Accredited by National Journal Accreditation (ARJUNA) Managed by
Ministry of Education, Culture, Research and Technology with Second Grade (Sinta 2)
since year 2021 to 2026 according to the decree No. 158/E/KPT/2021

Published online on the journal's webpage: **http://jurnal.iaii.or.id**

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 5 No. 6 (2021) 1090 - 1098 ISSN Media Elektronik: 2580-0760

Numerical Approach of Symmetric Traveling Salesman Problem Using Simulated Annealing

Iryanto[1], P. H. Gunawan[2]
[1]Informatics Department, Politeknik Negeri Indramayu
[2]School of Computing, Telkom University
[1]iryanto@polindra.ac.id, [2]phgunawan@telkomuniversity.ac.id

Abstract

The aim of this paper is to elaborate the performance of Simulated Annealing (SA) algorithm for solving traveling salesmen problems. In this paper, SA algorithm is modified by using the interaction between outer and inner loop of algorithm. This algorithm produces low standard deviation and fast computational time compared with benchmark algorithms from several research papers. Here SA uses a certain probability as indicator for finding the best and worse solution. Moreover, the strategy of SA as cooling to temperature ratio is still given. Thirteen benchmark cases and thirteen square grid symmetric TSP are used to see the performance of the SA algorithm. It is shown that the SA algorithm has promising results in finding the best solution of the benchmark cases and the squared grid TSP with relative error 0 - 7.06% and 0 – 3.31%, respectively. Further, the SA algorithm also has good performance compared with the well-known metaheuristic algorithms in references.

Keywords: simulated annealing, traveling salesman problem, symmetric TSP, square grid TSP.

1. Introduction

Finding an optimal route or shortest path is quite challenging due to its own difficulties. The problem can be implemented in some fields. One of them is finding minimum route for a salesman who want to visit place of all of his clients with constraint the places are visited exactly once and the salesman ends his tour in place in which he starts the journey. The problem is well known as TSP (traveling salesman problem) where the problem can be described in weighted graph. Here, the vertex and edges of the graph are used to describe city and distance between two cities, respectively. Moreover, this problem is also known as Hamiltonian circuits problem.

In recent years, many researchers developed and proposed methods to solve the problem. Several methods that has been proposed to solve the problem are discrete tree-seed algorithm (DTSA)[1], black hole algorithm (BHA) [2], discrete lion swarm optimization [3], a hybrid optimization algorithm based on wolf pack search and local search [4], heuristic shortest path algorithm [5], genetic algorithm (GA) [6], particle swarm optimization (PSO) [6-8], ant colony optimization (ACO) [7], shuffled frog leaping algorithms (SFLA) [7], and simulated annealing (SA) [9-18]. From [2], BHA has a good performance compared with GA, ACO, dan PSO for finding the optimal solution and computational time in 10 benchmark cases of TSP using 22 - 101 number of cities. Meanwhile in [7], ACO known has a good performance compared PSO, improved PSO dan SFLA algorithm for finding the optimal solution of six benchmark cases of TSP with totally 30 – 100 cities. However, comparing the computational time, PSO is shown slower than the other algorithms.

Considering the work of Chunhua, et al. in [19], it is shown that STA had the best performance to find the optimal solution and the fastest in computational time for solving three benchmark cases of TSP with the number of cities 16, 48, and 52, compared with ACO and SA. Moreover, STA had low standard deviation from three cases in [19]. This standard deviation is used to show the stability and reliability of algorithm for finding the optimal solution [2]. In [19], each one of algorithms is running in 20 times parallel for solving some problems.

In this paper, SA algorithm will be elaborated to solve the TSP. Moreover, to see the performance of the SA algorithm, comparison of SA and well-known metaheuristic algorithm such as BHA, GA, PSO, ACO and state transition algorithm (STA) will be given for

Accepted by Editor: 08-10-2021 | Final Revision: 07-12-2021 | Online Publication: 30-12-2021

solving some benchmark cases of TSP. Here, SA will be modified by using combination of outer and inner loop in algorithm. In this case, the input of outer loop (the number of experiments) will depend on the result of inner loop (the proses for finding optimal solution). With the modification, it is expected that the performance of the SA can improve.

There are two kinds of TSP namely symmetric and asymmetric TSP [20]. The symmetric TSP is when the distance between city A to city B is the same with the distance between city B to city A. Whereas when the distance is different, the problem is known as Asymmetric TSP. Note that, all mentioned benchmark cases above are belong to the symmetric TSP. The performance comparison will be conducted with three references, [2], [7], and [19]. From these references, there are twelve symmetric TSP benchmark cases where data of each case can be found in [21]. The benchmark cases are Ulysess16, Ulysess22, Bays29, Bayg29, Att48, Eil51, Berlin52, St70, Eil76, Gr96, KroA100, and Eil101. Therefore, the SA algorithm will be focused on solving the symmetric TSP using the benchmark cases to see its performance compared with other algorithms mentioned in those references.

