A Heuristic Algorithm and Its Iterative Approach for the Symmetric And Asymmetric Traveling Salesman Problems

[1]Muhammad Luthfi Shahab and [1]Mohammad Isa Irawan

[1]Department of Mathematics, Institut Teknologi Sepuluh Nopember, Indonesia

luthfishahab@matematika.its.ac.id; *mii@its.ac.id*

ABSTRACT

In this research, we explain a heuristic algorithm (S1) for solving the traveling salesman problem (TSP). It has polynomial-time complexity and equal complexity when it is used to solve the symmetric and asymmetric TSPs. We also explain its iterative approach (S2). The S2 uses the S1 iteratively with some different parameters or values of K and I. To know the performance of the S2, we use it to solve the symmetric and asymmetric TSPs obtained from TSPLIB. The S2 can find a very good solution in a short time. For 80 symmetric and 19 asymmetric problems, the average of the accuracy of the S2 are 99.83% and 99.92%, respectively. All our data, algorithm, and results obtained in this research can be freely accessed online through https://github.com/mlshahab/haiasatsp.

Keywords: traveling salesman problem, heuristic, algorithm, iterative, polynomial-time

INTRODUCTION

Suppose that there is a traveler who want to go to n different cities, each city is visited once, and after he visits all cities, he wants to go back to the first visited city. Finding the route which has the smallest total distance is usually called by the traveling salesman problem (TSP). The TSP is usually defined mathematically as a graph, where the nodes represent the cities and the edges represent the paths from a city to another cities. The TSP is a NP-hard problem. The symmetric and asymmetric TSPs with n nodes have $(n-1)!/2$ and $(n-1)!$, respectively, different solutions.

The TSP has many different types, according to the condition, addition or restriction on it, such as the symmetric TSP, the asymmetric TSP, the geometric TSP, the Euclidean TSP, the constant TSP, the generalized graphical TSP, the Maximum TSP, the generalized TSP, the prize collecting TSP, the bottleneck TSP, and many others (Gutin & Punnen, 2006). The most known generalization of the TSP are the traveling purchaser problem and the vehicle routing problem (Shahab, Utomo & Irawan, 2016).

The TSP has many applications in logistics (Maggioni, Perboli & Tadei, 2014), genome sequencing, bioinformatics (Fischer, Fischer, Jäger, Keilwagen, Molitor & Grosse, 2014), scan chains, drilling problems, aiming telescopes and x-rays, data clustering (Applegate, Bixby, Chvatal & Cook, 2006), chemical shipping (Elgesem, Skogen, Wang & Fagerholt, 2018), air logistics (Wu, Zhou, Du & Lv, 2019), and many others.

A dynamic programming approach for the TSP (Held & Karp, 1962) has been developed on 1962. Heuristic algorithms are usually used to solve the TSP. Yet heuristic algorithms rarely find the optimal solution, they can find relatively good solutions in a short time. Usually, heuristic algorithms are very simple, and it makes the algorithm can solve the problem faster than another meta-heuristic or complex algorithms. Many heuristic algorithm are based on Lin-Kernighan heuristic algorithm (Lin & Kernighan, 1973). The idea of their heuristic algorithm is removing and reconnecting the edges on better position.

In this research, we explain a heuristic algorithm (S1) and its iterative approach. The algorithm was firstly introduced by Shahab (Shahab, 2019). We will give a better explanation about it. We also give an iterative approach and combine it with the nearest neighbor algorithm, and 2-opt algorithm. We will use the proposed iterative heuristic algorithm to solve the symmetric and asymmetric TSPs from TSPLIB (Reinelt, 1991; Reinelt, 1995).

Section 2 describes the TSP. Section 3 describes the S1 and its iterative approach. Section 4 describes the results and discussions and section 5 is the conclusions.

THE TRAVELING SALESMAN PROBLEM

Suppose that there is a TSP with n nodes. The nodes are labeled by $1,2,\dots,n$. Every TSP has a distance matrix

$$D = \begin{pmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & \cdots & d_{nn} \end{pmatrix} \quad (1)$$

where d_{ij} represents the distance from node i to node j. The value of d_{ii} is 0 or a very big number. If the TSP is symmetric, then $d_{ij} = d_{ji}$ for all $i, j \in \{1,2,\dots,n\}$.

A solution of a TSP is a permutation of $\{1,2,\dots,n\}$. Suppose that we have a solution $\sigma = p_1 p_2 \dots p_n$. The distance of the solution, $f(\sigma)$, is usually calculated by

$$f(\sigma) = \sum_{i=0}^{n} d_{p_i p_{i+1}} \quad (2)$$

where $p_{n+1} = p_0$ and $d_{p_i p_{i+1}}$ is the element of D at p_i-th row and p_{i+1}-th column. The objective of the TSP is to find a solution σ that has smallest $f(\sigma)$. Note that $p_1 p_2 \dots p_n$ is equal with $p_i p_{i+1} \dots p_n p_1 p_2 \dots p_{i-1}$ for all $i \in \{1,2,\dots,n\}$.

