OPEN ACCESS

Saudi Journal of Engineering and Technology
Abbreviated Key Title: Saudi J Eng Technol
ISSN 2415-6272 (Print) |ISSN 2415-6264 (Online)
Scholars Middle East Publishers, Dubai, United Arab Emirates
Journal homepage: https://saudijournals.com

Original Research Article

Simulated Annealing Optimization Algorithm with Self-Escape Mechanism for Travelling Salesman Problem

Md. Azizur Rahman[1*], Mst Jannatun Nesa Mim[1], Sinthia Afrin[1], Ariful Islam[1], Raisa Ahmed[1]

[1]Mathematics Discipline, Science, Engineering and Technology School, Khulna University, Khulna-9208, Bangladesh

DOI: https://doi.org/10.36348/sjet.2025.v10i05.003 | **Received:** 10.04.2025 | **Accepted:** 16.05.2025 | **Published:** 21.05.2025

***Corresponding author:** Md. Azizur Rahman
Mathematics Discipline, Science, Engineering and Technology School, Khulna University, Khulna-9208, Bangladesh
*Email: mdazizur@math.ku.ac.bd

Abstract

The Traveling Salesman Problem (TSP) is a well-known combinatorial optimization problem with significant applications in logistics, transportation, and network design. Efficiently solving this problem requires a careful balance between exploration and exploitation while addressing challenges such as premature convergence and stagnation in local optima. To tackle these issues, numerous algorithms from different perspective have been designed and developed. Among them, Simulated Annealing (SA) is a widely used meta-heuristic approach for solving TSP due to its ability to escape local optima and explore a broad solution space. However, conventional SA can still become trapped in local minima, leading to suboptimal solutions. In this paper, we propose an enhanced SA algorithm that incorporates self-escape mechanism to improve the solution quality of the TSP instances. The self-escape mechanism dynamically identifies trapped routes and facilitate better exploration and diversification. Specifically, the self-escape mechanism introduces a local search refinement process, allowing solutions to effectively escape local optima. Simulation results on benchmark TSP instances demonstrate that the proposed algorithm outperforms conventional SA in terms of solution accuracy. The findings suggest that self-escape mechanism can significantly enhance the effectiveness of SA by preventing premature convergence in complex optimization problems.

Keywords: Combinatorial Optimization, Traveling Salesman Problem, Simulated Annealing Algorithm, Self-Escape Mechanism, Local Search.

1. INTRODUCTION

Optimization in mathematics is the process of identifying the best possible solution from a set of alternatives, according to a specific objective or criterion. One important subfield is combinatorial optimization, which focuses on finding the optimal solution from a finite set of discrete choices, often subject to various constraints. Combinatorial optimization plays a crucial role in many real-world applications, such as network design, scheduling, logistics, and resource allocation [1]. It also underpins the solution of numerous classical problems-most notably, the Traveling Salesman Problem (TSP). The TSP continues to challenge researchers in both theoretical and practical domains due to its complexity. Despite being easy to understand, it exemplifies all the key features of a combinatorial optimization problem and is notoriously difficult to solve efficiently.

The TSP was first mathematically formulated in the 19th century by the Irish mathematician William Rowan Hamilton and the British mathematician Thomas Kirkman [2]. It is a well-known combinatorial optimization problem that involves finding the shortest possible route through a set of cities, where each city must be visited exactly once and the salesperson must return to the starting point. The goal is to minimize both the total travel cost and the overall distance traveled. The problem is typically represented as a sequence of cities, each labeled with a unique positive integer.

The TSP can be modeled as an undirected, weighted graph in which cities correspond to the graph's vertices, paths between cities are represented as edges,

and the distances between cities are assigned as edge weights. The goal is to find a minimum-cost tour that starts and ends at a specified vertex while visiting every other vertex exactly once. In most formulations, the graph is assumed to be complete - meaning that an edge exists between every pair of vertices. If a direct path between two cities does not exist, an arbitrarily long edge can be added to preserve completeness. This does not affect the optimal solution, as such a high-cost edge would never be chosen in the optimal tour. Formally, in graph-theoretic terms, the TSP is defined on a complete, undirected graph $G = (V, E)$, where $V = \{1, 2, 3, \cdots, n\}$ represents the set of vertices (cities or nodes), and $E = \{(i, j): i, j \in V, i < j\}$ denotes the set of edges. A weight matrix $C = (c_{ij})$ is defined over the set E, where each element c_{ij} represents the distance between city i and city j, based on their coordinates (x_i, y_i) and (x_j, y_j), respectively. This distance is typically calculated using the Euclidean formula:

$$c_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \quad (1)$$

Given a list of n cities labeled as $0,1,\cdots,n-1$, and the pairwise distances c_{ij}, the objective function of the TSP, denoted as $Z_{\min}$, can be modeled as follows [4-5]:

$$Z_{\min} = \min \sum_i \sum_j c_{ij} y_{ij} \quad (2)$$

where y_{ij} is a binary decision variable that equals 1 if the edge from city i to city j is included in the route, and 0 otherwise:

$$y_{ij} = \begin{cases} 1 & \text{if city } j \text{ is visited immediately after city } i \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

To guarantee that the solution generates a feasible TSP route, the following constraints must be imposed:

(i) Go-to constraints- Salesman must visit exactly one other city after leaving each city, i.e.,

$$\sum_j y_{ij} = 1, \text{ for all } i = 0,1,\cdots,n-1 \quad (4)$$

(ii) Come-from constraints- Each city must be entered from exactly one other city, i.e.,

$$\sum_i y_{ij} = 1, \text{ for all } j = 0,1,\cdots,n-1 \quad (5)$$

(iii) Subtour elimination constraints- To prevent the formation of disconnected subtours (loops that do not include all cities), i.e.,

$$\sum_{i \in S} \sum_{j \in S} y_{ij} \leq |S| - 1, \text{ for all } S \subset V \text{ where } 2 \leq |S| \leq n-2 \quad (6)$$

where S is any subset of vertices (cities) in the graph G.

The TSP is a classic example of a discrete optimization problem and has been proven to be NP-hard. This means that there is no known polynomial-time algorithm that can solve all instances of the problem unless *P=NP* [3]. Due to its applicability and computational complexity, the TSP has become a central focus in both theoretical computer science and operations research. Despite its simple formulation, finding an exact solution for large instances is computationally expensive. Consequently, a wide range of heuristic, meta-heuristic and approximation algorithms have been developed over the past decade to obtain near-optimal solutions for practical applications. Among these, the Simulated Annealing (SA) algorithm is one of the most well-known and widely used meta-heuristic for solving the TSP. It is easy to understand and simple to implement, and it uses probabilistic acceptance of slightly worse solutions to escape local optima. However, the algorithm often suffers from premature convergence to a local optimum. Once trapped, it may fail to escape and can yield suboptimal results.

To mitigate this issue, we incorporate a self-escape strategy into the optimization process of the SA algorithm when applied to solving the TSP. This mechanism enhances the algorithm's ability to maintain solution diversity, enabling the search to break free from local minima and explore more promising regions of the solution space. The self-escape strategy is inspired by the adaptive behaviors observed in natural organisms, which often instinctively "jump out" of suboptimal conditions to seek better alternatives. Similarly, our approach empowers the algorithm to detect stagnation and respond dynamically. Specifically, the self-escape mechanism performs two critical functions: first, it identifies when the search is stuck in a local optimum; and second, it actively perturbs the current solution to escape the local trap and generate new, potentially better candidates. This dynamic intervention fosters broader exploration and improves the likelihood of reaching a near-global optimum.

The remainder of this paper is structured as follows. Section 2 provides an overview of some related algorithms used to solve the TSP problem. Section 3 provides an in-depth discussion of the proposed Simulated Annealing (SA) algorithm with a self-escape mechanism (SA+SE). Section 4 presents simulation results, analyzes performance, and compares the standard SA with the proposed SA+SE approach. Finally, Section 5 concludes the paper with a summary of the key findings.

2. Related Works

In this section, we review some studies that analyze the optimization procedures of the simulated annealing algorithm, focusing on various strategies used to solve the Traveling Salesman Problem (TSP).

