JTAM (Jurnal Teori dan Aplikasi Matematika)
http://journal.ummat.ac.id/index.php/jtam
p-ISSN 2597-7512 | e-ISSN 2614-1175
Vol. 6, No. 3, July 2022, pp. 601-615

Simulated Annealing Algorithm for Determining Travelling Salesman Problem Solution and Its Comparison with Branch and Bound Method

Bib Paruhum Silalahi[1], Farahdila Sahara[2], Farida Hanum[3], Hidayatul Mayyani[4]
[1,2,3,4] Department of Mathematics, IPB University, Bogor, Indonesia
bibparuhum@gmail.com[1], farahdila_shr@apps.ipb.ac.id[2], fhanum@apps.ipb.ac.id[3], mayyani_mat15@apps.ipb.ac.id[4]

Article History:
Received : 21-04-2022
Revised : 17-05-2022
Accepted: 20-05-2022
Online : 16-07-2022

Keywords:
Branch and Bound;
Integer Linear Programming;
Simulated Annealing;
Travelling Salesman Problem;

ABSTRACT

Travelling Salesman Problem (TSP) is a problem where a person must visit some places, starting from one city and then moving on to the next city with the conditions that the places visited can only be passed precisely once and then back to the starting city. TSP is an NP-hard, an important problem in operations research. TSP problems can be solved by an exact method or an approximation method, namely the metaheuristic method. This research aims to solve the TSP problem with an approximation method called the Simulated Annealing (SA), and then compare the results of this approximation method with the exact Branch and Bound method. The results indicated that the SA method could accomplish TSP problems. However, like other metaheuristic methods, SA only accomplishes it using an approach to get good results. Still, it cannot be determined that SA has the most optimal results, but the time needed by the SA method is faster than the Branch and Bound method. In case I, the percentage difference between the distance generated using the SA method with the B-and-B method is 0%, in case II it is 7% and in case III it is 8%.

https://doi.org/10.31764/jtam.v6i3.8481

This is an open access article under the CC–BY-SA license

A. INTRODUCTION

Travelling Salesman Problem (TSP) is a problem where a person must visit some places, starting from one city to others. The places visited can only be passed precisely once and then go back to the starting city. TSP is often used to solve the distribution activities of a company. The company will undoubtedly try to keep costs to a minimum to get the maximum possible profit and minimize distribution activities costs. (Lalang et al., 2018) modeled a distribution problem of vehicle routing with time windows and occasional drivers. Then this problem is expanded by the existence of multi-depots (Making et al., 2018). Minimizing the cost of distribution activities can be done by finding the shortest route because distance positively affects distribution costs. The closer the distance traveled, the less the cost of the distribution activity.

TSP problems can be solved by an exact method or approach method, namely the heuristic or metaheuristic method. In solving TSP with the exact method, TSP is modeled as

Integer Linear Programming (ILP). ILP is classified into four based on the number of decision variables with integer values. One of which is Mixed Integer Linear Programming (MILP), where not all decision variables are integer. Then the MILP problem can be solved by one of the exact methods. One of the exact methods applied to accomplish MILP problems is Branch and Bound (B-and-B) (Jünger et al., 1995).

There are many metaheuristics approximation technique that can be applied to accomplish TSP problems. Some of the methods for examples are: Ant Colony Optimization (ACO) algorithm (Dorigo & Gambardella, 1997; Silalahi et al., 2019; Rokbani et al., 2021); Particle Swarm Optimization (PSO) (Zhong et al., 2007; Silalahi et al., 2020; Qamar et al., 2021); genetic algorithm (Y. Y. Yu et al., 2014; Liu & Zeng, 2009); Simulated Annealing (SA) (Zhan et al., 2016; Botsali & Alaykiran, 2020). Next, there are also some combination/hybrid methods: fuzzy particle PSO combined with SA (Rehab, 2011); the ACO algorithm combined with PSO (M. Yu, 2019); genetic algorithm combined with multiagent reinforcement learning (Alipour et al., 2018), water flow-like algorithm with simulated annealing (Othman et al., 2017); the ACO algorithm combined with SA (Stodola et al., 2020).

Simulated annealing (SA) (Zhan et al., 2016; Botsali & Alaykiran, 2020) is one of the oldest metaheuristic methods and is one of the first algorithms to have the ability to keep away from minimum local traps. SA uses the analogy of cooling and freezing metal into a crystal structure with minimal energy. In principle, the liquid molecules have enormous energy at big temperatures, so it is easier for these molecules to relocate toward other molecules. Then a decrease in temperature will slowly form a stable state with a minimum energy level (Zhan et al., 2016).