TSPLIB

TSPLIB (traveling salesman problem library) is a library of sample instances for the TSP (symmetric and asymmetric TSPs) and related problems (hamiltonian cycle problem, sequential ordering problem, and capacitated vehicle routing problem) from various sources and types (Reinelt, 1991; Reinelt, 1995).

Each problem has its own edge weight type. The most used edge weight type is EUC_2D and the distance from node i to node j is calculated by

$$d_{ij} = \left\lfloor \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} + 0.5 \right\rfloor \quad (3)$$

where $\lfloor x \rfloor$ is the floor function of x. The ways to calculate the other types, can be read on (Reinelt, 1991; Reinelt, 1995).

THE HEURISTIC ALGORITHM

This heuristic algorithm (S1) was firstly introduced by Shahab (Shahab, 2019). We give a better explanation and example about it. The idea of the S1 is iteratively removing a segment and then inserting every node in it to the best position in the solution.

Suppose that $\sigma = p_1 p_2 \dots p_n$ is a solution of the TSP with n nodes. The solution is divided into two segments, the first segment is $\upsilon = p_i p_{i+1} \dots p_{i+K-1}$ of length K and the other segment is $\tau = a_1 a_2 \dots a_{n-K} = p_1 p_2 \dots p_{i-1} p_{i+K} \dots p_n$ of length $n - K$ obtained by removing segment υ from σ. We can calculate $f(\tau)$ using (2), but for faster calculation, we use

$$f(\tau) = f(\sigma) + d_{p_{i-1}p_{i+K}} - \sum_{k=i}^{i+K} d_{p_{k-1}p_k} \quad (4)$$

The order of elements in υ is shuffled to get new $\upsilon = b_1 b_2 \dots b_K$. The first node to be added is b_1. If b_1 is placed between two nodes in τ without changing the order of nodes in τ, then there are $n - K$ different possible positions, those are

- $\tau_1 = a_1 b_1 a_2 \dots a_{n-K}$,
- $\tau_2 = a_1 a_2 b_1 \dots a_{n-K}$,
- ⋮
- $\tau_k = a_1 a_2 \dots a_k b_1 a_{k+1} \dots a_{n-K}$,
- ⋮
- $\tau_{n-K-1} = a_1 a_2 \dots b_1 a_{n-K}$, and
- $\tau_{n-K} = a_1 a_2 \dots a_{n-K} b_1$.

Suppose that the distance of σ is $f(\sigma)$, then the distance of τ_k is

$$f(\tau_k) = f(\sigma) + d_{a_k b_1} + d_{b_1 a_{k+1}} - d_{a_k a_{k+1}} \quad (5)$$

The algorithm chooses the option with the smallest distance. Say that the best one is τ_1, then τ is set to be τ_1 and υ is set to be $b_2 \dots b_K$. Next, we add b_2 into the new τ using the same procedure. The process is repeated until υ is empty and τ contains all elements of $\{1,2, \dots, n\}$.

After that, the distance of τ and σ is compared. If $f(\tau) < f(\sigma)$, i.e. τ is better than σ, then we set $\sigma = \tau$. If not, then σ is remain unchanged. Again, the whole process is repeated for $i = 1,2, \dots, n$. One iteration of this heuristic algorithm is ended after the process for $i = n$ is completed. We can repeat the iteration I times.

With (4) and (5), we do not need to compute the distance of new τ from the beginning. Those equations will make the S1 run faster because we do not need to calculate the same values repeatedly. As we can see, the S1 does not has a part that depends on the symmetric or asymmetric conditions, so we can conclude that the S1 has an equal time and space complexities when it is used to solve the symmetric and asymmetric TSPs.

We can use $K = \sqrt{n}$ for small problems and choose the value between $n/20$ until $n/10$ for bigger problems. We can use $I = 100$ as its standard value. The value of I also can depends on n, like $n/10$. To generate an initial solution, first we pick a random node, and then use the nearest neighbor algorithm to create a complete solution.

For an example, suppose that we have a TSP with 16 nodes. You can see the coordinates (red dots) on Figure 1. Suppose that we have solution

1 9 13 2 6 16 11 7 5 4 12 14 10 8 3 15.

If we remove the first 5 nodes, then we get

16 11 7 5 4 12 14 10 8 3 15.

If we add

1 9 13 2 6

into the right places, then we get

16 11 7 6 5 4 12 13 14 10 9 8 3 2 1 15.

If we remove

11 7 6 5 4,

then we get

16 12 13 14 10 9 8 3 2 1 15.

If we add them again into the right places, then we get

16 12 13 14 11 10 9 8 7 6 5 4 3 2 1 15.

If we remove

15 16 12 13 14,

then we get

11 10 9 8 7 6 5 4 3 2 1.

