SAINTEKS: JURNAL SAIN DAN TEKNIK

Volume 6 Nomor 1 Tahun 2024

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.ac.id/index.php/sainteks

UICM – UNBAR
www.uicm.ac.id

Sainteks: Jurnal Sain dan Teknik
E-ISSN: 2685-8304
Volume 6 Nomor 1 Tahun 2024
http://ejournal.uicm.ac.id/index.php/sainteks/

Random Savings Algorithm for Solving Russian TSP Instances

Random Savings Algorithm Untuk Menyelesaikan *Russian TSP Instances*

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

Abstract: *Travelling Salesman Problem (TSP) is the problem for finding the shortest route starting from start node then visiting number of node exactly once and finally go back to start node. Savings Algorithm (SA) is a heuristic for solving TSP. In the Savings Algorithm, the first step that must be taken is to calculate the Savings for each pair of nodes. Then the Savings values that have been obtained are sorted from the largest Savings to the smallest Savings. Route is made by first inserting into the route the pair of nodes who has the highest Savings value. Sometimes there are many pairs of node who have the same Savings value, so it will become problem for SA to choose one of them. Random Algorithm can be a solution for solving this problem. Using Random on SA, makes SA become Random Savings Algorithm (RSA). Performance RSA on solving TSP must be tested on TSP Instance. Two important criterias on the test are solution route and CPU Time. Russian TSP Instances contain ten TSP Instances, on which RSA can be tested. The test result shows that RSA can improve the routes resulting from previous attempt.*
Keywords: Travelling Salesman, Random, Savings Algorithm, 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*. *Savings Algorithm* (*SA*) merupakan salah satu algoritma yang bekerja berdasarkan *heuristic*. Pada *Savings Algorithm*, langkah pertama yang harus dilakukan adalah menghitung *Savings* untuk setiap pasangan titik. Kemudian nilai-nilai *Savings* yang telah diperoleh, diurutkan mulai dari *Savings* tertinggi sampai ke *Savings* terendah. Pembentukan rute dilakukan dengan memasukkan ke dalam rute terlebih dahulu pasangan-pasangan titik yang memiliki nilai *Savings* lebih tinggi. Penentuan pasangan titik yang akan dimasukkan ke dalam rute akan menjadi masalah jika terdapat 2 atau lebih pilihan pasangan titik, dikarenakan kesamaan nilai *Savings*. Untuk menyelesaikan masalah tersebut, algoritma *Random* dapat menjadi sebuah solusi. Dengan demikian algoritma ini dapat disebut dengan *Random Savings Algorithm* (*RSA*). Kemampuan *RSA* 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*). *Russia TSP Instances* merupakan *TSP Instances* yang dapat digunakan untuk menguji *RSA*. Hasil pengujian menunjukkan bahwa *RSA* dapat memperbaiki rute yang dihasilkan dari percobaan sebelumnya.
Kata Kunci: *Travelling Salesman, Random, Savings Algorithm, Russian TSP Instances*

DOI: http://dx.doi.org/10.37577/sainteks.v%vi%i.654
Received: 02, 2024. Accepted: 03, 2024.
Published: 03, 2024

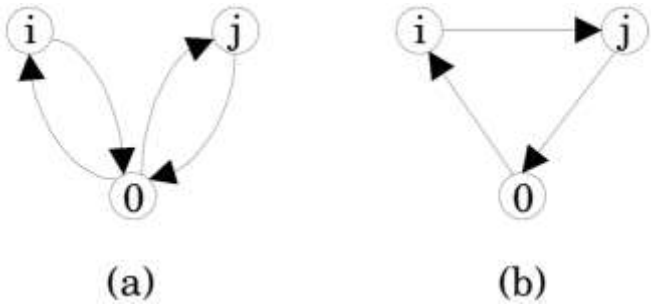
PENDAHULUAN

Seorang *salesman* ingin mengunjungi beberapa kota. Dia dapat memulai perjalanannya dari kota mana pun, lalu mengunjungi setiap kota lainnya tepat satu kali, kemudian kembali ke kota pertama. Dia ingin menemukan urutan kunjungan kota yang menghasilkan panjang rute perjalanannya seminimal mungkin. Jenis permasalahan ini seringkali didefinisikan sebagai *Travelling Salesman Problem* (*TSP*) (Shahab, 2019).