The mechanisms in SA avoid seeking for solutions to rapidly centre around local minimums. During the search, SA is unique in that it accepts not only a better solution but also a worse solution but decreases its probability. The likelihood of a worse solution being taken is set by two parameters, namely the temperature and the distinction between the current solution objective function values and the neighboring solutions. The purpose of taking a worse solution is to prevent convergence of the search to the local minimum. At larger temperatures, the chances of receiving a worse solution were much higher. However, as the temperature becomes less, the chances of accepting a worse solution decrease (Bayram & Şahin, 2013).

The simulated annealing algorithm is still developing for some TSP problems, which are also evolving. For example: improving the SA algorithm in TSP (He et al., 2018); SA based on symbiotic organisms optimization algorithm for TSP (Ezugwu et al., 2017); enlarging list-based SA algorithm to large-scale traveling salesman problem (Wang et al., 2019); SA with a nest box for solving large-scale TSP problem (Yang et al., 2020).

This research aims to accomplish the TSP optimization problem with the simulated annealing algorithm and then compare the results of this approximation method with the Branch and Bound method. We choose to use the SA algorithm because of its advantage avoiding the trap at a local minimum. We use the B-and-B method for the exact algorithm because the B-and-B method produces an exact optimum value. Then the results of the SA algorithm and the B-and-B method are compared in terms: total distance traveled, number of

iterations, and execution time. The computation results showed that the SA method could be used to solve TSP problems with a good approach value.

B. METHODS

1. Travelling Salesman Problem (TSP)

The Travelling Salesman Problem (TSP) can be viewed as a problem in determining Hamilton's cycle on a graph, namely the cycle that passes through all vertices of the graph exactly once (Fournier, 2009). TSP is a situation of deciding the most effective route or minimizing the distance that a person will travel to visit exactly once every predetermined place starting from one place and going back to that starting place. The following is the formula for the TSP problem (Benhida & Mir, 2018).

The function that will be optimized in TSP is:

min Z = $\sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij}\, x_{ij}$, with constraints:

a. Each place is visited exactly once,

$\sum_{i=1}^{n} x_{ij} = 1, \quad \forall j \neq i$

$\sum_{j=1}^{n} x_{ij} = 1, \quad \forall i \neq j$

b. Eliminate any sub tour in the solution,

$u_i - u_j + 1 \leq (n-1)\left(1 - x_{ij}\right), \quad \forall i, j = 2, \ldots, n$

$1 \leq u_i \leq n-1, \quad \forall i = 2, \ldots, n$

$u_i \geq 0, \quad \forall i = 2, \ldots, n$

c. Binary constraints,

$x_{ij} \in \{0,1\}, \forall i = 1,2, \ldots, n;\ \forall j = 1,2, \ldots, n$

where,

n = the amount of places to be visited,

d_{ij}= distance of place *i* to place *j*,

u_i= the order of place *i* in the tour.

and with the decision variables:

$$x_{ij} = \begin{cases} 1, \text{when there is a route from place } i \text{ to place } j \\ 0, \text{else} \end{cases}$$

2. Simulated Annealing (SA) Algorithm

The working principle of this algorithm is that at high temperatures the liquid particles have a high energy level so they are relatively easy to move to other particles, then when the temperature is lowered the particles will arrange themselves to find a stable arrangement with a minimum energy level (Rere et al., 2015). The flow of the SA algorithm is shown in Figure 1.

The following is the SA algorithm step-by-step flow.

a. The initial data is given in *x* and *y* coordinate points, so it is necessary to compute the distance between places using the Euclidean formula:

$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$

where:

d_{ij} = the distance between place *i* and place *j*.

x_i = the x coordinate at place i.
x_j = the x coordinate at place j.
y_i = the y coordinate at place i.
y_j = the y coordinate at place j.

b. Determine the initial route with the random function and calculate the length of the initial route $f(x_{old})$.
c. Determine the initial control parameter T_0 and the parameter α which will be used to derive control parameters.
d. Run the iteration by swapping two places from the initial route randomly by generating two random numbers in the range (0,n) then calculating the length of the new route $f(x_{new})$.
e. Evaluating the new route. If the length of the new route is less than the length of the initial route, define the new route as the initial route.
f. If the length of the new route is greater than the length of the initial route, a random number (r) is generated at the interval (0,1) and calculate the probability p:

$$p = e^{-\frac{f(x_{new})-f(x_{old})}{T}}$$

where:
r = random numbers in the interval (0,1),
$f(x_{new})$ = length of the new tour,
$f(x_{old})$ = length of the old tour,
T = control parameter.
Then do the criteria test:
1) If r < p then route is accepted, define new route as current route.
2) If r > p then the new route is ignored.

g. Cooling schedule,

$$T_k = \alpha \cdot T_{k-1} \quad , \quad 0 < \alpha < 1, \quad k = 1,2, \ldots$$

where α is a constant to derive the control parameter.
h. The iteration stops when the termination criteria have been met, that is when the iteration maximum has been reached.
i. Return to step 4 if the termination criteria have not been met, as shown in Figure 1.