If we add them again into the right places, then we get

16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1,

or we can write it as

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16,

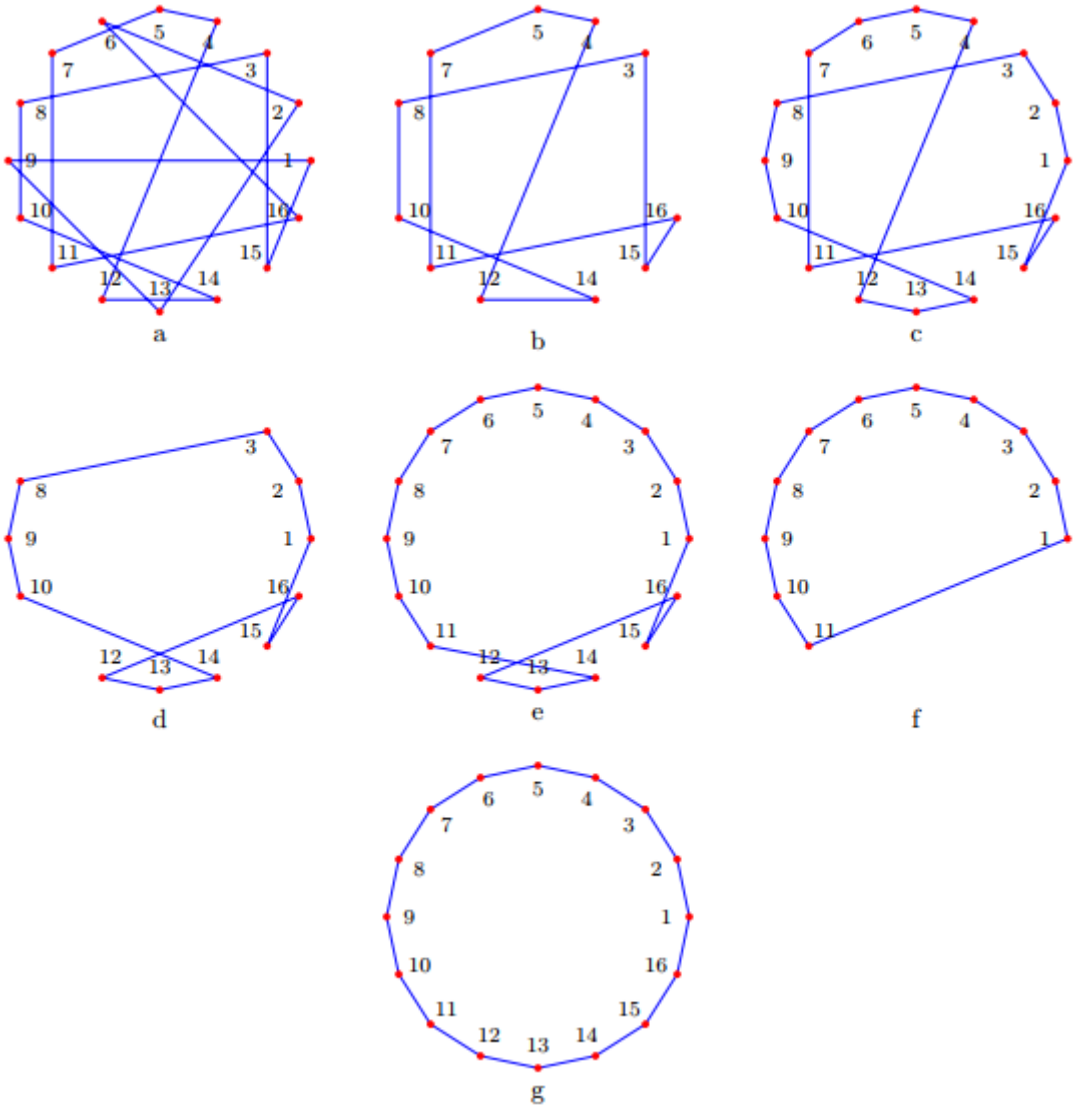

Figure 1. The process examples of the S1 (a) The initial solution (b) Removing 5 nodes (c) The 5 nodes are added again (d) Removing 5 nodes (e) The 5 nodes are added again (f) Removing 5 nodes (g) The 5 nodes are added again, it is the optimal solution

this is the optimal solution. In this example, the order of the 5 nodes to be added does not important since it is an easy problem. Whatever the order, we will get the same result. For hard problem, the order is so important. Because of that, the S1 always shuffles the order of the nodes in v to get larger search space.

The Complexity

The main part of the S1 is choosing the best position for b_i. Note that b_1 has $n - K$ options and we must check all of them one by one, b_2 has $n - K + 1$ options, …, and b_K has $n - K + K - 1$ options. So the S1 needs to checks

$$
\begin{aligned}
&(n-K)+(n-K+1)+\cdots+(n-K+K-1)\\
&\quad = Kn-K^2+(1+2+\cdots+K-1)\\
&\quad = Kn-K^2+\frac{1}{2}K(K-1)\\
&\quad = Kn-K^2+\frac{1}{2}K^2-\frac{1}{2}K\\
&\quad = Kn-\frac{1}{2}K^2-\frac{1}{2}K
\end{aligned}
$$

process. Because the S1 repeats the whole process I times, then we get

$$I\left(Kn-\frac{1}{2}K^2-\frac{1}{2}K\right) \tag{6}$$

Since Kn is the biggest component, and if we use $K=c_1\sqrt{n}$ and $I=c_2$, where c_1 and c_2 are constants, then the complexity of the S1 is $O\left(n^{3/2}\right)$. If the value of I depends on n, i.e. $I=n/10$, then the complexity is $O\left(n^{5/2}\right)$. We can conclude that the S1 is a polynomial-time algorithm.