To see further performance of the SA algorithm, another simulation in solving the symmetric TSP cases is conducted. In this case, $n \times n$ cities are well-order generated with distance between two neighboring cities is equal to one. Here, the case is called by name 'square grid symmetric TSP'. The simulations are conducted for $n = 3 - 15$. The values of n are chosen to see performance of the algorithm in solving symmetric TSP cases with small to medium number of cities. Using the values of n, numbers of city of the simulations are varied from 9 to 225.

2. Research Method

The simulated annealing (SA) is an optimization method that can be used to solve TSP. The method is used to find shortest route from all possible routes [11]. The optimum solution is obtained when the energy (distance) produced is the minimum and the path tracking is obtained from the route taken. As an optimization method, the difference between SA and other optimization methods is that there is a possibility to still accept worse solution than the current solution to elude local optimal solution [10]. Generally, the SA algorithm is probabilistic meta-heuristic method inspired by annealing of metal [11]. The algorithm was introduced in combinatorial problem to solve TSP by Kirkpatrick et al. in 1983 [9]. The idea to accept the worse solution in the process is the main idea that differentiates the algorithm with other optimization methods. The acceptance depends on probability function, p, as follows [9]

$$p(\Delta E, T) = \begin{cases} e^{\frac{-\Delta E}{T}} & \Delta E > 0 \\ 1 & \Delta E \leq 0 \end{cases} \quad (1)$$

Where ΔE denotes the energy difference and T represents the temperature.

Note that the energy difference is less or equal than zero, $\Delta E \leq 0$, means that the current solution is better than previous solution. Whereas the difference energy greater than zero, $\Delta E > 0$, means that the current solution is worse than previous solution. Criteria for acceptance of the worse solution depends on certain probability, $pr < e^{\frac{-\Delta E}{T}}$, where pr is random value between 0 and 1. In this case, the method utilizes random numbers in the process, so that each experiment, may produce different results and it is still possible that the solution obtained will be trapped in the local optimum solution. Therefore, it takes several trials (running program) to find out the optimum solution.

The cooling process of the temperature is one of factors that need to be considered. At the beginning, the initial value of the temperature is set. The greater the initial temperature used, the wider the range of the random search process [11]. The initial temperature will continue to decrease as the iteration goes to increase. Following [9] and [10], the geometric cooling schedule, $T = T \times r$, is used in here. Value of the cooling coefficient, r, is constant between 0 and 1 [9]. Further, value of the coefficient for slow cooling rate is between 0.8 to 0.99 [10]. Therefore, in this article, the cooling rate is $r = 0.9$.

In the SA method, at the beginning the initial of solution, S_0, is given. The solution is randomly generated. Following Algorithm 1 is given the modified SA procedures:

Algorithm 1

SA procedures by using inner and outer loop interaction

Set the initial value T, S_0 using (2), E_0, S_{opt} and E_{opt}
For 1 to n
    Update State S, Calculate Energy E using (3)
    Calculate $\Delta E = E - E_0$, $\omega \in [0,1]$, p using (1)
    If $\Delta E < 0$ then
        Set $S_0 = S$, $E_0 = E$
        If $E < E_{opt}$ then $S_{opt} = S_0$, $E_{opt} = E_0$
            Else go to Step 11
    Else
        If $p > pr$ then go to Step 5
        Else go to Step 11
    If satisfying Inner Loop Termination Criteria, then
        Do cooling schedule $T = T \times r$
        Go to Step 3
End For

The State (S) is used to track the path taken, for example, there are M cities to be taken, the state is a number from 1 to M on the condition that no number is similar. In

DOI: https://doi.org/10.29207/resti.v5i6.3549

other words, the state is a random permutation of the M cities. In this article, the initial state is created as

$$S_0 = \{1, 2, 3, 4, 5 \dots, M\} \quad (2)$$

The state is updated using reversion procedure. In this case, two random values within M (k1 and k2) are created and position of the state in the range are inversed as described in [16].

Energy (E) is used as an objective function. In this case, it is used to calculate the total distance travelled in one state. Note that in the TSP, the beginning and the end of the route is the same so the additional distance from the last travelled city to the first city is added.

$$E = \sum_{i=1}^{N-1} d_i + d_N \quad (3)$$

Where d_i denotes distance between two cities in the state. Position of the cities is expressed in Cartesian coordinates so that the distance can be expresses as given in equation (4).

$$d_i = \sqrt{\left(S_x(i) - S_x(i+1)\right)^2 + \left(S_y(i) - S_y(i+1)\right)^2}$$
$$d_N = \sqrt{\left(S_x(N) - S_x(1)\right)^2 + \left(S_y(N) - S_y(1)\right)^2} \quad (4)$$

Termination of the program can use given maximum iteration or it can use the specified/final temperature. In this article, the stopping criteria is the maximum iteration.