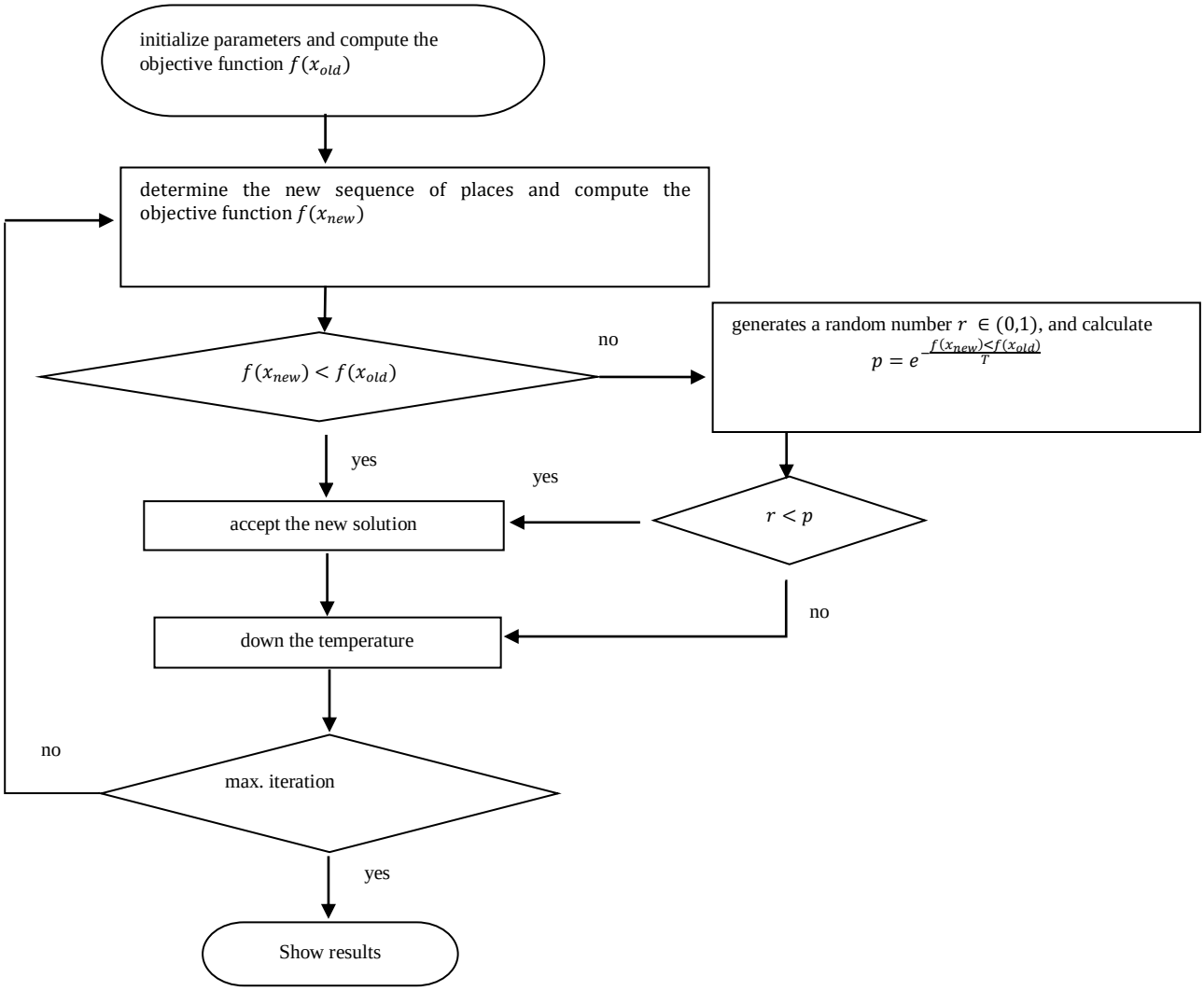

Figure 1. Flowchart of Simulated Annealing Algorithm

3. Data

A solution to TSP problems carried out using the SA method and the B-and-B method. There are three TSP issues that will be resolved. The first problem consists of 25 places, the second problem consists of 40 places, and the third problem consists of 68 places. The data used are data obtained from random functions with coordinates x and y which indicate the location of the places in Cartesian coordinates. The data are presented in Table 1.