2-opt Algorithm

The bad news about the S1 is it can not fixes a big crossed section with usual value of K. For example, the S1 with $K=5$ can not fixes the crossed section in Figure 2, although we repeat the process forever. To fix it, the S1 must use a big value of K, for example $K=8=n/2$. Of course this value will makes the S1 run much slower. To overcome that weakness, we use the 2-opt algorithm. This algorithm is a powerful algorithm for fixing crossed sections and very fast for the symmetric TSP, since the simplification below is true only for the symmetric case.

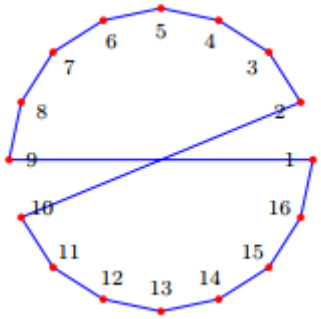

Figure 2. A big crossed section

Simplification of 2-opt Algorithm

Suppose that we have a solution $\sigma=p_1\dots p_{i-1}p_ip_{i+1}\dots p_{j-1}p_jp_{j+1}\dots p_n$ and the distance of the solution is $f(\sigma)$. If $\tau=p_1\dots p_{i-1}p_jp_{j-1}\dots p_{i+1}p_ip_{j+1}\dots p_n$ is obtained by reversing the segment $p_ip_{i+1}\dots p_{j-1}p_j$, then the distance is

$$
\begin{aligned}
f(\tau)=f(\sigma)&-d_{p_{i-1}p_i}-d_{p_ip_{i+1}}-\cdots-d_{p_{j-1}p_j}-d_{p_jp_{j+1}}\\
&+d_{p_{i-1}p_j}+d_{p_jp_{j-1}}+\cdots+d_{p_{i+1}p_i}+d_{p_ip_{j+1}}
\end{aligned}
$$

In the symmetric case, $d_{ij} = d_{ji}$ for all $i, j \in \{1,2, \dots, n\}$ and we get

$$d_{p_i p_{i+1}} + \cdots + d_{p_{j-1} p_j} = d_{p_j p_{j-1}} + \cdots + d_{p_{i+1} p_i}$$

and the value of $f(\tau)$ is

$$f(\tau) = f(\sigma) - d_{p_{i-1} p_i} - d_{p_j p_{j+1}} + d_{p_{i-1} p_j} + d_{p_1 p_{j+1}} \quad (7)$$

With (7), we can remove many unnecessary calculations, and it makes the 2-opt algorithm run faster. However, for the asymmetric TSP, since the distance from node i to node j is different with the distance from node j to node i, then we can not use this simplification.

The Iterative Approach

As we can see, the S1 depends on K and I. If K is too small, then it will not efficiently improve the solution. If it is too big, then it will make the algorithm run slow, because of the changing of big segment in the solution. If I is too small, then the algorithm does not have enough trial to produce better solution. If it is too big, then it will make the algorithm to repeatedly do unnecessary process.

The idea of the iterative approach (S2) of the S1, is to iteratively use S1 with some different good parameters or values of K and I. The values are obtained by many trial and error, and then calculating the result improvements based on the values.

We choose three values of I, those are $I_1 = 100$, $I_2 = 10$, $I_3 = 20$, and six values of K. The first value is $K_1 = \lfloor \sqrt{n} \rfloor + 1$. Its imitates the original value of K used by Shahab (Shahab, 2019). The other values are $K_2 = \left\lfloor \frac{3}{2}\sqrt{n} \right\rfloor$, $K_3 = \lfloor 2\sqrt{n} \rfloor$, $K_4 = \left\lfloor \frac{5}{2}\sqrt{n} \right\rfloor$, $K_5 = \lfloor 3\sqrt{n} \rfloor$, and $K_6 = \left\lfloor \frac{7}{2}\sqrt{n} \right\rfloor$.

Since the smallest TSP in TSPLIB is burma14 that has 14 nodes and

$$K_6 = \left\lfloor \frac{7}{2}\sqrt{14} \right\rfloor = 13$$

then we can use all values of $K_1, K_2, \dots, K_6$ for all problems in TSPLIB. The biggest problem used in this research is u1060 that has 1060 nodes, and we get $K_1 = 33$, $K_2 = 48$, $K_3 = 65$, $K_4 = 81$, $K_5 = 97$, and $K_6 = 113$. The value of K_6 is close to $1060/10$.

For the symmetric TSP, we use seven pairs of value for the parameters of S1, those are (K_1, I_1), (K_1, I_1), (K_2, I_2), (K_3, I_2), (K_4, I_2), (K_5, I_2), (K_6, I_2), and (K_7, I_2). We replace I_2 with I_3 for the asymmetric case. In general, we use the S1 8 times with different parameters (except for the first two pairs).