3. Result and Discussion

In this paper, two simulations of symmetric TSP are conducted. The first simulation is symmetric TSP based on Data in [21] and the second is square grid symmetric TSP generated in certain ways. In this case $n \times n$ points are well-order generated with distance between two neighboring points is equal to one. Results of the SA algorithm are compared with best known solution (BKS) from references for the symmetric case and analytic solution for the square grid case. Further, comparison with some algorithms is carried out to see the SA algorithm performance. All simulations in this paper are carried out using a PC with windows 10 pro 64-bit (OS), Intel® Core™ i7-8550U CPU @ 1.80 GHz processor and 16 GB RAM memory. The SA algorithm is based on C++ language. All simulations are run with parameter $T = 1000$ (initial temperature), $r = 0.9$, maximum iteration of outer loop is 100, maximum iteration of inner loop is 800*number of cities.

3.1. Symmetric TSP

In this case, all points are generated using TSPLIB data that can be downloaded in [21]. Optimal route of the cases is also available in the reference. The simulations are run for thirteen TSP cases such as Ulysess16, Ulysess22, Bays29, Bayg29, Att48, Eil51, Berlin52, St70, Eil76, Gr96, KroA100, Eil101, and Ch130. The number in the name of the case represents number of cities. Number of cities in the cases are 16, 22, 29, 29, 48, 51, 52, 70, 76, 96, 100, 101, and 130, respectively. These cases are chosen to conduct comparison with some references. Results of the simulations are presented in Table 1.

Table 1. Results of the Symmetric TSP Cases

Case	BS	WS	Ave	IL	Rep	BI	BT (s)
Ulysess16	73.9876	74.4602	74.2239	12800	2	593	0.333
Ulysess22	75.3097	76.8119	76.2203	17600	14	1334	4.349
Bays29	9074.15	9360.715	9202.625	23200	5	622	1.239
Bayg29	9074.15	9576.29	9259.601	23200	38	889	13.517
Att48	33882.48	34846.67	34766.32	38400	12	38400	6.367
Eil51	430.89	444.74	444.41	40800	52	40800	31.465
Berlin52	7544.366	8341.865	7845.615	41600	93	41600	57.396
St70	697.8861	727.8492	708.807	56000	31	56000	32.143
Eil76	563.6019	594.2049	575.5559	60800	86	60800	106.068
Gr96	539.9611	558.2102	550.0152	76800	11	76800	20.599
KroA100	21632.56	24181.945	22956.741	80000	17	80000	32.281
Eil101	673.4284	708.6054	688.9844	80800	34	80800	67.967
Ch130	6356.304	6657.21	6443.351	104000	91	104000	298.951

In the Table 1, notation BS means the best solution of the SA algorithm, WS is the worst solution of the SA algorithm, Ave is average of all solution of the SA algorithm, IL is maximum of number iterations of the inner loop, Rep means repetition of the outer loop, BI is the best iteration of inner loop to get the best solution,

DOI: https://doi.org/10.29207/resti.v5i6.3549

and BT is the total of time (in second) for inner loop and outer loop to get the best solution. To see performance of the SA algorithm, comparison between the results with the best-known solution (BKS) from references are executed. In this case, we calculate relative error using equation (1) for the best solution, the worst solution, and average of the solution. Further, average time is calculated to see average time for one inner loop using equation (2). The performance of the algorithm is given in Table 2.

It is shown in the Table 2 that results of the SA algorithm are acceptable. According to the relative error, the algorithm has good agreement with the BKS for the conducted cases. The algorithm gets the BKS for Ulysess16, Ulysess22, Bays29, and Bayg29 case. Whereas for the other cases it can be said that the algorithm is also has good comparison with the BKS. The best solution of the algorithm has the highest and the smallest relative error 7.06% and 0.0%, respectively. The worst solution has relative error 0.64% - 13.63%. Whereas average of the solutions has relative error 0.32% - 9.54%. It is clearly shown in the table that the best performance is in the Ulysess16 case in which there are 16 points/cities. When the number of points/cities is bigger the performance is decreasing, and the average time is increasing.