Table 1. Data coordinate x and y for each case

Case I		**Case II**		**Case III**	
x	y	x	y	x	y
53.49	94.14	53.49	94.14	53.49	94.14
28.84	75.95	28.84	75.95	28.84	75.95
44.31	28.58	44.31	28.58	44.31	28.58
16.33	71.57	16.33	71.57	16.33	71.57
50.71	48.65	50.71	48.65	50.71	48.65
54.12	98.45	54.12	98.45	54.12	98.45
24.28	75.06	24.28	75.06	24.28	75.06
35.78	25.31	35.78	25.31	35.78	25.31

Case I		**Case II**		**Case III**	
x	y	x	y	x	y
50.05	60.28	50.05	60.28	50.05	60.28
85.48	11.12	85.48	11.12	85.48	11.12
9.08	58.11	9.08	58.11	9.08	58.11
95.46	22.88	95.46	22.88	95.46	22.88
83.47	39.44	83.47	39.44	83.47	39.44
91.37	73.04	91.37	73.04	91.37	73.04
60.42	98.67	60.42	98.67	60.42	98.67
51.56	41.53	51.56	41.53	51.56	41.53
55.73	21.64	55.73	21.64	55.73	21.64
55.61	17.59	55.61	17.59	55.61	17.59
29.98	49.08	29.98	49.08	29.98	49.08
33.67	7.20	33.67	7.20	33.67	7.20
68.61	92.34	68.61	92.34	68.61	92.34
60.96	88.75	60.96	88.75	60.96	88.75
66.74	43.31	66.74	43.31	66.74	43.31
4.14	75.21	4.14	75.21	4.14	75.21
11.23	77.58	11.23	77.58	11.23	77.58
		92.32	97.11	92.32	97.11
		45.55	67.27	45.55	67.27
		13.68	19.89	13.68	19.89
		18.47	95.28	18.47	95.28
		44.22	50.67	44.22	50.67
		66.27	70.62	66.27	70.62
		5.68	35.22	5.68	35.22
		82.61	69.82	82.61	69.82
		96.89	11.82	96.89	11.82
		88.28	53.79	88.28	53.80
		57.57	27.97	57.57	27.97
		76.75	32.95	76.75	32.95
		17.56	58.62	17.56	58.62
		97.18	90.85	97.18	90.85
		65.43	10.42	65.43	10.42
				15.76	38.61
				71.08	47.37
				3.70	51.18
				5.97	35.71
				28.09	41.06
				22.67	62.66
				10.85	50.05
				82.87	3.65
				82.48	32.65
				25.32	87.28
				65.61	57.74
				34.64	40.66
				90.73	58.82

Case I		Case II		Case III	
x	y	x	y	x	y
				88.62	2.33
				30.88	22.01
				53.21	28.01
				59.87	89.16
				61.34	53.82
				35.30	73.89
				22.13	55.43
				75.29	90.68
				75.64	24.78
				97.62	2.25
				15.82	34.29
				19.67	86.95
				71.81	31.78
				27.42	65.93
				46.51	5.91

C. RESULT AND DISCUSSION

1. Solution Using the Simulated Annealing (SA) Algorithm

In this scientific work, Spyder (Python 3.7) software is used to find TSP solutions using the SA algorithm with the following syntax.

```
start_time = time.time()
def distance(x1,y1,x2,y2):
    return math.sqrt((x1-x2)**2+(y1-y2)**2)
xls=pandas.ExcelFile('data1.xlsx')
sheet=pandas.read_excel(xls,'Sheet7')
places=sheet.as_matrix()
n=len(places)
tur=random.sample(range(n),n);
print("tur awal =",tur)
def totaldistancetur(tur):
    d=0
    for i in range(1,len(tur)):
        x1=places[tur[i-1]][0]
        y1=places[tur[i-1]][1]
        x2=places[tur[i]][0]
        y2=places[tur[i]][1]
        d=d+distance(x1,y1,x2,y2)
    x1=places[tur[len(tur)-1]][0]
    y1=places[tur[len(tur)-1]][1]
    x2=places[tur[0]][0]
    y2=places[tur[0]][1]
    d=d+distance(x1,y1,x2,y2)
```

```
return d
 g=0.99
 max_temp=10000
 temperature=max_temp
 max_iter=15000
 o=0
 while (o<max_iter):
     oldDistances=totaldistancetur(tur)
     [i,j]=sorted(random.sample(range(n),2))

 newTur=tur[:i]+tur[j:j+1]+tur[i+1:j]+tur[i:i+1]+tur[j+1:]
     newDistances=totaldistancetur(newTur)
     m = random.random()
     if newDistances < oldDistances or
 math.exp((oldDistances-newDistances)/temperature)>m:
         tur=copy.copy(newTur)
     temperature=temperature*g
     o=o+1
 print("tur terbaik adalah",tur)
 print("distance terbaik adalah",totaldistancetur(tur))
 plt.plot([places[tur[i%n]][0] for i in range(n+1)],
 [places[tur[i%n]][1] for i in range(n+1)],'xb-');
 elapsed_time = time.time() - start_time
 print("waktu yang dibutuhkan adalah",elapsed_time)
```

Case I

In case I, several testings were done with the parameters as presented in Table 2.