Zhan *et al*., enhanced the performance of the Simulated Annealing (SA) algorithm by introducing a list-based cooling schedule (LBSA), where a predefined list of temperatures guides the acceptance of candidate solutions [6]. The method demonstrates robustness across various parameter settings and outperforms several state-of-the-art algorithms. To validate its effectiveness, they compared LBSA's temperature and search dynamics with classical geometric and arithmetic schedules. However, for large-scale problems, excessively long temperature lists significantly degrade

performance. Wang *et al*., proposed an enhanced LBSA algorithm for solving large-scale TSP instances [7]. The improved approach enables more effective sampling at suitable temperatures, leading to the formation of more promising neighborhoods. A heuristic-augmented sampling strategy is employed to increase the likelihood of generating high-quality neighbors. Extensive experiments on a wide range of large-scale problems demonstrate the effectiveness of this strategy.

Rahman and Parvez proposed a hybrid heuristic algorithm called RNN+SA, which combines Simulated Annealing (SA) with the Repetitive Nearest Neighbor (RNN) algorithm to solve the symmetric TSP [8]. In this approach, a set of feasible routes is first generated using the RNN algorithm, and these routes are then iteratively improved through the SA mechanism. They tested their method on benchmark symmetric TSP instances and demonstrated that RNN+SA outperforms both the basic RNN and SA algorithms, as well as several other standard algorithms. Wu *et al*., proposed a hybrid approach called the adaptive two-stage ant colony SA Algorithm for solving the TSP [9]. In this method, ant colony optimization (ACO) is used as a search strategy within the SA framework, and two adaptive search stages are designed based on the characteristics of the SA algorithm. Their approach improved both convergence speed and solution quality compared to traditional algorithms and demonstrated certain advantages over other standard methods.

Wang *et al*., focused on designing a two-stage Simulated Annealing (SA) algorithm for solving the TSP [10]. A basic SA approach is used in the first stage to generate feasible solutions or complete tours, while a more advanced SA method is applied in the second stage to refine and improve the quality of these solutions. Experimental results showed that this two-stage approach produces better solutions compared to four other recent algorithms, including ACO, self-organizing maps, and particle swarm optimization. Bayram *et al*., enhanced the performance of the Simulated Annealing (SA) algorithm by incorporating two novel neighborhood search mechanisms [11]. Their enhanced approach was compared to the traditional SA algorithm that uses the standard swap neighborhood strategy. Simulation results demonstrated that the modified SA, with its new neighborhood strategies, outperformed the basic version in both solution quality and computational efficiency.

Ezugwu *et al*., developed a hybrid optimization algorithm called SOS-SA, which leverages the complementary strengths of Symbiotic Organism Search (SOS) and Simulated Annealing (SA) [12]. The algorithm begins by generating initial feasible solutions using the standard SA method. These solutions are then further refined through the three adaptive optimization phases of the SOS algorithm. Simulation results indicated that SOS-SA effectively solves the TSP, delivering high-quality solutions and demonstrating competitive performance against other leading optimization algorithms.

3. Proposed Algorithm

In this section, we describe the optimization techniques proposed in this study for solving the Traveling Salesman Problem (TSP). Specifically, we enhance the performance of the classical Simulated Annealing (SA) algorithm by incorporating a self-escape mechanism into the optimization process. The SA algorithm with the self-escape mechanism (SA+SE) is discussed in detail in the following.

The SA is a probabilistic technique proposed by Kirkpatrick *et al*., in 1983 and later by Černý in 1985 [13]. The algorithm is inspired by the annealing process in materials science, where a material is heated to a high temperature and then gradually cooled to reduce internal stresses and increase ductility. In the context of the TSP, the core idea of SA is to occasionally accept moves that worsen the solution in order to escape local minima. The SA is particularly useful for solving challenging computational optimization problems like the TSP, where exact algorithms often fail. However, it often converges prematurely to a solution, causing the process to become trapped in a local optimum.