For the symmetric case, we add the 2-opt algorithm after the S1 has completed its process for one parameter. The 2-opt algorithm is ran until there is no improvement anymore. After that we use again the S1 with the next parameter.

The overall processes of the S2 for the symmetric TSP are building initial solution with the nearest neighbor algorithm, S1(K_1,I_1), 2-opt, S1(K_1,I_1), 2-opt, S1(K_2,I_2), 2-opt, S1(K_3,I_2), 2-opt, S1(K_4,I_2), 2-opt, S1(K_5,I_2), 2-opt, and S1(K_6,I_2). For the asymmetric TSP, we do not use the 2-opt algorithm because

the simplification does not work. Because of that, for the asymmetric case, we use $I_3 = 20$ instead of $I_2 = 10$.

RESULTS

To test the performance of the S2, we use 80 symmetric and 19 asymmetric problems obtained from TSPLIB. We have changed all those problems into FULL MATRIX type. You can use or access them through https://github.com/mlshahab/tsplib. Transforming into FULL MATRIX type will reduces useless calculations for same i and j, but need more memory space to save $n \times n$ matrix instead of $n \times 2$ coordinate array.

For each problem, we have tested the S2 30 times and then we save the results. We use a personal computer with Windows 10 Pro 64-bit, Intel(R) Core(TM) i5-2400S CPU @ 2.50GHz, and 4 GB RAM.

The results of the S2 for the symmetric TSP are shown in Table 1 and for the asymmetric TSP are shown in Table 2. The first column is the problem name. The second is the dimension or the number of nodes from the problem. The third is the optimal distance of the problem, it is obtained from http://elib.zib.de/pub/mptestdata/tsp/tsplib/stsp-sol.html. The fourth is the distance obtained by the S2. The fifth is the accuracy (in %) of obtained distance compared to the optimal distance. The accuracy is calculated by

$$\left(1 - \frac{d - d^*}{d^*}\right) 100 \tag{8}$$

where d is the obtained distance and d^* is the optimal distance for the problem. The last column is the computational time (in second) needed for the S2 to solve the problem.

The S2 finds optimal solutions for 50 different symmetric problems. The lowest accuracy is 98.94% and the average of the accuracy is 99.83%. It is better than 99.50% obtained by the S1 (Shahab, 2019). For the asymmetric case, the S2 finds optimal solutions for 15 different problems. The lowest accuracy is 99.02% and the average of the accuracy is 99.92%. In term of computational time, we can see that the time grow regularly, depends on the dimension of the problem.

Figure 3 shows the comparison of the accuracy obtained by the S1 (the red line) and the S2 (the blue line) for 80 symmetric problems. The accuracy of the S1 are obtained from (Shahab, 2019). It can be seen that the S2 is better than the S1 for almost all of the problems. The S1 is better than the S2 only in two problems. It is probably because the S1 was ran for 100 times instead of 30 times (used in this research). For larger problems, the accuracy of the S1 and the S2 decrease. It is because the algorithms do not guarantee the optimality of the solution.

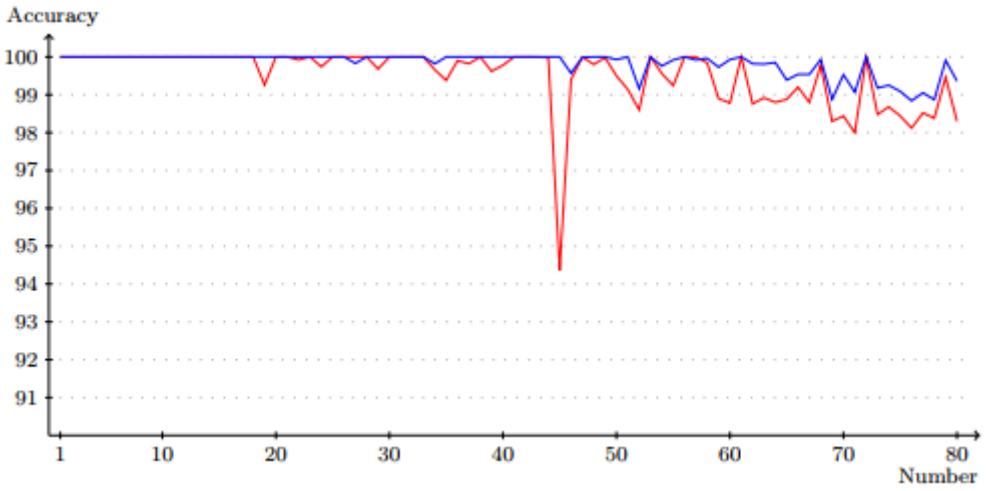

Figure 3. The comparison of the accuracy obtained by the S1 and the S2 for 80 symmetric problems