Table 2. Results of case I with several different parameters and number of iterations

Number of Iterations	T_0	a	Total Distance (km)
8000	10000	0.99	461.72
9000	10000	0.99	422.69
10000	10000	0.99	410.09
15000	10000	0.9	407.82
15000	10000	0.99	394.34

After doing several experiments using different parameters and iterations, the smallest approach distance is 394.34 km with 15000 iterations, and it takes 14 seconds. The route with the smallest distance obtained is presented in Figure 2.

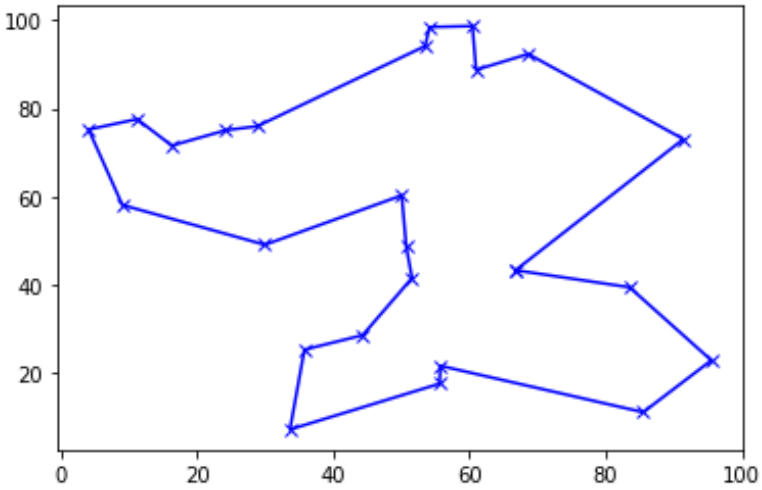

Figure 2. The optimal route of case I obtained by SA.

Case II

In case II, several experiments have been done using parameters such as Table 3.

Table 3. Results of case II with several different parameters and number of iterations

Number of Iterations	T_0	a	Total Distance (km)
15000	10000	0.99	700.55
30000	10000	0.99	692.99
35000	10000	0.99	676.65
50000	10000	0.99	659.21
50000	10000	0.999	587.61

From the experimental results using different parameters and iterations, the smallest approach distance is 587.61 km with 50000 iterations, and it takes 1 minute 15 seconds. The route with the smallest distance is presented in Figure 3.

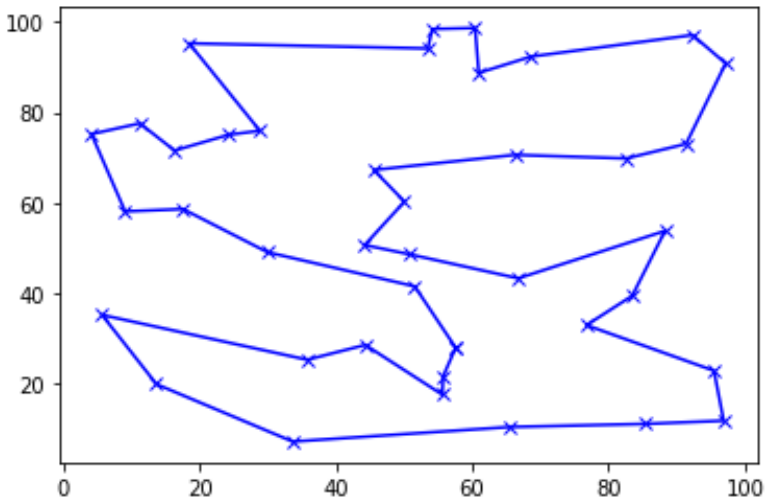

Figure 3. The optimal route of case II obtained by SA.

Case III

In case III several experiments were carried out using the parameters as shown in Table 4.

Table 4. Results of case III with several different parameters and number of iterations

Number of Iterations	T_0	a	Total Distance (km)
200000	10000	0.99	861.82
1500000	10000	0.99	774.79
2000000	10000	0.99	763.97
3000000	10000	0.99	755.35
3000000	10000	0.999	727.90

After conducting several experiments using different parameters and iterations, the smallest approach distance is 727.90 km with 3000000 iterations, and it takes 1 hour 18 minutes 42 seconds. The route with the smallest distance obtained is shown in Figure 4.