$$BE = \frac{BS - BKS}{BKS}, WE = \frac{WS - BKS}{BKS}, \quad AE = \frac{Ave - BKS}{BKS}. \tag{5}$$

$$Average\ Time = \frac{BT}{Rep}. \tag{6}$$

Table 2. Performance of the SA algorithm for the Symmetric TSP Cases

Case	BKS	BS	BE	WE	AE	Average Time (s)
Ulysess16	73.9876 [19]	73.9876	0.0000	0.0064	0.0032	0.1665
Ulysess22	75.3097 [2]	75.3097	0.0000	0.0199	0.0121	0.3106
Bays29	9074.15 [8]	9074.15	0.0000	0.0316	0.0142	0.2478
Bayg29	9074.15	9074.15	0.0000	0.0553	0.0204	0.3557
Att48	33522	33882.48	0.0108	0.0395	0.0371	0.5306
Eil51	426	430.89	0.0115	0.0440	0.0432	0.6051
Berlin52	7542	7544.366	0.0003	0.1061	0.0403	0.6172
St70	675	697.8861	0.0339	0.0783	0.0501	1.0369
Eil76	538	563.6019	0.0476	0.1045	0.0698	1.2333
Gr96	514 [22]	539.9611	0.0505	0.0860	0.0701	1.8726
KroA100	21282	21632.56	0.0165	0.1363	0.0787	1.8989
Eil101	629	673.4284	0.0706	0.1266	0.0954	1.9990
Ch130	6110	6356.304	0.0403	0.0896	0.0546	3.2852

Comparison is also conducted with other algorithms in several references. The first comparison is with algorithms in [7]. In the reference, there are six algorithms such as ACO, PSO, improved PSO (IPSO), SFLA, order crossover with inversion mutation (OXIM) SFLA, and cycle crossover with inversion mutation (CXIM) SFLA. The algorithm was used to solve six symmetric TSP cases namely Oliver30, Eil51, Berlin52, St70, Eil76, and KroA100. All simulations in the reference are based on MATLAB 6.5 tool and run in PC with a 1.70 GHz processor and 4.00 GB RAM memory. All results given in the reference were gotten after the program was run 1000 times and the time in the reference was an average running time. It is shown in the reference that ACO and Improved PSO have better results according to their error.

Comparisons between the SA algorithm and the other algorithms are carried out for the cases minus Oliver30 case because data of the case are not available in [21]. Results of the comparison are presented in Table 3. Note that maximum number of iterations of ACO, PSO, and IPSO in the reference for each TSP case is 1000. The number of ants in ACO is 100 and the number of populations in PSO and IPSO is equal to number of cities. Further, it is stated in the reference that in case of large number of cities, number of ants may be increased in ACO and number of iterations may be increased in IPSO. Thus, the setting of inner loop which is equal to number of cities times 800 is acceptable.

It is shown in the Table 3 that the SA algorithm has better results than the algorithms in [7] for the TSP cases. The SA algorithm has superior performance than the algorithms in the references. It can be seen also that computational time of the SA algorithm is better than the mentioned algorithm. But the PC specification to run the SA algorithm is better than the PC specification to run the other algorithm. Note that, value of the average time

DOI: https://doi.org/10.29207/resti.v5i6.3549

of the SA algorithm is average time for one outer loop iteration.

To see further performance of the SA algorithm, another comparison with other algorithms in [2] and [19] is conducted. In this case, there are five compared parameters such as best solution, worst solution, average of solution, standard deviation, and time. In [2], there are four algorithms such as ACO, PSO, GA, and BHA that are compared each other to solve ten symmetric TSP cases; Ulysess22, Bays29, Bayg29, Att48, Eil51, Berlin52, St70, Eil76, Gr96, and Eil101. All simulations in the reference are based on MATLAB and were executed on PC with Intel® Core™ 2 Duo CPU @ 2 GHz processor and 2 GB RAM memory. Results of the algorithms using 100 population size, 200 iterations, and 5 independent running are taken and compared with the SA algorithm and presented in Table 4. To represent the reference setting, the SA algorithm runs with 20000 inner loop iterations and 5 outer loop iterations.

In [19], performances of SA, ACO, and state transition algorithm (STA) are compared each other to solve Ulysesss16, Att48, and Berlin52 TSP. All the simulations in the reference are based on MATLAB and were run on PC with Intel® Core™ i3-2310M CPU @ 2.10 GHz processor. Note that, in the reference [19], the run time is the average time used in 20 execution and the maximum iteration for SA is 4000. Therefore, to adopt the reference, iterations of inner and outer loop are set 4000 and 20, respectively. Comparison results are given in Table 5.

