Bitlis Eren Üniversitesi Fen Bilimleri Dergisi
BİTLİS EREN UNIVERSITY JOURNAL OF SCIENCE
ISSN: 2147-3129/e-ISSN: 2147-3188
VOLUME: 13 NO: 1 PAGE: 216-231 YEAR: 2024
DOI:10.17798/bitlisfen.1380086

Comparison of New and Old Optimization Algorithms for Traveling Salesman Problem on Small, Medium, and Large-scale Benchmark Instances

Md Al Amin HOSSAIN[1*], Züleyha YILMAZ ACAR[2]

[1]Selçuk University, Graduate School of Natural and Applied Sciences, Department of Information Technology Engineering
[2]Selçuk University, Faculty of Technology, Department of Computer Engineering
(ORCID: 0000-0003-3382-5300) (ORCID: 0000-0002-4488-478X)

Keywords: Benchmark, Metaheuristic, Optimization, Travelling Salesman Problem.

Abstract
The Traveling Salesman Problem (TSP), a prominent combinatorial optimization issue, is the subject of this study's evaluation of the performance of new and old optimization techniques. This paper seeks to expand knowledge of optimization techniques and how they might be applied to solve TSP challenges. The goal of the research is to compare various algorithms' scalability, convergence, and computation times on benchmark instances of several sizes. To achieve this goal, this paper carried out extensive testing using the Artificial Bee Colony, Grey Wolf Optimization, and Salp Swarm Algorithm as new optimization algorithms and the Genetic Algorithm, Ant Colony Optimization, and Simulated Annealing as old optimization algorithms. In small, medium, and large-scale benchmark cases, these algorithms were examined. The findings of this investigation show that the new optimization techniques are more convergent and scalable than the old ones, especially for medium-scale scenarios. They perform better in terms of solution quality by applying objective function values. The new methods also exhibit improved scalability, successfully adjusting to medium-scale instances. However, there were no discernible changes between the smaller and larger instances. This study makes an impact by offering insightful information about how well optimization methods perform while solving the TSP. Each algorithm's strengths and downsides have been reported, and these details offer useful guidance for choosing an algorithm for a certain scenario. The results also show the practical ramifications of applying novel optimization techniques, especially in medium-scale instances.

1. Introduction

The study of optimization algorithms has grown significantly in relevance for their effective and economical solutions to challenging issues. The Traveling Salesman Problem (TSP) stands out among these issues because of its numerous real-world applications across multiple sectors [1]. The TSP is a prevalent issue in operations research and computational science that requires figuring out the quickest path a traveling salesperson can take to visit a certain list of cities, specifically once, and then head back to the beginning point [2]. Even though its formulation appears easy, the TSP becomes more difficult as the number of cities increases exponentially, creating a combinatorial eruption of alternative solutions [3]. Optimization algorithms provide promising avenues for tackling the inherent complexity of the TSP. These algorithms utilize advanced search and optimization techniques to explore the solution space efficiently and find high-quality solutions within a reasonable timeframe [3]. By intelligently navigating through the vast solution space, optimization algorithms can discover near-

[*]Corresponding author: *alamin.ict.iu@gmail.com*

Received: 23.10.2023, Accepted: 29.12.2023

optimal or even optimal solutions, resulting in minimized travel distances, optimized resource allocation, and improved overall efficiency in real-world scenarios [4].

The primary objective of this research is to comprehensively investigate and evaluate the performance of different optimization algorithms for solving the TSP. Specifically, we aim to compare the effectiveness of well-established algorithms such as Genetic Algorithms (GA), Ant Colony Optimization (ABC), and Simulated Annealing (SA) as old optimization algorithms and Artificial Bee Colony (ABC), Salp Swarm Algorithm (SSA), and Grey Wolf Optimization (GWO) as new optimization algorithms in finding optimal or near-optimal solutions to the TSP. By assessing their performance in terms of solution quality, convergence, and computational efficiency, we seek to provide valuable insights into the strengths and limitations of these algorithms in addressing the TSP's complexities.