To address this issue, we incorporate a self-escape strategy into the SA process to avoid premature convergence. By applying the self-escape mechanism, the diversity of solutions is enhanced, allowing the current best solution to deviate from local minima during the computation process. This strategy is inspired by mutual interactions among natural organisms: when trapped in local minima, organisms instinctively “jump out” to explore better solutions. Specifically, the self-escape mechanism accomplishes two key tasks: first, detecting whether a solution is trapped in a local optimum; and second, facilitating its escape to develop a new solution. This concept was first introduced by Wang *et al*., in 2007 within the Particle Swarm Optimization (PSO) algorithm [14]. Later, Wang *et al*., applied it again in 2019 to improve the performance of the discrete symbiotic organisms’ search (DSOS) algorithm [15]. Recently, Rahman *et al*., adopted this mechanism to enhance the probe-based search algorithm for the TSP [16]. The proposed SA+SE algorithm is outlined step by step below, and its pseudocode is presented in Algorithm 1:

Step 1*:* The algorithm begins by randomly generating an initial set of feasible solutions to the TSP, denoted as $R = \{r_1, r_2, \cdots, r_n\}$. The fitness value of each route in this set is then evaluated. The fitness of a route is defined as the inverse of the total distance it covers - meaning that shorter routes are considered more fit, and longer routes less so. If $f(r_i)$ and $d(r_i)$ represent the fitness

and the distance of a given route r_i, respectively, then the relationship can be expressed as follows [16]:

$$f(r_i) = \frac{1}{d(r_i)} \quad (7)$$

Step 2: The new candidate solution, denoted as r_i', is generated and its fitness value, denoted as $f(r_i')$, is measured based on the initial solutions and the specified neighborhood structure such as swap, insertion and reversion.

Step 3: The new solution is accepted or rejected based on a transition probability ρ, defined as follows [13]:

$$\rho = \begin{cases} 1 \; if \; f(r_i') \le f(r_i) \\ e^{-\frac{\Delta f}{T}}, \text{otherwise} \end{cases} \quad (8)$$

Where $T > 0$ is the temperature parameter, and $f(r_i)$ and $f(r_i')$ represent the fitness values of the current route and the new route, respectively. If $\Delta f = f(r_i') - f(r_i) \le 0$, then the new solution r_i' is accepted unconditionally. If $\Delta f > 0$, the current solution r_i is replaced with r_i' based on the probability ρ defined in equation (8). The probability of accepting a worse solution decreases as the temperature T decreases, according to the following limit [13]:

$$\lim_{T \to 0} e^{(-\frac{\Delta f}{T})} = 0 \quad (9)$$

Additionally, a cooling schedule is used to gradually decrease the temperature during the process [13]:

$$T_{k+1=}\gamma T_k \quad (10)$$

where γ is the cooling coefficient or temperature reduction rate. Typically, γ is a value between 0 and 1.

Step 4: Identify the best solution from the updated set of solutions, and evaluate potentially trapped solutions using the following two formulas [14–16]:

$$|\Gamma_l| < \frac{1}{n}\sum_{l=1}^{n} |\Gamma_l| \quad (11)$$

$$\Gamma_l = e(r_i) \cap e(r_{best}) \quad (12)$$

Where $r_i \; (i = 1, 2, 3, \cdots, n)$ represents a complete TSP route, and r_{best} denotes the current best TSP route in the set R. The sets $e(r_i)$ and $e(r_{best})$ denote the edges of the route r_i and r_{best}, respectively. Γ_l denotes the set of common edges between $e(r_i)$ and $e(r_{best})$, $|\Gamma_l|$ is the number of these common edges. The value n represents the total number of complete TSP routes in the set R. A route r_i is considered trapped in a local optimum if and only if inequality in Equation (11) is satisfied.

Step 5: After identifying the trapped routes, they are further improved using a randomized 3-opt process.

Step 6: Repeat steps 2–5 until the stopping condition is met, typically defined by a maximum number of iterations.

Algorithm 1: Pseudocode of the Proposed Simulated Annealing algorithm with self-escape mechanism (SA+SE)

Input: Data matrix (Q), Initial set point temperature (T_0), Temperature decrease rate (γ), Maximum number of iterations (Maxit)

Output: Best feasible TSP route or solution

1: Randomly generate n feasible TSP routes and the set of routes is denoted by $R = \{r_1, r_2, \cdots, r_n\}$

2: Evaluate the Euclidean distance (fitness value) for every route, and denote the collection of fitness values as $f(R) = \{f(r_1), f(r_2), \cdots, f(r_n)\}$

3: For every iteration, $i = 1$ to Maxit

3: For every TSP route r_j, $j = 1$ to n

4: Generate a new route, denoted by $r_i^{'}$, by applying neighborhood structures such as reversion, swap, and inversion