Table 3. Performance Comparison between the SA algorithm and algorithms in [7] for the TSP Cases

Problem	Algorithm	Best	Mean	Error	Average Time
Eil51 (51 cities) BKS – 426	ACO	443	516	0.0399	194
	PSO	908	1313	1.1315	9
	IPSO	464	543	0.0892	259
	SFLA	1169	1703	1.7441	1136
	OXIMSFLA	534	593	0.2535	16241
	CXIMSFLA	671	671	0.5751	5147
	SA	**430.9**	**444.4**	**0.0115**	**0.6051**
Berlin52 (52 cities) BKS – 7542	ACO	7549	9385	0.0009	276
	PSO	17296	22206	1.2933	15
	IPSO	7816	8723	0.0363	469
	SFLA	19865	30598	1.6339	1150
	OXIMSFLA	8362	9987	0.1087	21907
	CXIMSFLA	12266	15109	0.6264	5732
	SA	**7544.4**	**7845.6**	**0.0003**	**0.6172**
St70 (70 cities) BKS – 675	ACO	707	888	0.0474	1678
	PSO	2009	3411	1.9763	28
	IPSO	755	871	0.1185	1058
	SFLA	2615	3759	2.8741	1859
	OXIMSFLA	892	1004	0.3215	1771
	CXIMSFLA	1355	1734	1.0074	3042
	SA	**697.9**	**708.8**	**0.0339**	**1.0369**
eil76 (76 cities) BKS – 538	ACO	573	659	0.0651	2035
	PSO	1662	1975	2.0892	28
	IPSO	584	641	0.0855	3036
	SFLA	1904	2580	2.5390	1771
	OXIMSFLA	733	819	0.3625	99104
	CXIMSFLA	1072	1326	0.9926	28007
	SA	**563.6**	**575.6**	**0.0476**	**1.2333**
KroA100 (100 cities) BKS – 21282	ACO	22388	28655	0.0520	4516
	PSO	113174	191394	4.3178	45
	IPSO	24596	28385	0.1557	9426
	SFLA	128520	175413	5.0389	3042

DOI: https://doi.org/10.29207/resti.v5i6.3549

Table 3. Performance Comparison between the SA algorithm and algorithms in [7] for the TSP Cases

Problem	Algorithm	Best	Mean	Error	Average Time
	OXIMSFLA	37212	41371	0.7485	197264
	CXIMSFLA	58069	78451	1.7285	52365
	SA	**21632.6**	**22956.7**	**0.0165**	**1.8989**

Table 4 shows performance comparison of the SA algorithm and the other algorithms in [2]. It is clearly shown in the table that results of the SA algorithm are better than results of the other algorithms in almost every case and parameter. The SA algorithm has better performance in finding the best solution of each case except for the Ulysess22 case in which the algorithm gets the same results with the BHA algorithm. In terms of computational time, it is hard to compare due to the PC specification but note that the column time for the SA algorithm is the total time for running the program with 200 iterations of inner loop and 5 iterations of outer loop.

Table 4. Performance Comparison between the SA algorithm and algorithms in [2] for the TSP Cases

Problem	Algorithm	Best	Worst	Average	Std. Dev	Time
ulysses22 (22 cities) BKS – 75.3097	ACO	75.3984	**75.8409**	75.4869	0.19789	84.27123
	PSO	75.9104	77.1857	76.2186	0.55273	61.87992
	GA	75.7744	76.4434	75.9878	1.2307	63.39216
	BHA	75.3097	75.9343	75.6844	0.34208	50.44745
	SA	**75.3097**	76.09488	**75.5638**	**0.3253**	**1.515**
bays29 (29 cities) BKS – 9074.15	ACO	9239.197	11014.45	9823.202	722.4152	88.2566
	PSO	9120.339	9498.171	9195.905	168.9717	88.82869
	GA	9751.426	10513.91	10015.23	319.8788	57.11864
	BHA	9396.475	9507.17	9463.252	**60.9588**	52.1048
	SA	**9076.983**	**9299.9**	**9166.15**	109.206	**1.375**
bayg29 (29 cities) BKS – 9074.15	ACO	9447.493	11033.55	9882.22	675.8331	99.95724
	PSO	9329.251	11332.72	9947.026	799.4073	75.29661
	GA	9579.123	10411.2	9771.954	127.1131	56.16117
	BHA	9375.442	**9375.442**	9375.442	**0**	45.87095
	SA	**9094.635**	9752.892	**9314.914**	278.230	**1.642**
att48 (48 cities) BKS – 33522	ACO	35230.9	46204.24	39436.18	4874.295	133.4571
	PSO	36996.44	61421.99	47018.41	9685.894	84.73842
	GA	35312.52	50671.45	43620.64	2004.001	57.35453
	BHA	34200.86	35528.52	34473.84	589.8024	43.21174
	SA	**34056.95**	**34056.95**	**34056.95**	**0**	**1.917**
eil51 (51 cities) BKS – 426	ACO	454.3895	469.0531	461.0175	**6.2974**	59.19328
	PSO	469.1551	737.5258	574.8022	107.2371	57.25646
	GA	448.8397	**462.1142**	453.4773	9.4157	59.63916
	BHA	437.893	526.8977	458.9252	38.6365	44.39009
	SA	**429.484**	476.9822	**446.996**	17.4654	**2.568**
berlin52 (52 cities) BKS – 7542	ACO	**7757.026**	10541.12	8522.902	1152.2	65.07013
	PSO	9218.468	14279.43	11089.53	2067.932	68.64806
	GA	8779.756	9565.374	9288.448	1301.211	52.73534
	BHA	8188.071	9356.748	8455.83	508.9871	43.40446
	SA	7957.667	**8074.314**	**7997.202**	**49.713**	**1.907**
st70 (70 cities) BKS – 675	ACO	711.6515	855.2032	757.754	59.6079	94.56822
	PSO	1030.848	1756.123	1321.814	269.2793	55.28412
	GA	1112.308	1242.201	1158.846	52.1734	55.09585
	BHA	723.2691	1081.109	797.5745	125.2272	45.3308
	SA	**702.9316**	**705.1066**	**703.802**	**1.066**	**2.312**
eil76 (76 cities) BKS – 538	ACO	574.2404	665.9995	594.1442	40.2152	61.7418
	PSO	804.2667	1195.902	975.6397	152.4061	56.76708
	GA	619.2262	679.7864	652.0593	122.0972	46.69151
	BHA	566.243	925.8417	659.1021	152.1754	46.54038
	SA	**564.345**	**595.6808**	**582.495**	**11.924**	**2.499**
gr96 (96 cities) BKS – 514	ACO	555.7535	639.9167	580.5406	33.9301	84.38977
	PSO	1095.111	1728.824	1378.87	247.5099	56.21171
	GA	737.9671	748.3543	742.4275	4.3282	63.24444