This study's contribution lies in its rigorous evaluation and comparison of multiple optimization algorithms for the TSP. While previous studies have individually examined the performance of these algorithms, this research takes a comprehensive approach by directly comparing their effectiveness side by side. This comparative analysis enables us to identify the algorithm(s) that exhibit superior performance characteristics for solving the TSP, offering valuable guidance for researchers and practitioners in selecting the most suitable algorithm for their specific problem instances. Through a thorough evaluation of their performance in terms of solution quality, convergence, and computational efficiency, this analysis strives to provide comprehensive insights into the strengths and limitations of each algorithm, thus facilitating informed decision-making for future research and practical applications. By shedding light on the capabilities and performance trade-offs of different optimization algorithms for the TSP, this study endeavors to advance the understanding and utilization of these algorithms in solving real-world optimization problems.

The following is the format of the study's accomplishing sections: The literature on TSP is delved into in Section 2. Section 3 provides a summary of the approaches used to address the relevant TSP issues, with an emphasis on optimization algorithms and metaheuristic techniques. The experimental results and subsequent discussions are outlined in Section 4. Conclusions from the study's findings are in Section 5.

2. Literature Review

The TSP is a renowned stochastic optimization concern that seeks the fastest route to travel between a starting point and a number of cities. Numerous optimization techniques have been created over time to effectively solve the TSP. The usefulness and performance of several optimization algorithms for the TSP have been thoroughly evaluated through prior comparative studies. The available literature is reviewed in this section, and the conclusions of these comparative investigations are outlined. The study undertaken by Şahin [4] presents new relocation and city selection functions that are incorporated into the bees algorithm to improve its TSP solution efficiency. This work demonstrates the efficiency of the approach in optimizing TSP solutions by notably increasing solution reliability, especially in cases with higher city counts. Li et al. [5] proposed the TSP solution using a differential edge information-based ACO. Heterogeneous population automation, tour creation, and smooth search operators improve candidate solution quality in the algorithm. The algorithm outperformed state-of-the-art algorithms on TSPLIB benchmark examples for mid-scale and small-scale TSP instances. It surpassed the other methods, proving its efficacy. Middle-scale TSP situations need faster solving. Ajayi et al. [6] sought TSP optimization methods. ACO was compared to Dijkstra's method and particle swarm optimization. The ACO algorithm excelled in route quality and total length, while Dijkstra's method excelled in minimum cost calculations. The research's application to other optimization problems, the need for further parameter exploration, and the lack of comparison with other state-of-the-art TSP algorithms were shortcomings. The algorithm's large-scale TSP scalability needs further study. Mondal and Srivastava [7] used evolutionary algorithms to solve a time-limited TSP with time limits for each city. The technique used cyclic crossover and specific mutation operations. The approach worked with benchmark instances in computations. The authors advised future research on multiple TSPs, probabilistic TSPs with fuzzy parameters, cost-limited TSPs, and bi-objective TSPs in a fuzzy environment, as well as travel time and asymmetric cases.

Hasan [8] compared the Bat Algorithm (BA) and ABC for solving the TSP. ABC found the best tour faster than BA, but BA needed more parameters and a better control method. The conclusion suggests future research into combining both algorithms in a system. Khajehzadeh et al. [9] introduced an Adaptive Salp Swarm Algorithm (ASSA) for geotechnical structure optimization. A new leader and follower position updating equations improved exploration and prevented premature convergence. ASSA outperformed other optimization methods in benchmark tests. ASSA optimized geotechnical constructions while satisfying geotechnical and structural limit states, yielding cost-effective designs with superior results than competing methods. Liu et al. [10] tackled complex optimization challenges involving multiple objectives and modes within the TSP. A test problem generator designed for problems involving multiple depots and multiple traveling salesmen simultaneously creates instances that exhibit known Pareto optimal solutions. This tool with three or more objectives, test problem generators for specific features, and crossover operators that investigate a wider solution space are future research possibilities. Khan et al. [11] provide Modified GWO to solve the multiple objectives and covering constraints for the salesman problem. K-bit exchange, K-opt, and non-dominated sorting-based GA are incorporated to improve search and solution quality. Their proposed algorithm outperforms other multi-objective optimization algorithms on standard benchmark examples, proving its multi-objective-based problem-solving efficiency. The suggested approach can only handle this covering problem with crisp data sets and needs adjustments to accommodate imprecise data sets.