5: Calculate the fitness value for the new route using Equation (7) and designate it as $f(r_i^{'})$

6: Calculate Δf based on Step-3

7: If $\Delta f \le 0$, then replace the current route r_i by assigning $r_i \leftarrow r_i^{'}$

8: If $\Delta f > 0$, calculate the acceptance probability, ρ, using Equation (8). If $\rho \ge u$, replace the current route r_i by assigning $r_i \leftarrow r_i^{'}$, where u is the random value between 0 and 1

9: Check whether the current route r_i is trapped, based on Equation (11)

10. While improvement is found

11. Generate a new route, denoted by $r_i^{'}$, by applying randomize 3-opt.

12. Calculate the fitness value for the new route using Equation (7) and designate it as $f(r_i^{'})$

13. If $f(r_i^{'}) \le f(r_i)$, then update the current route r_i by assigning $r_i \leftarrow r_i^{'}$

14. End while

15: End for

16: Decrease the temperature on the basis of the Equation (10)

17: Update the best solution found so far

18: End for

4. SIMULATION RESULTS AND DISCUSSION

In this section, we present our simulation results for 18 benchmark TSP datasets using the Simulated Annealing (SA) algorithm, both with and without the self-escape (SE) mechanism. The benchmark datasets are sourced from TSPLIB [17], a publicly available online library for TSP data. The datasets range from small instances, such as "elion50," to larger ones like "rd400," demonstrating the algorithm's performance across varying levels of complexity. This library also provides the best-known optimal solutions (BKS) for each dataset. Both the classical SA and the proposed SA+SE algorithms were executed independently five times under identical parameter settings and computing environments. From these 5 runs, we calculated the average, best, worst, and standard deviation (SD) of the results. The algorithms are implemented in MATLAB R2024b and executed on a standard desktop computer with a 12th Gen Intel(R) Core (TM) i7-12700K CPU (3.60 GHz), 16 GB RAM, and the Windows 11 Pro operating system. The maximum number of iterations was set to 1000 for both algorithms. For each run, the best solution obtained over the 1000 iterations was recorded for each TSP dataset. Table 1 summarizes the parameter settings used for both the classical SA and the proposed SA+SE algorithms.

Table 1: List of parameters and their values for both classical SA and proposed SA+SE algorithms.

Serial Number	Name of the Parameter	Used Value of Parameter
1	Number of Populations	*n*
2	Number of Run	5
3	Cooling Coefficient	0.99
4	Reversion Occurrence Probability	0.5
5	Number of Moves	*n*
6	Number of Maximum Iteration	1000
7	Swap Occurrence Probability	0.2
8	Initial Setting Temperature	0.025
9	Insertion Occurrence Probability	0.3

The simulation results obtained from both the classical SA and the proposed SA+SE algorithms are presented in Table 2. The first column lists the serial numbers of the datasets, while the second column provides the names of the datasets. The number following the dataset name indicates its dimension—that is, the number of cities in the TSP instance. For example, "berlin52" refers to a dataset representing 52 locations in Berlin, where 52 denotes the number of nodes. The dataset contains distance information between these 52 nodes. The objective is to determine the shortest possible route that visits each point exactly once and returns to the starting location. The third column shows the best-known solution (BKS) for each dataset, as provided by the TSPLIB database [17]. The fourth through seventh columns display the best solution, average solution, worst solution, and standard deviation (SD) obtained using the classical SA algorithm. The final four columns present the corresponding results for the proposed SA+SE algorithm.

Table 2 clearly shows that incorporating the SE mechanism significantly enhances the performance of the SA algorithm in nearly all considered cases. For example, in the "kroA100" dataset, the best route length improves from 21,418.595 (without SE) to 21,322.737 (with SE). Similarly, the average route length decreases from 21,683.805 to 21,467.501, the worst route length drops from 22,004.847 to 21,577.087, and the standard deviation (SD) is reduced from 213.75404 to 114.32256 - indicating more stable and consistent results. This trend is observed across nearly all datasets: the inclusion of the SE mechanism consistently leads to shorter route lengths and reduced variability.