DOI: https://doi.org/10.29207/resti.v5i6.3549

Table 4. Performance Comparison between the SA algorithm and algorithms in [2] for the TSP Cases

Problem	Algorithm	Best	Worst	Average	Std. Dev	Time
eil101 (101 cities) BKS – 629	BHA	546.8397	1197.876	807.2465	258.815	43.58791
	SA	**539.4268**	**539.4268**	**539.4268**	**0**	**2.969**
	ACO	725.0996	868.2047	763.9207	59.9684	89.63974
	PSO	1158.704	1973.819	1499.991	319.7468	62.09302
	GA	828.8806	854.4381	838.8307	9.9642	55.18821
	BHA	720.3838	1249.868	897.3813	210.1446	45.83337
	SA	**687.9949**	**704.1819**	**697.707**	**7.930**	**3.166**

Performance of the SA algorithm compared with algorithms in [19] is shown in Table 5. It is clearly shown that performance the SA algorithm is in good comparison with the other algorithms. In terms of the best solution, the SA algorithm has the same result with STA in Ulysess16 and Berlin52 TSP and better result in Att48 TSP case. Further, in Ulysess16 TSP case, the SA algorithm is better than the other algorithms in remaining parameters. Neglected the STA, the SA algorithm has the better performance compared with ACO given in [19]. Note that, value of the average time of the SA algorithm is average time for one outer loop iteration.

Table 5. Performance Comparison between the SA algorithm and algorithms in [19] for the TSP Cases

Problem	Algorithm	Best	Worst	Average	Std. Dev	Average Time
ulysess16 (16 cities) BKS – 73.9876	ACO	74.6287	78.7728	76.0864	1.1062	11.3038
	STA	73.9876	74.5939	74.0779	0.1626	1.2223
	SA	**73.9876**	**73.9998**	**73.9937**	**0.0061**	**0.2593**
att48 (48 cities) BKS – 33522	ACO	37015	39801	38449	862.4546	102.4784
	STA	33724	36205	34872	668.7553	3.0462
	SA	**33710.99**	36568.22	34946.77	777.57	**0.2442**
berlin52 (52 cities) BKS – 7542	ACO	8240.4	9151.3	8777.6	267.1124	118.0948
	STA	7544.4	**8630.5**	8247.2	273.4509	3.3438
	SA	**7544.4**	8918.8	**7706.5**	360.1	**0.2369**

In the SA algorithm, random value is used, therefore its result is influenced by the value. Depends on the value, the algorithm may get better result in faster computational time or it is possible to get worse solution since SA algorithm has possibility to accept worse solution leading to get local optimum. Repetitions of the program execution are to see the convergence of the results. The lower standard deviation shows that the related algorithm is more stable and reliable in finding the optimal/best solution [2].

3.2. Square Grid Symmetric TSP

In the square grid symmetric TSP, $n \times n$ cities are well-order generated with distance between two neighboring cities is equal to one. In this case, simulations are conducted for $n = 3 - 15$. The values of n are chosen to see performance of the algorithm in solving symmetric TSP cases with small to medium number of cities. Using the values of n, numbers of city of the simulations are varied from 9 to 225. Results of the simulations are presented in Table 6 and Table 7. The best solution of the case for $n = 7$ can be seen in Figure 1.