TSP dapat diaplikasikan pada berbagai bidang, termasuk logistik (rute bus sekolah, pengiriman paket), pengurutan *genome*, masalah pengeboran, *data clustering* dan lain-lain (Applegate et al., 2011). *TSP* merupakan sebuah dasar untuk berbagai masalah yang lebih besar. Sebagai contoh, pada *Capacitated Vehicle Routing Problem* (*CVRP*), terdapat sejumlah *customer* yang masing-masing mempunyai *demand* dan tersedia kendaraan dengan kapasitas yang lebih kecil dari jumlah *demand* seluruh *customer*, sehingga diperlukan lebih dari satu rute untuk melayani seluruh *customer*. Titik keberangkatan kendaraan untuk setiap rute adalah *depot*, kemudian mengunjungi setiap *customer* yang termasuk di dalam rute tepat sekali dan kembali ke *depot*. Dengan demikian *CVRP* terdiri dari beberapa *TSP* (Golden et al., 2023).

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 (Wu, 2020). 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 (Elshaer & Awad, 2020).

Salah satu algoritma yang populer untuk menyelesaikan *TSP* adalah *Savings Algorithm* (*SA*), algoritma tersebut merupakan sebuah *heuristic* (Kunnapapdeelert & Thawnern, 2021). Clarke dan Wright merupakan penemu *Savings Algorithm*, mereka menemukannya pada tahun 1964. Penghematan (*Savings*) jarak tempuh yang didapatkan dengan menggabungkan dua rute menjadi dasar bagaimana *Savings Algorithm* bekerja. Gambaran mengenai penggabungan tersebut dapat dilihat pada Gambar 1, dimana titik 0 merupakan depot (Lysgaard, 1997).



Gambar 1: Contoh Penggabungan 2 Rute (Lysgaard, 1997)

Pada gambar 1(a) terlihat bahwa titik i dan titik j dilayani oleh rute yang berbeda. Pada gambar 1(b) diperlihatkan bahwa titik i dan titik j dilayani oleh rute yang sama. Dapat diketahui bahwa jarak tempuh D_a pada gambar 1(a) adalah sebagai berikut ini (Jiménez Aguirre, 2022):

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

Sedangkan untuk jarak tempuh D_b pada gambar 1(b) dapat diketahui sebagai berikut ini (Jiménez Aguirre, 2022):

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

Nilai *Savings* diperoleh dengan mengurangkan jarak tempuh D_a dengan jarak tempuh D_b, sehingga diperoleh persamaan nilai *Savings* S_{ij} berikut ini (Jiménez Aguirre, 2022):

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

Dalam menggunakan *Saving Algorithm,* langkah awal yang perlu dilakukan adalah menghitung nilai *Savings* untuk setiap pasangan titik. Lalu nilai-nilai *Savings* yang telah didapatkan, disusun mulai dari *Savings* tertinggi sampai ke *Savings* terendah. Pembuatan rute

dibuat dengan memasukkan ke dalam rute terlebih dahulu pasangan-pasangan titik yang memiliki nilai *Savings* lebih tinggi (Jiménez Aguirre, 2022).

Cara kerja *Savings Algorithm* sangat sederhana sehingga *Savings Algorithm* ini sering digunakan. Dalam menentukan pasangan titik yang akan dimasukkan ke dalam rute, *Savings Algorithm* seringkali menghadapi adanya 2 atau lebih pilihan pasangan titik karena pasangan-pasangan titik tersebut mempunyai nilai *Savings* yang sama. Hal tersebut tentunya menjadi masalah bagi *Savings Algorithm*. Penentuan secara *Random* bisa menjadi solusi untuk menyelesaikan masalah tersebut, tetapi disebabkan adanya peluang jika memilih titik yang lain dapat memberikan rute yang lebih baik, tentunya perlu dilakukan percobaan penyelesaian sebuah *TSP* dengan algoritma ini secara berulang kali sampai batas jumlah percobaan yang ditentukan. Maka algoritma ini dapat disebut dengan *Random Savings Algorithm* (*RSA*).