Panwar and Deep [12] stated the Discrete Salp Swarm Algorithm (DSSA) as an improved version of the SSA for solving the TSP. Swap, shift, and symmetry operators are used for global exploration and local exploitation, while the 2-opt technique improves local search. DSSA outperformed the GA, ABC, Spider Monkey, Jaya, Black Hole, and Symbiotic Organism Search on 45 TSP instances. The paper advises using DSSA to solve other discrete optimization problems like scheduling and routing issues. Khan and Maiti [13] offer a modified ABC algorithm to solve the TSP in their research paper. Swap sequences and city sequence swap procedures create various solution-updating rules in the algorithm. The suggested method is tested on TSPLIB benchmark TSP problems, showing its accuracy, efficiency, and consistency compared to existing algorithms. With appropriate changes, ABC can handle discrete optimization problems, including TSP, according to the study. The existing literature includes computational evaluations on large-scale TSP scenarios, algorithm behavior analysis under varied issue forms or restrictions, and a defined benchmarking methodology for fair contrasts. These gaps suggest a full comparative study to solve these constraints and better understand TSP optimization algorithms' strengths and drawbacks. The literature review section summarizes TSP optimization techniques, evaluates comparative research, and highlights gaps that require a complete comparative investigation. A brief summary of the explored literature is listed in Table 1.

3. Methodology

This section covers the theoretical basis of TSP as well as how the performance of various optimization algorithms, including GA, ACO, SA, ABC, SSA, and GWO, was compared in this study. Figure 1 displays the methodological approach's structural flowchart.

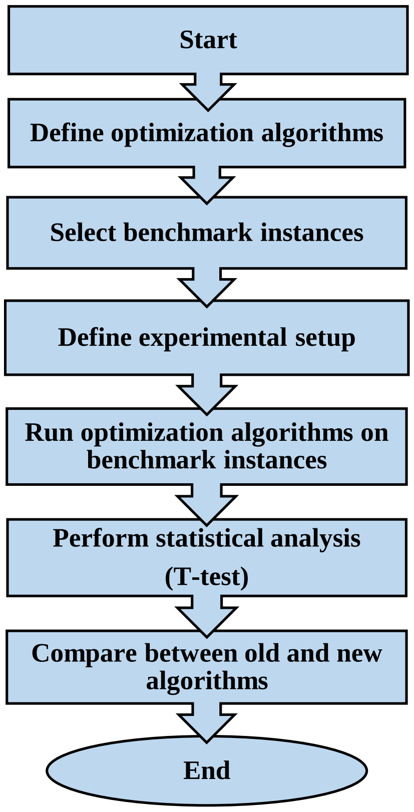

Start
Define optimization algorithms
Select benchmark instances
Define experimental setup
Run optimization algorithms on benchmark instances
Perform statistical analysis (T-test)
Compare between old and new algorithms
End

Figure 1. Structural flowchart of methodology.

3.1. Traveling Salesman Problem (TSP)

The TSP is a prominent optimization problem that includes finding the most efficient way for a salesperson to visit a series of cities and return to the beginning city [14]. The TSP reduces the salesman's route distance and cost [7]. This problem is formally described by creating choice variables to represent direct travel between cities and a distance or cost matrix to show distances between pairs of cities: a collection of n cities, $C = \{c_1, c_2, \dots, c_n\}$, and a distance or cost matrix D, where D_{ij} is the distance between cities c_i and c_j. Some TSP variables and parameters represent :

Table 1. A brief overview of the reviewed literature

#	Author, Year	Used Algorithm	Problem – Average Value	Instance Size
1	Khan and Maiti, 2019 [13]	Swap sequece based ABC	berlin52 – 7543	small
2	Şahin, 2022 [4]	Improved bees algorithm	kroA100 – 21466	small
3	Ajayi et al., 2022 [6]	Particle swarm optimization and ant colony optimization with Dijkastra's algorithm	No specific problem	-
4	Hasan, 2022 [8]	Artificial bee colony and bat algorithm	No specific problem	-
5	Khajehzadeh et al., 2022 [9]	Adaptive salp swarm optimization	Real world problems	-
6	Panwar and Deep, 2022 [12]	Discrete salp swarm optimization	tsp225 – 3940.23	medium
7	Li et.al., 2023 [5]	Heuristic smoothing differential ACO	dsj1000 – 18897396.15	large
8	Mondal and Srivastava, 2023 [7]	cyclic crossover based GA	kro124p – 36612.94	small
9	Liu et al., 2023 [10]	Evolutionary mutltimodal multiobjective algorithm	TSPXEA problems	-
10	Khan et al., 2023 [11]	Decomposition based GWO	–	-