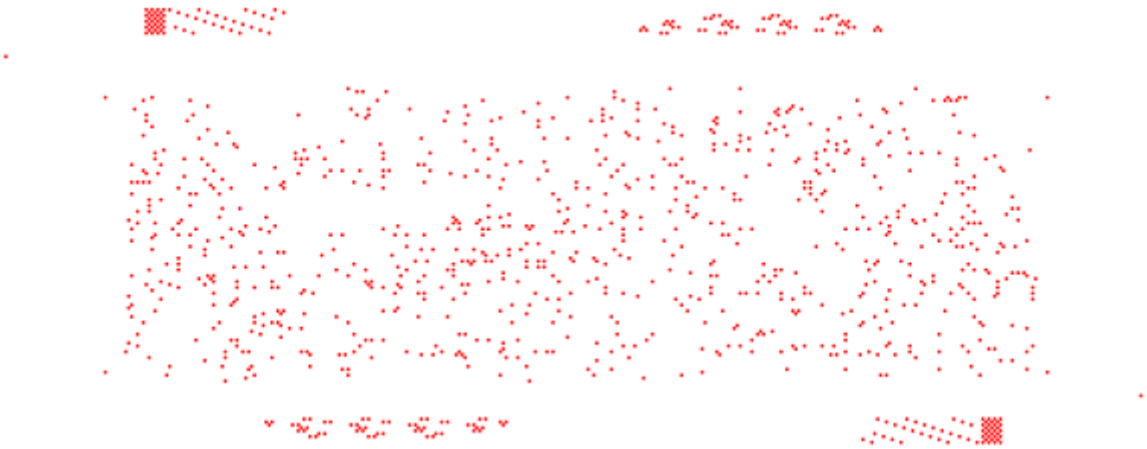

Figure 4. The node coordinates of u1060 problem

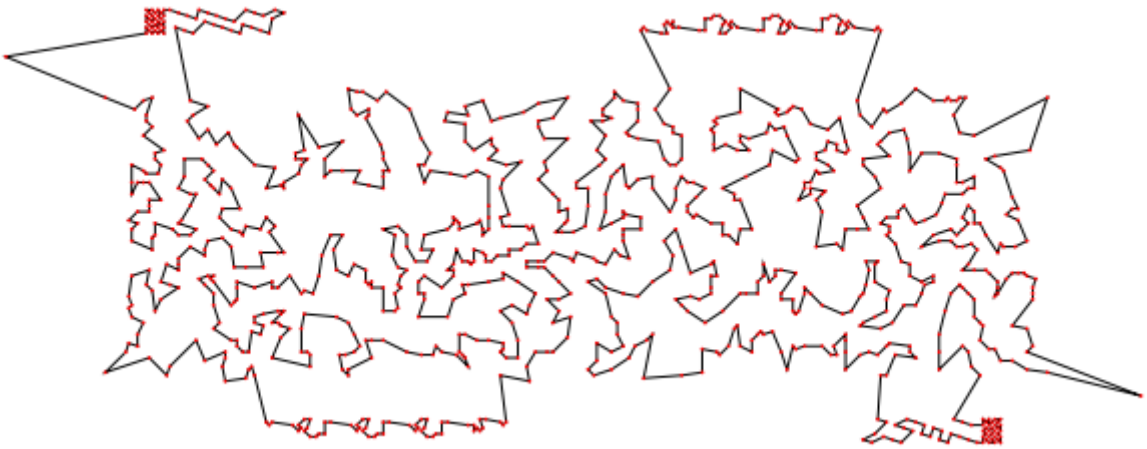

Figure 5. The solution of u1060 problem obtained by the S2

Figure 4 shows the node coordinates of u1060 problem. It has 1060 nodes. Figure 5 shows a solution example obtained by the S2. Obviously, it does not have a crossed edge because we also use 2-opt algorithm and is better than the one obtained by the S1. Its distance is 225535 and its accuracy is 99.36%.

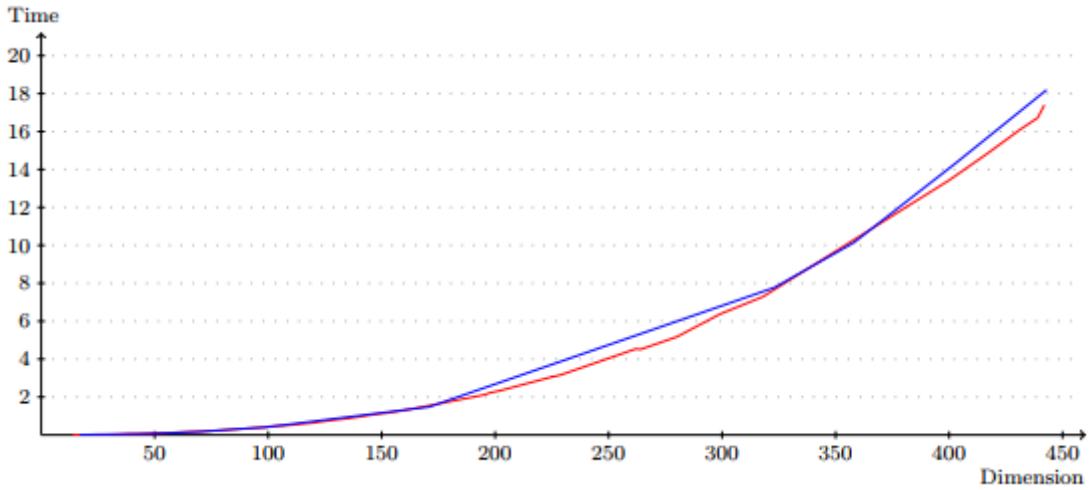

Figure 6. The comparison of the computational time for the S2 to solve the symmetric and asymmetric TSPs