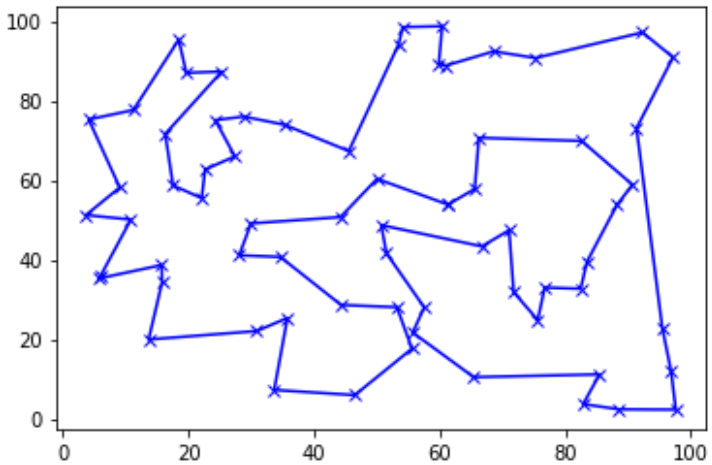

Figure 4. The optimal route of case III obtained by SA.

2. Solution using Branch and Bound (B-and-B)

In this scientific paper, LINGO 17.0 software is used to find TSP solutions using B-and-B with the syntax as follows.

```
MODEL:
SETS:
 CITY / 1.. 25/: U;
 LINK(CITY, CITY):
      DIST,
         X;
ENDSETS
DATA:
 DIST = @OLE('D:\data1.xlsx','Q') ;
ENDDATA

N = @SIZE(CITY);
 MIN = @SUM(LINK: DIST * X);
```

```
 @FOR( CITY(K):
  @SUM( CITY(I)| I #NE# K: X(I, K)) = 1;
  @SUM( CITY(J)| J #NE# K: X(K, J)) = 1;
  @FOR( CITY(J)| J #GT# 1 #AND# K #GT# 1:
      U(J) - U(K) + 1 <= (N - 1) * (1 - X(J, K))
 );
 );
 @FOR(LINK: @BIN(X));
END
```

Case I

In case I, there are 25 places and 88580 iterations are carried out. After executing it using LINGO 17.0 software, the minimum distance is 394.34 km and it takes 27 seconds. The output obtained using the LINGO 17.0 software is presented in Figure 5.

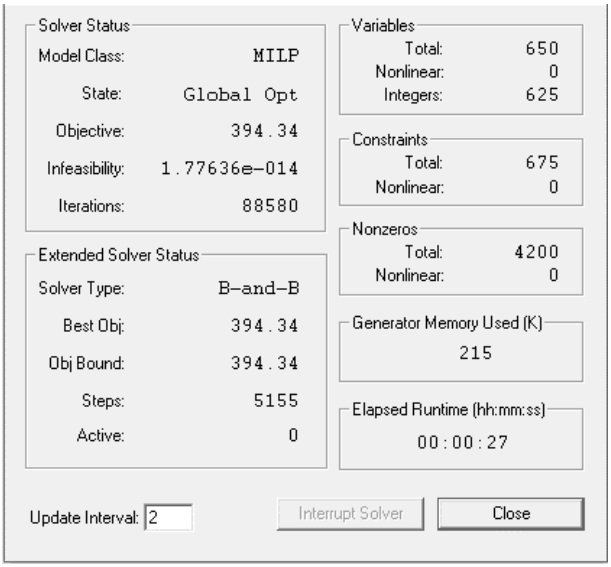

Figure 5. Output for case I using LINGO

Case II

In case II there are 40 places and 631296 iterations are carried out. After being executed using LINGO 17.0 software, the minimum distance is obtained which is 546.448 km and it takes 3 minutes 5 seconds. The output using the LINGO 17.0 is presented in Figure 6.

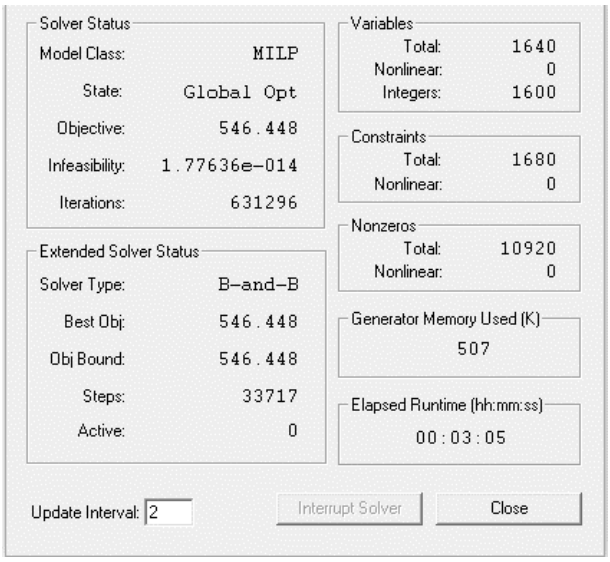

Figure 6. Output for case II using LINGO

Case III

In case III there are 68 places and 546168160 iterations are carried out. After being executed using LINGO 17.0, the minimum distance is obtained, namely 669.449 km and it takes 44 hours 44 minutes 41 seconds. The output using the LINGO 17.0 is presented in Figure 7.

