229-206-172-95-22-119-189-249-79-24-50-65-76-212-143-83-223-107-111-72-25-26-126-202-23-133-90-51-104-60-42-1