Kemampuan *RSA* 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*) (Zhang, 2021). Untuk menguji kemampuan *RSA* dalam menyelesaikan *TSP* maka diperlukan *TSP Instances* yang akan diselesaikan oleh *RSA*. *Russian TSP Instances* merupakan *TSP Instances* yang dapat digunakan untuk menguji sebuah algoritma (Sanggala & Bisma, 2023). Maka pada *paper* ini akan diulas tentang *Random Savings Algorithm* dalam menyelesaikan *Russian TSP Instances*.

METODOLOGI

Secara umum tahapan-tahapan yang dilakukan adalah sebagai berikut ini:

1. Penentuan *Instance*
   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 (Sanggala & Bisma, 2023):
   a) AK-47-TSP
      Memperingati senjata AK-47 yang ditemukan oleh *Mikhail Kalashnikov*.

   b) Gagarin-108-TSP
      Memperingati *Yuri Gagarin* sebagai manusia pertama yang melakukan perjalanan ke luar angkasa dan mengorbit selama 108 menit.

   c) Mendeleev-101-TSP
      Memperingati *Dmitri Mendeleev* sebagai penemu sistem periodik. Unsur dengan nomor atom 101 diberi nama *Mendelevium* untuk mengenang *Mendeleev*.

   d) Petersburg-182-TSP
      Terdiri dari 182 area berpenduduk di wilayah *Saint Petersburg*.

   e) Popov-250-TSP
      Memperingati *Alexander Popov* sebagai penemu radio. Pada tanggal 24 Maret 1896, dia berhasil mentransmisikan sinyal radio sejauh 250 meter.

   f) Russia-10-Nodes-TSP
      Terdiri dari 10 kota berpenduduk terbanyak di *Russia*.

   g) Russia-20-Nodes-TSP
      Terdiri dari 20 kota berpenduduk terbanyak di *Russia*.

   h) Siege-of-Leningrad-872-TSP
      Memperingati 872 hari blokade *Leningrad* saat perang dunia kedua.

i) World-Cup-Stadium-12-TSP
Memperingati *World Cup 2018* di *Russia*. Sebanyak 12 stadion digunakan pada *World Cup* tersebut.

j) 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/ (*GeoNames*, n.d.). *Haversine Formula* digunakan untuk melakukan perhitungan jarak antar titik, yang hasil perhitungannya akan dibulatkan ke atas sampai nilai satauan terdekat.

2. Menghitung Jarak Antar Titik Pada Setiap *Russian-TSP-Instances*
Haversine Formula merupakan persamaan yang digunakan untuk menghitung jarak antara dua titik di Bumi, bentuk persamaannya adalah seperti berikut ini (Idrizi, 2020):

$$a = sin^2\left(\frac{\Delta\phi}{2}\right) + \cos\phi_1 . \cos\phi_2 \ . sin^2\left(\frac{\Delta\lambda}{2}\right)$$
$$c = 2 \, . atan2\left[\sqrt{a}, \sqrt{(1-a)}\right]$$

$$d = R \, . c$$

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)

Berikut ini merupakan contoh penggunaan *Haversine Formula* dalam menghitung jarak antara dua titik di Bumi:

Novosibirsk (Latitude: 55,04150; Longitude: 82,93460)
Yekaterinburg (Latitude:56,85190; Longitude: 60,61220)

$$a = sin^2\left(\frac{\Delta\phi}{2}\right) + \cos\phi_1 \ . \ \cos\phi_2 \ . \ sin^2\left(\frac{\Delta\lambda}{2}\right)$$
$$a = 0{,}011989$$

$$c = 2 \, . atan2\left[\sqrt{0{,}011989}, \sqrt{0{,}988011}\right]$$
$$c = 0{,}219429$$

$$d = 6371 \, . 0{,}219429$$
$$d = 1397{,}985 \ \approx 1398 \ km$$

Tabel 1: Contoh Matriks Jarak Pada Russia-10-Nodes-TSP

Jarak (Km)		Node									
Node	Kota	1	2	3	4	5	6	7	8	9	10
1	Moscow	0	635	2812	1418	402	857	2236	719	959	1496
2	Saint Petersburg	635	0	3105	1783	896	1420	2584	1200	1540	1912
3	Novosibirsk	2812	3105	0	1398	2412	2127	610	2115	3083	1363
4	Yekaterinburg	1418	1783	1398	0	1017	780	820	718	1773	196
5	Nizhniy Novgorod	402	896	2412	1017	0	526	1834	324	1054	1096
6	Samara	857	1420	2127	780	526	0	1520	296	994	765
7	Omsk	2236	2584	610	820	1834	1520	0	1527	2474	760
8	Kazan	719	1200	2115	718	324	296	1527	0	1151	778
9	Rostov-na-Donu	959	1540	3083	1773	1054	994	2474	1151	0	1740
10	Chelyabinks	1496	1912	1363	196	1096	765	760	778	1740	0

3. Menghitung *Savings*
   Berikut ini merupakan contoh menghitung *Savings* antara dua titik:
   Moscow – Saint Petersburg = 635 Km
   Moscow – Novosibirsk = 2812 Km

   Maka nilai *Savings* antara *Saint Petersburg* dengan *Novosibirsk* adalah sebagai berikut ini:

   $D_a = d_{0i} + d_{i0} + d_{0j} + d_{j0}$
   $D_a = 635 + 635 + 2812 + 2812 = 6894$ Km

   $D_b = d_{0i} + d_{ij} + d_{j0}$
   $D_b = 635 + 3105 + 2812 = 6552$ Km

   $S_{ij} = D_a - D_b = d_{i0} + d_{0j} - d_{ij}$
   $S_{ij} = 6894 - 6552 = 342$ Km

Tabel 2: Contoh Nilai *Savings* Pada Russia-10-Nodes-TSP

Node 1	Node 2	*Savings (Km)*
2	3	342
2	4	270
2	5	141
2	6	72
2	7	287
2	8	154
2	9	54
2	10	219
3	4	2832
3	5	802
3	6	1542
3	7	4438
3	8	1416
3	9	688
3	10	2945
4	5	803
4	6	1495
4	7	2834
4	8	1419
4	9	604
4	10	2718
5	6	733
5	7	804
5	8	797
5	9	307
5	10	802
6	7	1573
6	8	1280
6	9	822
6	10	1588
7	8	1428
7	9	721
7	10	2972
8	9	527
8	10	1437
9	10	715

4. Pengujian *Random Savings Algorithm*
   Random Savings Algorithm diujikan pada *Russian-TSP-Instances* agar dapat diketahui kehandalannya dalam menyelesaikan *TSP*. Pada *paper* ini banyaknya percobaan yang harus dilalui oleh sebuah rute agar terpilih menjadi rute solusi adalah 10 percobaan.

Untuk mempermudah dalam penyelesaian pengujian, digunakan *software GNU Octave*. *GNU Octave* merupakan *tool* dengan berbagai macam fungsi untuk melakukan analisis numerik. Macam-macam alat bantu yang terdapat di dalam *GNU Octave* adalah sebagai berikut ini (Ramos, 2020):

1. Berbagai macam *function* untuk menyelesaikan bermacam-macam kasus.
2. Bahasa pemrograman yang dapat dipakai untuk meningkatkan kemampuan *GNU Octave*.
3. Bermacam-macam fasilitas untuk menggambarkan *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 (Nagar & Nagar, 2018).