Figure 7. Output for case II using LINGO

3. Results Comparison

Table 5 compares the total distance, the number of iterations, and the time required by the two methods used, the exact method, namely Branch and Bound (B-and-B), and the approximation method, namely SA. The B-and-B method produces the optimal distance, while the SA method produces the approximate distance. In case I, the percentage difference between the distance generated using the SA method with the B-and-B method is 0%, in case II it is 7% and in case III it is 8%, as shown in Table 5

Table 5. Comparison of the results obtained in cases I, II, and III

Case	Total Distance(km)		Iteration		Time (hour: minute: second)	
	B-and-B	SA	B-and-B	SA	B-and-B	SA
I	394.34	394.34	88580	15000	00:00:27	00:00:14
II	546.448	587.61	631296	50000	00:03:05	00:01:15
III	669.449	727.90	546168160	3000000	44:44:41	01:18:42

Based on Table 5, it can also be seen that the number of iterations and the time required by the SA method is much less and faster. For case I, the number of iterations of the SA method is 8 times less and the time required is 2 times faster than the B-and-B method. For case II, the number of iterations of the SA method is 12 times less and the time required is 2 times faster than that of the B-and-B method. For case III, the number of iterations of the SA method is 182 times less and the time required is 34 times faster than that of the B-and-B method.

D. CONCLUSIONS AND SUGGESTIONS

The results showed that the SA method could be used to solve TSP problems. But just like other metaheuristic methods, SA only accomplishes it using an approach so that it gets good results, but it cannot be determined that SA has the most optimal results, but the time needed by the SA method is faster than the B-and-B method. From the three cases that have been resolved using the SA method above, it can be seen that for case I where there are 25 places, the SA method has a difference of 0% from the B-and-B method, while the time needed by the SA method is 2 times faster than method B -and-B. For case II where there are 40 places, the SA method has a difference of 7% from the B-and-B method with the time needed 2 times faster than the B-and-B method. For case III where there are 68 places, the SA method has a difference of 8% from the B-and-B method but the time required is 34 times faster than the B-and-B method.

ACKNOWLEDGEMENT

I would like to thank my colleagues and students involved in this research. I also thank the mathematics department of FMIPA IPB which has provided financial assistance for this research.

REFERENCES

Alipour, M. M., Razavi, S. N., Feizi Derakhshi, M. R., & Balafar, M. A. (2018). A hybrid algorithm using a genetic algorithm and multiagent reinforcement learning heuristic to solve the traveling salesman problem. *Neural Computing and Applications*, *30*(9). https://doi.org/10.1007/s00521-017-2880-4

Bayram, H., & Şahin, R. (2013). A new simulated annealing approach for travelling salesman problem. *Mathematical and Computational Applications*, *18*(3), 313–322. https://doi.org/10.3390/mca18030313

Benhida, S., & Mir, A. (2018). Generating subtour elimination constraints for the Traveling Salesman Problem. *IOSR Journal of Engineering*, *8*(7), 17–21.

Botsali, A. R., & Alaykiran, K. (2020). Analysis of TSP: Simulated Annealing and Genetic Algorithm Approaches. *International Journal of Computational and Experimental Science and Engineering*, *6*(1). https://doi.org/10.22399/ijcesen.637445

Dorigo, M., & Gambardella, L. M. (1997). Ant colonies for the travelling salesman problem. *BioSystems*, *43*(2). https://doi.org/10.1016/S0303-2647(97)01708-5

Ezugwu, A. E. S., Adewumi, A. O., & Frîncu, M. E. (2017). Simulated annealing based symbiotic organisms search optimization algorithm for traveling salesman problem. *Expert Systems with Applications*, *77*. https://doi.org/10.1016/j.eswa.2017.01.053

Fournier, J. C. (2009). Graph Theory and Applications: With Exercises and Problems. In *Graph Theory and Applications: With Exercises and Problems*. Wiley Online Books. https://doi.org/10.1002/9780470611548

He, Q., Wu, Y. le, & Xu, T. W. (2018). Application of improved genetic simulated annealing algorithm in TSP optimization. *Kongzhi Yu Juece/Control and Decision*, *33*(2). https://doi.org/10.13195/j.kzyjc.2016.1666

Jünger, M., Reinelt, G., & Rinaldi, G. (1995). Chapter 4 The traveling salesman problem. In *Handbooks in Operations Research and Management Science* (Vol. 7, Issue C). https://doi.org/10.1016/S0927-0507(05)80121-5

Lalang, D., Silalahi, B. P., & Bukhari, F. (2018). Vehicle Routing Problem Time Windows Dengan Pengemudi Sesekali. *Journal of Mathematics and Its Applications*, *17*(2), 87–98. https://doi.org/10.29244/jmap.17.2.87-99