- Choice variables: If the salesman travels directly from city i to city j, $x_{ij} = 1$, $otherwise\ x_{ij} = 0$.
- Parameters: D_{ij} = Cost or distance between cities i and j.
  Mathematically optimize the TSP:
  $$minimize\colon\ Z = \Sigma\Sigma D_{ij} * x_{ij} \tag{1}$$
- Subject to:
  1) For each city i, $\sum x_{ij} = 2$, ensuring that each city is visited precisely once.
  2) $\sum x_{ij} = 2$ for each city j to leave each city precisely once.
  3) The salesman travels to and from each city by setting $\sum x_{ij}$ - $\sum x_{ij} = 0$

Solving the TSP involves finding choice variables x_{ij} that minimize the total distance or cost Z while fulfilling constraints. As the number of cities rises, solving the TSP optimally becomes an extremely difficult issue [3].

3.2. Optimization Algorithms and Benchmark Instances

In this study, a comprehensive comparison of optimization algorithms is conducted. A brief description is provided for each optimization algorithm, focusing on their underlying principles, search strategies, and important parameters that contribute to their performance. The algorithms considered in this research are GA, ACO, SA, ABC, SSA, and GWO, all recognized for their efficacy in solving optimization problems.

I. Genetic Algorithm : Genetics-inspired GA optimizes. It addresses complicated problems like evolution [15]. GA analyzes potential solutions. Selection, crossover, and mutation generate new solutions. Repeat until a termination requirement is reached [7]. GA is utilized in many fields because it can handle complex search spaces and identify optimal or near-optimal answers. The stepwise GA algorithm to solve TSP is illustrated in Algorithm 1.

II. Ant Colony Optimization algorithm: ACO optimizes like ants. Pheromone trails help it find optimal solutions [16]. Ants use probabilistic pheromones and heuristics to solve problems. The quality of solutions determines pheromone trail updates. ACO promotes exploration and delays convergence [5]. It solved optimization difficulties. Algorithm 2 illustrates the stepwise ACO approach to solve TSP.

III. Simulated Annealing algorithm: Metallurgy's annealing process influenced SA's metaheuristic algorithm [17]. It starts with a solution and allows occasional "worse" steps based on a probability distribution to find better ones. Simulating cooling and adopting a worse answer reduces over time. SA helps avoid local optima and explore solution space [18]. Many optimization problems utilize it. Algorithm 3 depicts the stepwise SA algorithm to solve TSP.

Algorithm 1. GA algorithm to solve TSP

```
1: Initialize population p using the initialize_population method.
2: Calculate the fitness for each individual in p using the evaluate_fitness method.
3: Set the iteration counter T = 0.
4: Set max_iterations = 1000.
5: while T < max_iterations do
6: Select parents from the population p using the selection method.
7: Perform crossover on the selected parents with a probability of 0.8 using the crossover method to generate offspring.
8: Perform a mutation on the offspring with a probability of 0.2 using the mutation method.
9: Calculate fitness for each offspring using the evaluate_fitness method.
10: Replace the worst individuals in the population p with the best offspring.
11: Update the best solution found so far.
12: Increment iteration counter T = T + 1
13: End while
14:Return the best solution found (best_individual).
```

Algorithm 2. ACO algorithm to solve TSP

```
1: Initialize the pheromone matrix with initial_pheromone
2: Set best_tour = none and best_distance = infinity.
3: Start timer
4: Repeat num_iterations times:
5:     Generate tours using construct_tour method for num_ants.
6:     For each tour:
7:         Calculate tour distance
8:         If distance is less than best_distance:
9:             Update best_tour and best_distance
10: Update pheromone matrix using the update_pheromone_matrix method.
11: End repeat
12: Stop timer
13: Calculate computation_time as the difference between start_time and end_time.
13: Return best_tour, best_distance, and computation_time.
```

IV. Artificial Bee Colony algorithm: Honeybee browsing inspired ABC, a population-based metaheuristic algorithm [19]. It simulates bee colony intelligence to find optimal solutions. The algorithm uses employed, bystanders, and scout bees. Onlooker bees choose solutions based on fitness, while employed bees move solutions across the search space. Scout bees randomly investigate new areas [20]. ABC solves complicated optimization issues quickly and easily. Algorithm 4 illustrates the stepwise ABC approach to solve TSP.