Listing Program yang ditulis untuk menjalankan *Random Savings Algorithm* terdiri dari 302 baris dan setiap kali *listing program* ini dieksekusi akan menghasilkan 70 *files* bertipe *text* yang menyimpan secara detail hasil seluruh percobaan dari setiap *instance*. Dikarenakan keterbatasan tempat pada *paper* ini, maka *listing program* dan seluruh hasil percobaan dari setiap *instance* tidak dapat ditampilkan. Bagi siapa pun yang memerlukan *file-file* tersebut dapat langsung mengontak penulis melalui *e-mail*.

Pseudocode dari *Random Savings Algorithm* adalah seperti berikut ini:

Algoritma: *Random Savings Algorithm*
Output: *return* titik yang dikunjungi & *route length*

Initialize
Hitung Jarak Antar Titik
Hitung Nilai Savings Antar Titik
Urutkan Nilai Savings Antar Titik Mulai Dari Nilai Savings Tertinggi Sampai Nilai Savings Terendah

Repeat
//R: banyaknya pasangan titik dengan nilai Savings tertinggi yang belum masuk ke dalam rute
Hitung R

if *R=1*
Masukkan pasangan titik tersebut ke dalam rute
else
Secara random pilih pasangan titik yang belum terkunjungi dengan nilai Savings Tertinggi
end if

Until *semua pasangan titik masuk ke dalam rute*

End

HASIL PENELITIAN DAN PEMBAHASAN

Pada Tabel 3 ditampilkan urutan-urutan penentuan rute solusi untuk setiap *instance*.

Tabel 3: Urutan Penentuan Rute Solusi

AK-47-TSP

Panjang Rute Yang Diperoleh (Km)	Percobaan	Banyaknya Percobaan Yang Dilalui
23874	1	3
23861	4	10
CPU TIME (Detik)	**43,6335**	

Gagarin-108-TSP

Panjang Rute Yang Diperoleh (Km)	Percobaan	Banyaknya Percobaan Yang Dilalui
30283	1	10
CPU TIME (Detik)	**84,1313**	

Mendeleev-101-TSP

Panjang Rute Yang Diperoleh (Km)	Percobaan	Banyaknya Percobaan Yang Dilalui
34366	1	3
34365	4	10
CPU TIME (Detik)	**100,6986**	

Petersburg-182-TSP

Panjang Rute Yang Diperoleh (Km)	Percobaan	Banyaknya Percobaan Yang Dilalui
1410	1	6
1402	7	10
CPU TIME (Detik)	**43,6335**	

Popov-250-TSP

Panjang Rute Yang Diperoleh (Km)	Percobaan	Banyaknya Percobaan Yang Dilalui
53536	1	1
53526	2	1
53033	3	2
52989	5	10
CPU TIME (Detik)	**391,6717**	

Russia-10-Nodes-TSP

Panjang Rute Yang Diperoleh (Km)	Percobaan	Banyaknya Percobaan Yang Dilalui
8059	1	10
CPU TIME (Detik)	**7,6128**	

Tabel 3: Urutan Penentuan Rute Solusi (Lanjutan)

Russia-20-Nodes-TSP

Panjang Rute Yang Diperoleh (Km)	Percobaan	Banyaknya Percobaan Yang Dilalui
10043	1	10
CPU TIME (Detik)	**16,5517**	

Siege-of-Leningrad-872-TSP

Panjang Rute Yang Diperoleh (Km)	Percobaan	Banyaknya Percobaan Yang Dilalui
4072	1	1
4041	2	5
4015	7	3
4013	10	10
CPU TIME (Detik)	**3776,1758**	

World-Cup-Stadium-12-TSP

Panjang Rute Yang Diperoleh (Km)	Percobaan	Banyaknya Percobaan Yang Dilalui
7250	1	10
CPU TIME (Detik)	**10,9357**	

Yashin-270-TSP

Panjang Rute Yang Diperoleh (Km)	Percobaan	Banyaknya Percobaan Yang Dilalui
40036	1	1
40016	2	5
40007	7	2
40002	9	3
39998	12	10
CPU TIME (Detik)	**698,0577**	