Table 2 also shows that the averages of the best solutions, average solutions, worst solutions, and standard deviation (SD) across all 18 TSP instances are 28,281.6289, 28,519.2522, 28,837.4971, and 248.4176, respectively, for the proposed SA+SE algorithm. These values are consistently better than those of the classical SA algorithm, which are 28,294.6969, 28,737.8429, 29,416.1777, and 467.5426, respectively. This indicates that the performance of the proposed SA+SE algorithm is better than the basic SA algorithm.

From the above analysis, it can be concluded that the integration of the SE mechanism enhances both the quality and consistency of the SA algorithm's solutions for the TSP problem. Specifically, the SE mechanism increases the algorithm's ability to escape local optima and explore more promising regions of the solution space, resulting in consistently better performance across all instances with lower complexity. Therefore, incorporating the SE mechanism proves to be a valuable enhancement to the SA algorithm for solving TSP instances.

Table 2: Simulation results and comparison between the classical SA algorithm and the proposed SA+SE algorithm

S/N	Datasets	BKS	Performance of Basic SA				Performance of Proposed SA+SE			
			Best	Average	Worst	SD	Best	Average	Worst	SD
1	eilon50	425	428.5138	**430.5008**	**434.3380**	**2.2889**	**427.9651**	431.7084	435.6781	3.1821
2	eil51	426	428.98165	432.87246	440.06047	4.556948	**428.87176**	**431.88112**	**436.97081**	**3.8827033**
3	berlin52	7542	7544.3659	7745.8856	7969.9401	209.5317	7544.3659	**7625.4232**	**7777.3323**	**113.04436**
4	st70	675	682.57482	688.26798	692.80452	4.149435	682.57482	**686.53464**	**692.19676**	**3.7140812**
5	eilon75	535	544.1830	551.1607	555.4610	4.3816	**543.5472**	**549.9307**	**555.3704**	**4.2027**
6	eil76	538	**548.72959**	556.16424	560.85953	4.584221	551.34140	**554.68980**	**559.25252**	**3.0611156**
7	pr76	108159	108159.44	109991.32	112843.39	1759.260	108159.44	**109010.25**	**110358.04**	**1166.1222**
8	rd100	7910	8035.4466	**8092.8462**	**8157.5000**	**48.71723**	**8011.4672**	8187.8204	8383.0728	159.39877
9	kroA100	21282	21418.595	21683.805	22004.847	213.7540	**21322.737**	**21467.501**	**21577.087**	**114.32256**
10	kroC100	20749	20813.270	21134.819	21582.746	333.4157	20881.614	**21086.485**	**21327.968**	**203.93263**
11	kroE100	22068	**22068.759**	22275.288	22836.800	320.1160	22073.249	**22209.155**	**22334.681**	**114.56562**
12	lin105	14379	14420.101	14654.826	**14876.205**	**209.4868**	**14382.996**	**14652.532**	14945.750	244.58374
13	pr107	44303	44301.684	44522.738	44762.432	167.7416	44301.684	**44373.328**	**44436.741**	**68.067025**
14	pr144	58537	**58587.142**	59517.269	61032.524	1070.115	59111.032	**59506.645**	**60719.228**	**680.81501**
15	pr152	73682	74438.938	**75236.004**	**76348.414**	997.7797	**74292.173**	75326.213	**76600.347**	**962.37339**
16	kroA200	29368	29662.317	30286.748	30690.395	412.6708	**29495.052**	**29983.025**	**30207.555**	**293.5315**
17	pr226	80369	81273.974	83304.522	87326.581	2481.469	**80847.420**	**81134.874**	**81496.356**	**235.92862**
18	rd400	15281	**15947.528**	16176.136	16375.900	171.747	16011.790	**16128.543**	**16231.322**	**96.7880**
Average			28294.69	28737.842	29416.1777	467.5426	**28281.6289**	**28519.2522**	**28837.49715**	**248.4176**

5. CONCLUSION

We have enhanced the performance of the classical Simulated Annealing (SA) algorithm for solving the Travelling Salesman Problem (TSP). In the improved version, we adopt a self-escape (SE) mechanism into the SA process to promote diversification and more effective exploration of the search space. This integration address key challenges of the standard SA optimization algorithm, including premature convergence to local optima and the imbalance between exploration and exploitation. The SE mechanism dynamically identifies worse solutions during the search process and enhances diversification by maintaining a more varied population. As a results the algorithm can effectively escape from local optima. Simulation results on 18 benchmark TSP datasets demonstrate that the enhanced SA consistently outperforms the baseline version in terms of best, average, and worst solutions, while also exhibiting lower variability.