V. Salp Swarm Algorithm (SSA): The SSA optimizes marine invertebrates, salps, and swarming. Buoyancy guides salps to better solutions when traversing the search space [21]. The SSA has found optimal solutions to several optimization issues [22]. The stepwise SSA algorithm to solve TSP is illustrated in Algorithm 5.

Algorithm 3. SA algorithm to solve TSP

```
1: Initialize Simulated Annealing with cities, initial_temperature, cooling_rate, num_iterations
2: Initialize the distances matrix by calculating the distances between cities.
3: Define the calculate_tour_distance function to calculate the total distance of a tour.
4: Define the generate_neighbor function to generate a neighbor tour by swapping two cities.
5: Define the acceptance_probability function to calculate the acceptance probability based on current_distance, new_distance, and temperature.
6: Initialize current_tour with a random tour.
7: Initialize best_tour and current_distance as current_tour and its distance.
8: Initialize best_distance as current_distance and temperature as initial_temperature.
9: Start the timer T.
10: Repeat num_iterations times:
11: Generate a new tour, new_tour, as a neighbor of current_tour.
12: Calculate the distance of new_tour, new_distance
13:     If acceptance_probability (current_distance, new_distance, temperature) is greater than a random number:
14:         Update current_tour and current_distance as new_tour and new_distance
15:     If current_distance is less than best_distance:
16: Update best_tour and best_distance as current_tour and current_distance.
17: Decrease temperature by multiplying with cooling_rate.
18: End the timer.
19: Calculate computation_time as the difference between start_time and end_time.
20: Return best_tour, best_distance, and computation_time.
```

Algorithm 4. ABC algorithm to solve TSP

```
1: Initialize population p
2: Set iteration counter T = 0.
3: Set max_iterations = 1000
4: while T < max_iterations do
5:    Calculate fitness for each individual in p using $f_i$
6:    Perform an employed bees phase to generate a new population.
          list new_population
          select and swamp two positions (i and j) from the bee's solution.
7:    Perform the onlooker bee phase to select the population based on fitness
          list selected_population
```

```
        Calculate the selection probabilities based on
the fitness values.
8:    Perform the scout bee phase to replace poor
solutions.
        Find the index of the bee with the best fitness
(min distance)
        Find the best fitness value
        list *new_population*
9:    Update the best solution found so far.
        Find the index of the bee with the best fitness
(min distance)
        Update the best solution and best distance if the
current best fitness is better
10:   Increment iteration counter $T = T + 1$
11: End while
12: Return the best solution found.
```

VI. Grey Wolf Optimization algorithm: Grey wolves inspired GWO's metaheuristic algorithm. It simulates wolf leadership and cooperative hunting to find the best solutions. Alpha, beta, delta, and omega wolves signify hierarchy in the algorithm [23]. Wolves position themselves according to the alpha and hunting distance. Exploration and exploitation help GWO uncover optimal solutions [24]. It solved optimization difficulties. Algorithm 6 depicts the stepwise GWO algorithm to solve TSP.

Algorithm 5. SSA algorithm to solve TSP

```
1: Initialize salps population
2: Calculate the distance for each salp in the population
using the given cities.
3: Set *best_salp* as None
4: Set the iteration counter to 0.
5: Set *max_iterations = 1000.*
6: while *iteration_counter* < *max_iterations* do
7:    For each salp in the population, do
8:        If *best_salp* is None or the distance of the
current salp is smaller than the distance of *best_salp,*
then
9:            set *best_salp* as the current salp
10:       Create a new salp by updating the position of
the current salp based on the *best_salp.*
11:   Update the population with the new salps.
12:   Increment the iteration counter by 1.
13: End while
14: Return *best_salp(best_solution)*
```

Algorithm 6. GWO algorithm to solve TSP

```
1: Initialize wolves
2: Set the iteration counter.
3: Set the maximum number of iterations
4: while *iteration_counter* < *max_iterations* do
5:    Evaluate fitness for each wolf
6:    Update the best wolf found so far
7:    Determine alpha, beta, and delta wolves as the
best, second-best, and third-best solutions
8:    For each wolf, do
9:        Update the position of the wolf based on alpha,
beta, and delta.
10:   End for
11:   Increment *iteration_counter*
12: End while
13: Return the best wolf found (*best_solution).*
```

4. Experimental Analysis

This section examines many aspects through a variety of experiments. This includes a close study of how to choose benchmark instances, the specifics of the experimental layout, how convergence works across optimization strategies, and how well statistical analysis works. Additionally, this segment provides a complete discussion of the insights gained from the results as well as a detailed explanation of the outcomes.