Figure 6 shows the comparison of computational time for the S2 to solve the symmetric (the red line) and asymmetric TSPs (the blue line). We use computational times of 64 symmetric problems and all asymmetric problems. It can be seen that the difference is small enough. It is because we use different settings for the symmetric and asymmetric cases.

CONCLUSIONS

In this research, we explain a heuristic algorithm (S1) and its iterative approach (S2) for the symmetric and asymmetric TSPs. The S1 was firstly introduced by Shahab (Shahab, 2019).

We explain the S1 more clearly with easy symbol and clear function. We also give the example on easy small problem. We use the nearest neighbor algorithm to produce an initial solution for it. It is fast, has polynomial-time complexity and very easy to implement. Unlike usual another algorithm, the S1 has an equal time and space complexities when it is used to solve the symmetric and asymmetric TSPs.

We also give an iterative approach, the S2, of it. The S2 use the S1 iteratively with some different parameters or values of K and I. We use six values of K and three values of I. For the symmetric TSP, we add 2-opt algorithm between two use of S1. Either the S1 or the S2 can be used to solve the symmetric and asymmetric TSPs. The algorithm is fast, yet it can finds very good solution.

We have shared our data, algorithm, and results (30 solutions for each problem and corresponding distances). Other researchers can freely look at or access them through https://github.com/mlshahab/haiasatsp.

ACKNOWLEDGEMENT

This research was supported by Institution of Research and Community Service (LPPM), Institut Teknologi Sepuluh Nopember (ITS), Ministry of Research, Technology, and Higher Education (KEMENRISTEKDIKTI). In accordance with the funding agreement of Penelitian Pemula, contract number: 1193/PKS/ITS/2019.

REFERENCES

Applegate, D. L., Bixby, R. E., Chvatal, V., & Cook, W. J. (2006). *The traveling salesman problem: a computational study*. Princeton university press.

Elgesem, A. S., Skogen, E. S., Wang, X., & Fagerholt, K. (2018). A traveling salesman problem with pickups and deliveries and stochastic travel times: An application from chemical shipping. *European Journal of Operational Research*, *269*(3), 844-859.

Fischer, A., Fischer, F., Jäger, G., Keilwagen, J., Molitor, P., & Grosse, I. (2014). Exact algorithms and heuristics for the quadratic traveling salesman problem with an application in bioinformatics. *Discrete Applied Mathematics*, *166*, 97-114.

Gutin, G., & Punnen, A. P. (Eds.). (2006). *The traveling salesman problem and its variations* (Vol. 12). Springer Science & Business Media.

Held, M., & Karp, R. M. (1962). A dynamic programming approach to sequencing problems. *Journal of the Society for Industrial and Applied mathematics*, *10*(1), 196-210.

Lin, S., & Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling-salesman problem. *Operations research*, *21*(2), 498-516.

Maggioni, F., Perboli, G., & Tadei, R. (2014). The multi-path traveling salesman problem with stochastic travel costs: Building realistic instances for city logistics applications. *Transportation Research Procedia*, *3*, 528-536.

Reinelt, G. (1991). TSPLIB—A traveling salesman problem library. *ORSA journal on computing*, *3*(4), 376-384.

Reinelt, G. (1995). Tsplib95. *Interdisziplinäres Zentrum für Wissenschaftliches Rechnen (IWR), Heidelberg*, *338*.

Shahab, M. L. (2019, May). New heuristic algorithm for traveling salesman problem. In *Journal of Physics: Conference Series* (Vol. 1218, No. 1, p. 012038). IOP Publishing.

Shahab, M. L., Utomo, D. B., & Irawan, M. I. (2016). Decomposing and solving capacitated vehicle routing problem (CVRP) using two-step genetic algorithm (TSGA). *Journal of Theoretical & Applied Information Technology*, *87*(3).

Wu, J., Zhou, L., Du, Z., & Lv, Y. (2019). Mixed steepest descent algorithm for the traveling salesman problem and application in air logistics. *Transportation Research Part E: Logistics and Transportation Review*, *126*, 87-102.