Liu, F., & Zeng, G. (2009). Study of genetic algorithm with reinforcement learning to solve the TSP. *Expert Systems with Applications*, *36*(3 PART 2). https://doi.org/10.1016/j.eswa.2008.08.026

Making, S. R. M., Silalahi, B. P., & Bukhari, F. (2018). Multi Depot Vehicle Routing Problem dengan Pengemudi Sesekali. *Journal of Mathematics and Its Applications*, *17*(1), 75–86. https://doi.org/10.29244/jmap.17.1.75-86

Othman, Z. A., Al-Dhwai, N. H., Srour, A., & Diyi, W. (2017). Water flow-like algorithm with simulated annealing for travelling salesman problems. *International Journal on Advanced Science, Engineering and Information Technology*, *7*(2). https://doi.org/10.18517/ijaseit.7.2.1837

Qamar, M. S., Tu, S., Ali, F., Armghan, A., Munir, M. F., Alenezi, F., Muhammad, F., Ali, A., & Alnaim, N. (2021). Improvement of traveling salesman problem solution using hybrid algorithm based on best-worst ant system and particle swarm optimization. *Applied Sciences (Switzerland)*, *11*(11). https://doi.org/10.3390/app11114780

Rehab, F. (2011). Fuzzy Particle Swarm Optimization with Simulated Annealing and Neighborhood Information Communication for Solving TSP. *International Journal of Advanced Computer Science and Applications*, *2*(5). https://doi.org/10.14569/ijacsa.2011.020503

Rere, L. M. R., Fanany, M. I., & Arymurthy, A. M. (2015). Simulated Annealing Algorithm for Deep Learning. *Procedia Computer Science*, *72*, 137–144. https://doi.org/10.1016/j.procs.2015.12.114

Rokbani, N., Kumar, R., Abraham, A., Alimi, A. M., Long, H. V., Priyadarshini, I., & Son, L. H. (2021). Bi-heuristic ant colony optimization-based approaches for traveling salesman problem. *Soft Computing*, *25*(5). https://doi.org/10.1007/s00500-020-05406-5

Silalahi, B. P., Fathiah, N., & Supriyo, P. T. (2019). Use of Ant Colony Optimization Algorithm for Determining Traveling Salesman Problem Routes. *Jurnal Matematika "MANTIK," 5*(2), 100–111. https://doi.org/10.15642/mantik.2019.5.2.100-111

Silalahi, B. P., Fatihin, K., Supriyo, P. T., & Guritman, S. (2020). Algoritme Sweep dan Particle Swarm Optimization dalam Optimisasi Rute Kendaraan dengan Kapasitas. *Jurnal Matematika Integratif*, *16*(1), 29. https://doi.org/10.24198/jmi.v16.n1.27474.29-40

Stodola, P., Michenka, K., Nohel, J., & Rybanský, M. (2020). Hybrid algorithm based on ant colony optimization and simulated annealing applied to the dynamic traveling salesman problem. *Entropy*, *22*(8). https://doi.org/10.3390/E22080884

Wang, L., Cai, R., Lin, M., & Zhong, Y. (2019). Enhanced List-Based Simulated Annealing Algorithm for Large-Scale Traveling Salesman Problem. *IEEE Access*, *7*. https://doi.org/10.1109/ACCESS.2019.2945570

Yang, L., Hu, X., Li, K., Ji, W., Hu, Q., Xu, R., & Wang, D. (2020). Nested Simulated Annealing Algorithm to Solve Large-Scale TSP Problem. *Communications in Computer and Information Science*, *1205 CCIS*. https://doi.org/10.1007/978-981-15-5577-0_37

Yu, M. (2019). A solution of TSP based on the ant colony algorithm improved by particle swarm optimization. *Discrete and Continuous Dynamical Systems - Series S*, *12*(4–5). https://doi.org/10.3934/dcdss.2019066

Yu, Y. Y., Chen, Y., & Li, T. Y. (2014). Improved genetic algorithm for solving TSP. *Kongzhi Yu Juece/Control and Decision*, *29*(8). https://doi.org/10.13195/j.kzyjc.2013.0598

Zhan, S., Lin, J., Zhang, Z., & Zhong, Y. (2016). List-Based Simulated Annealing Algorithm for Traveling Salesman Problem. *Computational Intelligence and Neuroscience*, *2016*, 1–12. https://doi.org/10.1155/2016/1712630

Zhong, W. L., Zhang, J., & Chen, W. N. (2007). A novel discrete particle swarm optimization to solve traveling salesman problem. *2007 IEEE Congress on Evolutionary Computation, CEC 2007*. https://doi.org/10.1109/CEC.2007.4424894