To assess the performance of the optimization algorithms, a selection of benchmark instances is made from the TSPLIB library [25]. The chosen instances encompass a range of problem sizes and complexities, ensuring a diverse set of challenges for the algorithms. The benchmark instances include burma14, berlin52, and kroA100 as small-sized instances, while ts225 and att532 represent medium-sized instances. Additionally, rat783 and dsj1000 are selected as large-sized instances. These carefully chosen instances enable a comprehensive analysis of the algorithms' performance across various problem sizes and complexities.

3.1. Experimental Setup

Table 2 presents common and specific parameters for several optimization algorithms, including GA, ACO, SA, ABC, SSA, and GWO. The population size is set to 100 for all algorithms with 1000 iterations. GA uses a crossover rate of 0.8 and a mutation rate of 0.2. ACO employs alpha and beta values of 1.0 and 5.0, respectively. SA starts with an initial temperature of 1000 and a cooling rate of 0.95. ABC has a limit of 5 for employed bee trials. Performance metrics for evaluation include the best optimal solution, average solution, standard deviation, standard deviation (%), and computation time. The algorithms are applied to different sets of city coordinates, including burma14, berlin52, kroA100, ts225, att532, rat783, and dsj1000.

3.2. The Statistical Tests for Comparing Performance

To evaluate the statistical significance of the obtained results, appropriate statistical tests will be employed.

Table 2. The common and specific parameters of optimization algorithms used

Parameter	Algorithm & Value					
	GA	ACO	SA	ABC	SSA	GWO
Population Size	100					
No. of Iterations	1000					
Crossover Rate	0.8					
Mutation Rate	0.2					
Alpha		1.0				
Beta		5.0				
Evaporation Rate		0.5				
Initial Temperature			1000			
Cooling Rate			0.95			
Limit				5		
City Coordinates	burma14, berlin52, kroA100, ts225 and att532, rat783 and dsj1000					
Performance Metrics	best optimal solution, average solution, standard deviation, standard deviation (%), and computation time					

Specifically, t-tests will be utilized to compare the convergence and efficiency of the optimization algorithms. The t-tests enable the determination of whether there exist statistically significant differences in the performance of the algorithms in terms of their convergence to the optimal solution. The t-test is a widely used statistical test that assesses the difference between the means of two groups [26]. In this case, the groups correspond to the different optimization algorithms under investigation. The t-test calculates a t-statistic, which measures the extent of the difference between the means of the two groups relative to the variability within each group (in this case, the algorithms). A larger absolute t-statistic indicates a greater difference between the means. The t-statistic is calculated using the following equation (2):

$$t = \frac{(mean_1 - mean_2)}{\sqrt{(std_1^2 \,/\, n_1)} + \sqrt{(std_2^2 \,/\, n_2)}} \qquad (2)$$

Where $mean_1$ and $mean_2$ are the sample means of the best optimal solution for the two algorithms, std_1 and std_2 are the sample standard deviations, and n_1 and n_2 are the sample sizes. The p-value, which the t-test also presents, denotes the likelihood of detecting the determined difference in means under the presumption that there is no real difference between the groups. A lower p-value indicates greater statistical significance and stronger evidence against the null hypothesis of no difference. The significance level in this study is 0.05, which denotes a 5% chance of detecting significant differences as a result of random oscillation. We deny the null assumption and endorse the alternative hypothesis, determining that the algorithms differ effectively if the computed p-value is smaller than the significance level. This technique evaluates optimization algorithms across benchmark instances. The statistical tests reveal considerable differences in the convergence and effectiveness of the algorithms, revealing their relative performance and applicability for problem instances of varied sizes and complexities.

3.3. Results Assessments

In this paper, all of the suggested algorithms were coded in Python in the Anaconda environment using Matplotlib, Seaborn, SciPy, and the Pandas library, and several estimates were made. Table 3 presents a comprehensive comparison of three new metaheuristic algorithms— ABC, SSA, and GWO — on various instances of the TSP of different sizes. Table 3 includes seven TSP instances ranging from 14 to 1000 nodes, and for each instance, it provides the best optimal solution (best opt.), average solution (avg), standard deviation (std), standard deviation percentage (std%), and time taken by each algorithm to solve the TSP instance.