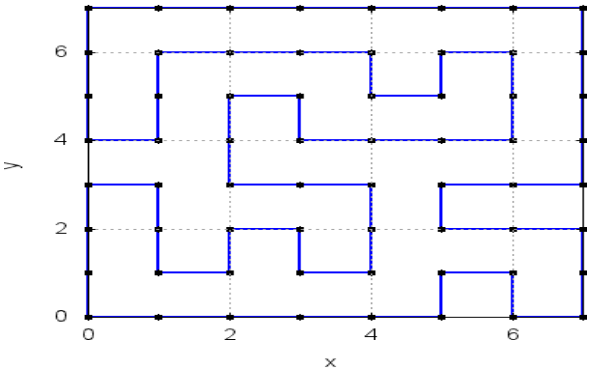

Figure 1. Solution of Square Grid Symmetric TSP for $n = 7$

Table 6. Results of the Square Grid TSP Cases

n	Number of City	BS	WS	Ave	IL	Rep	BI	BT
3	9	9.4142			7200	0	154	
4	16	16			12800	0	445	
5	25	25.4142			20000	0	1353	
6	36	36	36.8284	36.7249	28800	8	12353	2.841

DOI: https://doi.org/10.29207/resti.v5i6.3549

7	49	49.4142	51.0711	50.5188	39200	3	21047	1.216
8	64	64	66.4853	64.9527	51200	60	36647	54.219
9	81	81.4142	86.3848	82.57401	64800	65	57198	83.574
10	100	100.8284	104.1421	103.0376	80000	3	80000	4.377
11	121	123.0711	125.5563	124.5622	96800	5	96800	10.964
12	144	147.3137	152.2843	150.9036	115200	12	115200	43.283
13	169	174.3848	177.6985	176.5939	135200	3	135200	12.852
14	196	201.799	208.4264	206.2173	156800	3	156800	16.894
15	225	232.8701	237.8406	235.7136	180000	37	180000	330.465

Table 7. Performance of the SA algorithm for the Square Grid TSP Cases

n	Number of City	BKS	BS	Error	Average Time
3	9	9.4142	9.4142	**0**	
4	16	16	16	**0**	
5	25	25.4142	25.4142	**0**	
6	36	36	36	**0**	0.3551
7	49	49.4142	49.4142	**0**	0.4053
8	64	64	64	**0**	0.9037
9	81	81.4142	81.4142	**0**	1.2858
10	100	100	100.8284	0.0083	1.4590
11	121	121.4142	123.0711	0.0136	2.1928
12	144	144	147.3137	0.0230	3.6069
13	169	169.4142	174.3848	0.0293	4.284
14	196	196	201.799	0.0296	5.6313
15	225	225.4142	232.8701	0.0331	8.9315

It is shown in Table 7 that performance of the SA algorithm are acceptable. The algorithm has no error in finding the optimal solution for cases with number of cities are 9, 16, 25, 49, 84, and 81. For the remaining cases, the error is getting larger when the number of the cities are getting bigger. The error are 0.% - 3.31% which mean the performance of the SA algorithm is good.

4. Conclusion

Elaboration of simulated annealing algorithm using inner and outer loop has been carried out. Several simulations have been conducted to see performance of the algorithm. The simulations are carried out for 13 symmetric TSP cases taken from [21]. The algorithm shows promising performance with relative error; the best solution 0 - 7.06%, the worst solution 0.64% - 13.63%, and average of the solutions 0.32% - 9.54%. The algorithm also has good performance compared with the well-known metaheuristic algorithms. The SA algorithm has better performance in finding the best solution compared with the other algorithms in [2], [7], and [19]. Moreover, the SA algorithm has small enough standard deviation value 0 – 777.57 with average value 124.118. In the square grid TSP, the SA algorithm also has good performance with relative error 0 – 3.31%. The SA algorithm can find the best solution of the cases. However, in some cases the SA could not find the best solution. Here, existence of the random value influences result of the algorithm. For future work, adding local search algorithm is expected to increase the performance of SA. Moreover, decreasing the computational time of SA algorithm by parallel algorithm can be considered for the future work.

Acknowledgements

Authors want to say thank you to Telkom University for research funding.