Table 1. The results of the S2 for 80 symmetric problems

Number	Problem	Dimension	Optimal Distance	S2 Distance	Accuracy	Time
1	burma14	14	3323	3323	100	0.01
2	ulysses16	16	6859	6859	100	0.01
3	gr17	17	2085	2085	100	0.01
4	gr21	21	2707	2707	100	0.01
5	ulysses22	22	7013	7013	100	0.01
6	gr24	24	1272	1272	100	0.02
7	fri26	26	937	937	100	0.02
8	bayg29	29	1610	1610	100	0.02
9	bays29	29	2020	2020	100	0.02
10	dantzig42	42	699	699	100	0.06
11	swiss42	42	1273	1273	100	0.06
12	att48	48	10628	10628	100	0.07
13	gr48	48	5046	5046	100	0.07
14	hk48	48	11461	11461	100	0.07
15	eil51	51	426	426	100	0.09
16	berlin52	52	7542	7542	100	0.09
17	brazil58	58	25395	25395	100	0.12
18	st70	70	675	675	100	0.18
19	eil76	76	538	538	100	0.22
20	pr76	76	108159	108159	100	0.22
21	gr96	96	55209	55209	100	0.38
22	rat99	99	1211	1211	100	0.4
23	kroA100	100	21282	21282	100	0.43
24	kroB100	100	22141	22141	100	0.44
25	kroC100	100	20749	20749	100	0.43
26	kroD100	100	21294	21294	100	0.44
27	kroE100	100	22068	22106	99.83	0.44
28	rd100	100	7910	7910	100	0.44
29	eil101	101	629	629	100	0.45
30	lin105	105	14379	14379	100	0.48
31	pr107	107	44303	44303	100	0.49
32	gr120	120	6942	6942	100	0.64
33	pr124	124	59030	59030	100	0.71
34	bier127	127	118282	118490	99.82	0.76
35	ch130	130	6110	6110	100	0.8
36	pr136	136	96772	96772	100	0.88
37	gr137	137	69853	69853	100	0.9
38	pr144	144	58537	58537	100	1.03
39	ch150	150	6528	6528	100	1.13
40	kroA150	150	26524	26524	100	1.13

Continuation ofTable 1

Number	Problem	Dimension	Optimal Distance	S2 Distance	Accuracy	Time
41	kroB150	150	26130	26130	100	1.13
42	pr152	152	73682	73682	100	1.15
43	u159	159	42080	42080	100	1.28
44	si175	175	21407	21407	100	1.64
45	brg180	180	1950	1950	100	1.76
46	rat195	195	2323	2333	99.57	2.11
47	d198	198	15780	15781	99.99	2.23
48	kroA200	200	29368	29368	100	2.28
49	kroB200	200	29437	29437	100	2.28
50	gr202	202	40160	40187	99.93	2.33
51	ts225	225	126643	126643	100	3.06
52	tsp225	225	3919	3952	99.16	3.07
53	pr226	226	80369	80369	100	3.08
54	gr229	229	134602	134924	99.76	3.17
55	gil262	262	2378	2380	99.92	4.54
56	pr264	264	49135	49135	100	4.51
57	a280	280	2579	2581	99.92	5.17
58	pr299	299	48191	48209	99.96	6.37
59	lin318	318	42029	42141	99.73	7.29
60	rd400	400	15281	15291	99.93	13.43
61	fl417	417	11861	11861	100	14.85
62	gr431	431	171414	171724	99.82	16.08
63	pr439	439	107217	107426	99.81	16.72
64	pcb442	442	50778	50853	99.85	17.4
65	d493	493	35002	35217	99.39	22.06
66	att532	532	27686	27814	99.54	27.95
67	ali535	535	202310	203235	99.54	33.17
68	si535	535	48450	48482	99.93	32.57
69	pa561	561	2763	2794	98.88	37.44
70	u574	574	36905	37079	99.53	39.03
71	rat575	575	6773	6836	99.07	37.4
72	p654	654	34643	34643	100	54.46
73	d657	657	48912	49312	99.18	54.7
74	gr666	666	294358	296569	99.25	57.64
75	u724	724	41910	42293	99.09	70.12
76	rat783	783	8806	8908	98.84	101.42
77	dsj1000	1000	18659688	18836222	99.05	237.25
78	pr1002	1002	259045	261978	98.87	198.4
79	si1032	1032	92650	92732	99.91	231.45
80	u1060	1060	224094	225535	99.36	242.05

Table 2. The results of the S2 for 19 asymmetric problems

Number	Problem	Dimension	Optimal Distance	S2 Distance	Accuracy	Time
1	br17	17	39	39	100	0.01
2	ftv33	34	1286	1286	100	0.03
3	ftv35	36	1473	1473	100	0.04
4	ftv38	39	1530	1530	100	0.05
5	p43	43	5620	5620	100	0.05
6	ftv44	45	1613	1613	100	0.06
7	ftv47	48	1776	1776	100	0.07
8	ry48p	48	14422	14422	100	0.07
9	ft53	53	6905	6905	100	0.09
10	ftv55	56	1608	1608	100	0.1
11	ftv64	65	1839	1839	100	0.15
12	ft70	70	38673	38797	99.68	0.17
13	ftv70	71	1950	1950	100	0.18
14	kro124p	100	36230	36241	99.97	0.42
15	ftv170	171	2755	2755	100	1.48
16	rbg323	323	1326	1339	99.02	7.77
17	rbg358	358	1163	1165	99.83	10.14
18	rbg403	403	2465	2465	100	14.34
19	rbg443	443	2720	2720	100	18.2