From Table 3, we can observe that for smaller TSP instances such as burma14 and berlin52, all three algorithms perform relatively well, with ABC and SSA providing the best solutions. The average solution metric reveals the algorithms' overall performance in finding solutions, with ABC consistently delivering low average solutions across all instances. The standard deviation metric highlights the stability of the algorithms' solutions, with ABC and GWO exhibiting low standard deviations.

Table 4 presents a detailed performance comparison of the GA, ACO, and SA algorithms on various instances of the TSP categorized into small, medium, and large-scale instances. For small-scale instances, such as burma14, berlin52, and kroA100, the algorithms demonstrate similar trends. GA achieves the best optimal solutions of 42, 24,515, and 153,460, respectively. ACO and SA also perform well, with ACO achieving optimal solutions of 31, 7,677, and 22,946 and SA achieving optimal solutions of 30, 12,490, and 74,583 for the respective instances. The average solutions for these instances follow a similar pattern, with GA, ACO, and SA showing competitive performance. In the medium-scale instances, ts225 and att532, ACO outperforms the other algorithms. It achieves the best optimal solutions of 133,285 and 99,268, respectively, whereas GA and SA achieve optimal solutions of 1,478,608 and 1,035,202 for ts225 and 1,538,293 and 1,034,340 for att532. The average solutions also show ACO as the top performer. For large-scale instances, rat783 and dsj1000, GA demonstrates its strength by achieving the best optimal solutions of 170,626 and 534,398,427, respectively. ACO and SA lag behind in these instances, with ACO achieving optimal solutions of 10,431 and 22,404,181, and SA achieving optimal solutions of 130,100 and 417,124,012.

In Table 4, we can see that the Std(%) values for all three algorithms are relatively low, ranging from 0.12% to 10.06%. This suggests that the algorithms are relatively robust and can provide consistent solutions across different TSP instances.

In Table 5, the t-test analysis revealed that there were no statistically significant differences between the performance of different optimization algorithms when considering all instances. However, in small

Table 3. Performance comparison of new metaheuristic algorithms on tsp instances of diverse sizes

Algorithm	Performance metrics	Problem instance						
		burma14	berlin52	kroA100	ts225	att532	rat783	dsj1000
ABC	best opt.	66	31671	181460	1649553	1663663	184334	567420224
	avg	57	29331	170553	1585724	1612119	178992	555245806
	std	6	1447	8172	48380	30457	2845	7572149
	std (%)	11.83	4.93	4.79	3.05	1.88	1.58	1.35
	time (s)	4	14	29	63	163	239	317
SSA	best opt.	13	12879	95966	1116522	1087426	139897	475808151
	avg	57	29380	169162	1587326	1613307	179095	554705324
	std	7	1617	9358	43940	34175	2994	8011712
	std (%)	12.72	5.5	5.53	2.76	2.11	1.67	1.44
	time (s)	3	16	29	67	177	277	358
GWO	best opt.	21	16410	116098	1245252	1252724	152429	474605615
	avg	21	16485	116455	1247805	1255686	152592	475494467
	std	1	824	3402	25904	25024	1446	6689991
	std (%)	7.53	4.99	2.92	2.07	1.99	0.94	1.40
	time (s)	7	28	46	105	279	396	511

Table 4. Performance comparison of old metaheuristic algorithms on TSP instances of diverse sizes

Algorithm	Performance metrics	Problem instance						
		burma14	berlin52	kroA100	ts225	att532	rat783	dsj1000
GA	best opt.	42	24515	153460	1478608	1538293	170626	534398427
	average	57	29366	169691	1595693	1613034	179461	556043666
	std	5	1618	7476	44677	33547	2986	7717300
	std (%)	10.06	5.51	4.41	2.79	2.07	1.66	1.38
	time (s)	3	10	22	76	330	688	1200
ACO	best opt.	31	7677	22946	133285	99268	10431	22404181
	average	38	10838	32636	187488	139957	14227	30817615
	std	3	854	2219	11950	4937	395	947631
	std (%)	9.56	7.88	6.79	6.37	3.52	2.77	3.07
	time (s)	68	449	1476	4966	18075	27890	43420
SA	best opt.	30	12490	74583	904130	1035202	130100	417124012
	average	47	22784	190729	299669	983336	72635	557653234
	std	5	749	2183	12473	6616	526	701515
	std (%)	12.56	3.28	1.14	4.16	0.67	0.72	0.12
	time (s)	0.02	0.05	0.11	0.18	0.36	0.63	0.66

and large instances, no significant differences were observed either. On medium instances, the GA algorithm showed significantly better performance compared to the ACO, SA, ABC, SSA, and GWO algorithms, indicating its effectiveness in solving medium-sized TSP problems.