In the future, the SE mechanism could be integrated with other meta-heuristic algorithms, such as Ant Colony Optimization, Genetic Algorithms, and others, to further improve solution quality. Additionally, the proposed SA+SE approach could be applied to other optimization problems - such as delivery route planning and job scheduling - to demonstrate its versatility. Its effectiveness may also be further improved by applying it to larger real-world problems and leveraging faster computing techniques.

Conflict of Interest: The authors confirm that there are no conflicts of interest related to the publication of this article.

REFERENCES

1. Papadimitriou, C. H., & Steiglitz, K. (1998). Combinatorial Optimization: Algorithms and Complexity. Dover Publications.
2. Travelling Salesman Problem. (2024). In Wikipedia. Retrieved from https://en.wikipedia.org/wiki/Travelling_salesman_problem
3. Garey, M. R., & Johnson, D. S. (2002). *Computers and intractability: Vol. 29*. New York: W. H. Freeman.
4. Matai, R., Singh, S.P., and Mittal, M. L. (2010). Traveling salesman problem: an overview of applications, formulations, and solution approaches. Traveling Salesman Problem, *Theory and Applications, 1(1)*, pp.1- 25. https://doi.org/10.5772/12909
5. Rahman, M. Z., Sheikh, S. R., Islam, A., & Rahman, M. A. (2024). Improvement of the Nearest Neighbor Heuristic Search Algorithm for Traveling Salesman Problem. *Journal of Engineering Advancements, 5(01)*, 19–26. https://doi.org/10.38032/jea.2024.01.004
6. Zhan, S. H., Lin, J., Zhang, Z. J., & Zhong, Y. W. (2016). List-Based Simulated Annealing Algorithm for Traveling Salesman Problem. *Computational*

intelligence and neuroscience, 2016(1), 1712630. https://doi.org/10.1155/2016/1712630

7. Wang, L., Cai, R., Lin, M., & Zhong, Y. (2019). Enhanced list-based simulated annealing algorithm for large-scale traveling salesman problem. *IEEE Access*, *7*, 144366-144380. https://doi.org/10.1109/ACCESS.2019.2945570
8. Rahman, M. A., & Parvez, H. (2021). Repetitive nearest neighbor based simulated annealing search optimization algorithm for traveling salesman problem. *Open Access Library Journal*, *8(6)*, 1-17. https://doi.org/10.4236/oalib.1107520
9. Wu, Y., Wang, H., Li, M., Tan, H., Wang, D., & Sheng, M. (2025). The adaptive two-stage ant colony Simulated Annealing Algorithm for solving the Traveling Salesman Problem. *RAIRO-Operations Research*, *59(2)*, 1199-1213. https://doi.org/10.1051/ro/2025031
10. Wang, Z., Geng, X., & Shao, Z. (2009). An effective simulated annealing algorithm for solving the traveling salesman problem. *Journal of Computational and Theoretical Nanoscience*, *6*(7), 1680-1686. https://doi.org/10.1166/jctn.2009.1230
11. Bayram, H., & Şahin, R. (2013). A new simulated annealing approach for travelling salesman problem. *Mathematical and computational Applications*, *18*(3), 313-322.
12. 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*, 189-210. https://doi.org/10.1016/j.eswa.2017.01.053
13. Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science, 220(4598)*, 671-680. https://doi.org/10.1126/science.220.4598.671
14. Wang, W., Liu, G., & Wen, W. (2007). Study of a self-escape hybrid discrete particle swarm optimization for TSP. *Computer Science*, *34*, 143-145.
15. Wang, Y., Wu, Y. W., & Xu, N. (2019). Discrete symbiotic organism search with excellence coefficients and self-escape for traveling salesman problem. *Computers & Industrial Engineering*, *131*, 269-281. https://doi.org/10.1016/j.cie.2019.04.008
16. Rahman, M. A., & Ma, J. (2024). Two-Stage Probe-Based Search Optimization Algorithm for the Traveling Salesman Problems. *Mathematics*, *12(9)*, 1340. https://doi.org/10.3390/math12091340
17. TSPLIB. http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/