References

[1] A. C. Cinar, S. Korkmaz, and M. S. Kiran. “A discrete tree-seed algorithm for solving symmetric traveling salesman problem.” Engineering Science and Technology, an International Journal, vol. 23, no. 4, pp. 879-890, 2020.
[2] A. Hatamlou. “Solving travelling salesman problem using black hole algorithm.” Soft Computing, vol. 22, no. 24, pp. 8167-8175, 2018.
[3] Z. Daoqing and J. Mingyan. “Parallel discrete lion swarm optimization algorithm for solving traveling salesman problem.” Journal of Systems Engineering and Electronics, vol. 31, no. 4 pp. 751-760, 2020.
[4] R. Dong, S. Wang, G. Wang, and X. Wang. “Hybrid optimization algorithm based on wolf pack search and local search for solving traveling salesman problem.” Journal of Shanghai Jiaotong University (Science), vol. 24, no. 1, pp. 41-47, 2019.
[5] S. A. Bakar and M. Ibrahim. “Optimal solution for travelling salesman problem using heuristic shortest path algorithm with imprecise arc length.” In AIP Conference Proceedings, AIP Publishing LLC, vol. 1870, no. 1, pp. 040061, 2017.
[6] Panda, Madhumita. “Performance Comparison of Genetic Algorithm, Particle Swarm Optimization and Simulated Annealing Applied to TSP.” International Journal of Applied Engineering Research, vol. 13, no. 9, pp. 6808-6816, 2018.
[7] S. Saud, H. Kodaz, and I. Babaoğlu, “Solving Travelling Salesman Problem by Using Optimization Algorithms” in The 9th International Conference on Advances in Information Technology, KnE Life Sciences, pp 17-32, 2017.
[8] X. Wang, A. Mu, and S. Zhu. “ISPO: A new way to solve traveling salesman problem.” Intelligent Control and Automation, vol. 4, no. 2, 2013.

DOI: https://doi.org/10.29207/resti.v5i6.3549

[9] Zhou, Ai-Hua, Li-Peng Zhu, Bin Hu, Song Deng, Yan Song, Hongbin Qiu, and Sen Pan. "Traveling-salesman-problem algorithm based on simulated annealing and gene-expression programming." Information, vol. 10, no. 1, 2019.
[10] A. E. S. Ezugwu, A. O. Adewumi, and M. E. Frîncu. "Simulated annealing based symbiotic organisms search optimization algorithm for traveling salesman problem." Expert Systems with Applications, vol. 77, pp. 189-210, 2017.
[11] Zhan, Shi-hua, Juan Lin, Ze-jun Zhang, and Yi-wen Zhong. "List-based simulated annealing algorithm for traveling salesman problem." Computational intelligence and neuroscience, 2016.
[12] A. Khumaidi, R. Raafi'udin, and I. P. Solihin. (2020) "Simulation Of Traveling Salesman Problem For Distribution Of Fruits In Bogor City With Simulated Annealing Method." Jurnal Mantik, vol. 3, no. 4, pp. 611-618, 2020.
[13] C. Wang, M. Lin, Y. Zhong, and H. Zhang. "Swarm simulated annealing algorithm with knowledge-based sampling for travelling salesman problem." International Journal of Intelligent Systems Technologies and Applications, vol. 15, no. 1, pp. 74-94, 2016.
[14] M. Makuchowski. "Effective algorithm of simulated annealing for the symmetric traveling salesman problem." In International Conference on Dependability and Complex Systems (pp. 348-359). Springer, Cham, July. 2018.
[15] X. Han, Y. Dong, L. Yue, and Q. Xu. "State transition simulated annealing algorithm for discrete-continuous optimization problems." IEEE Access, 7, 44391-44403, 2019.
[16] L. Xiong and S. Li. "Solving TSP Based on the Improved Simulated Annealing Algorithm with Sequential Access Restrictions." In 2016 6th International Conference on Mechatronics, Computer and Education Informationization (MCEI 2016), Atlantis Press, pp. 610-616, 2016.
[17] M. Rahbari and A. Jahed. "A Hybrid Simulated Annealing Algorithm for Travelling Salesman Problem with Three Neighbor Generation Structures". In 10th International Conference of Iranian Operations Research Society (ICIORS 2017), Babolsar, Iran, May. 2017.
[18] L. Wang, R. Cai, M. Lin, and Y. Zhong. "Enhanced list-based simulated annealing algorithm for large-scale traveling salesman problem." IEEE Access, 7, 144366-144380, 2019.
[19] Y. Chunhua, T. Xiaolin, Z. Xiaojun, and G. Weihua. "State transition algorithm for traveling salesman problem." In Proceedings of the 31st Chinese Control Conference, IEEE, pp. 2481-2485, 2012.
[20] E. Osaba, J. Del Ser, A. Sadollah, M. N. Bilbao, and D. Camacho. "A discrete water cycle algorithm for solving the symmetric and asymmetric traveling salesman problem." Applied Soft Computing, 71, 277-290, 2018.
[21] G. Reinelt, "TSPLIB", 19 February 1997. Available: http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html [Accessed on 18 January 2021]
[22] P. H. Siqueira, S. Scheer, and M. T. A. Steiner. "A recurrent neural network to traveling salesman problem." Travelling Salesman Problem, I-Tech Veinna, pp. 135-156, 2008.

DOI: https://doi.org/10.29207/resti.v5i6.3549