Pada Tabel 3 dapat dilihat bahwa *instance* Gagarin-108-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 *Random Savings Algorithm* tidak pernah mengalami keharusan memilih satu pasangan titik dari beberapa opsi pasangan titik. Sementara *instance* lainnya mengalami perbaikan panjang rute karena beberapa kali *Random Savings Algorithm* harus memilih satu pasangan titik dari beberapa opsi pasangan titik. Hal ini menunjukkan bahwa *Random Savings Algorithm* mempunyai kemampuan untuk melakukan perbaikan terhadap rute yang terbentuk dari percobaan sebelumnya.

Untuk rute solusi dari setiap *instance* dapat dilihat pada bagian lampiran.

SIMPULAN

Random Savings Algorithm saat menyelesaikan *Travelling Salesman Problem* dapat memperbaiki rute yang sudah terbentuk dari percobaan sebelumnya dan waktu penyelesaiannya pun sangat wajar untuk ditunggu. Agar dapat diketahui lebih lanjut mengenai kemampuan *Random Savings Algorithm* dalam menyelesaikan *TSP*, maka perlu dilakukan pengujian lebih lanjut pada berbagai *instance* yang lain.

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

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

DAFTAR PUSTAKA

Applegate, D. L., Bixby, R. E., Chvátal, V., & Cook, W. J. (2011). The traveling salesman problem: A computational study. In *The Traveling Salesman Problem: A Computational Study*. https://doi.org/10.5860/choice.45-0928

Elshaer, R., & Awad, H. (2020). A taxonomic review of metaheuristic algorithms for solving the vehicle routing problem and its variants. *Computers & Industrial Engineering*, *140*, 106242.

GeoNames. (n.d.). https://download.geonames.org/export/

Golden, B., Wang, X., & Wasil, E. (2023). The Evolution of the Vehicle Routing Problem—A Survey of VRP Research and Practice from 2005 to 2022. In *The Evolution of the Vehicle Routing Problem: A Survey of VRP Research and Practice from 2005 to 2022* (pp. 1–64). Springer.

Idrizi, B. (2020). Necessity for geometric corrections of distances in web and mobile maps. *International Conference on Cartography and GIS, Bulgaria*.

Jiménez Aguirre, M. F. (2022). *Algoritmo de Clarke and Wright para mejorar la gestión de ruta del transporte de acopio de leche en Cañete, 2020*.

Kunnapapdeelert, S., & Thawnern, C. (2021). Capacitated vehicle routing problem for Thailand's steel industry via saving algorithms. *Journal of System and Management Sciences*, *11*(2), 171–181.

Lysgaard, J. (1997). Clarke & Wright's savings algorithm. *Department of Management Science and Logistics, The Aarhus School of Business*, *44*, 1–7.

Nagar, S., & Nagar, S. (2018). *Introduction to Octave*. Springer.

Ramos, V. S. (2020). SIHR: a MATLAB/GNU Octave toolbox for single image highlight removal. *Journal of Open Source Software*, *5*(45), 1822.

Sanggala, E., & Bisma, M. A. (2023). *Random Nearest Neighbour Untuk Menyelesaikan Russian TSP Instances*. *15*(1), 63–69.

Shahab, M. L. (2019). New heuristic algorithm for traveling salesman problem. *Journal of Physics:*

Conference Series, *1218*(1), 12038.

Wu, Z. (2020). A comparative study of solving traveling salesman problem with genetic algorithm, ant colony algorithm, and particle swarm optimization. *Proceedings of the 2020 2nd International Conference on Robotics Systems and Vehicle Technology*, 95–99.

Zhang, J. (2021). Comparison of various algorithms based on TSP solving. *Journal of Physics: Conference Series*, *2083*(3), 32007.

LAMPIRAN
Rute Solusi Yang Dihasilkan Dari *Random Savings Algorithm*

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