Table 5. T-test analysis between all old and new algorithms on different sizes ınstances for the best opt. solution

Instances	Algorithm 1	Algorithm 2	P-value	Significance	T-statistic
Small-scale instances (burma14, berlin52, kroA100)	GA	ACO	0.364547	-**	1.022030
	GA	SA	0.597260	-	0.573092
	ACO	SA	0.477207	-	-0.783348
	ABC	SSA	0.613100	-	0.547580
	ABC	GWO	0.707362	-	0.403291
	SSA	GWO	0.875125	-	-0.167471
	GA	ABC	0.880887	-	-0.159660
	GA	SSA	0.703137	-	0.409523
	GA	GWO	0.812443	-	0.253410
	ACO	ABC	0.340987	-	-1.079814
	ACO	SSA	0.445224	-	-0.845939
	ACO	GWO	0.409399	-	-0.920508
	SA	ABC	0.525536	-	-0.694607
	SA	SSA	0.857553	-	-0.191376
	SA	GWO	0.742382	-	-0.352314
Medium-scale instances (ts225&att532)	GA	ACO	**0.000608***	s**	40.530090
	GA	SA	**0.017399**	s	7.482005
	ACO	SA	**0.006236**	s	-12.604129
	ABC	SSA	**0.000849**	s	34.303569
	ABC	GWO	**0.000383**	s	51.060030
	SSA	GWO	**0.010278**	s	-9.787848
	GA	ABC	**0.040269**	s	-4.831471
	GA	SSA	**0.006605**	s	12.243384
	GA	GWO	**0.013171**	s	8.627054
	ACO	ABC	**0.000143**	s	-83.651662
	ACO	SSA	**0.000515**	s	-44.040645
	ACO	GWO	**0.000236**	s	-65.046094
	SA	ABC	**0.009082**	s	-10.421690
	SA	SSA	0.187524	-	-1.970884
	SA	GWO	0.051037	-	-4.255207
Large-scale instances (rat783 & dsj1000)	GA	ACO	0.439222	-	0.957841
	GA	SA	0.878502	-	0.173106
	ACO	SA	0.444202	-	-0.945509
	ABC	SSA	0.912785	-	0.123813
	ABC	GWO	0.911569	-	0.125553
	SSA	GWO	0.998748	-	0.001771
	GA	ABC	0.970035	-	-0.042397
	GA	SSA	0.942148	-	0.081953
	GA	GWO	0.940911	-	0.083711
	ACO	ABC	0.438202	-	-0.960386
	ACO	SSA	0.441407	-	-0.952411
	ACO	GWO	0.441436	-	-0.952339
	SA	ABC	0.850683	-	-0.213561
	SA	SSA	0.934531	-	-0.092787
	SA	GWO	0.935760	-	-0.091037

** The "Significance" column indicates the significance level (s for significant, - for not significant) of the difference between the algorithms.
* Significant p-values are bolded.

From Table 6, the t-test results of the average performance metric for medium-scale instances indicate statistically significant differences in the performance of certain optimization algorithms. On the contrary, the t-test findings for Table 7 show that there were no significant differences in the computation time performance of the majority of algorithm pairs. Table 8 presents the results of a statistical analysis comparing the performance of two algorithm groups, "old" and "new," based on different metrics. The P-values indicate the statistical significance of the differences observed, while the T-statistic represents the magnitude of the differences.

Table 6. T-test results of all algorithms on medium-scale instances for average solution

Instances	Algorithm 1	Algorithm 2	P-value	Significance	T-statistic
Medium-scale instances (ts225&att532)	GA	ACO	**0.000308**	s	56.947378
	GA	SA	0.106373	-	2.815849
	ACO	SA	0.297916	-	-1.394332
	ABC	SSA	0.946	-	-0.075331
	ABC	GWO	**0.001570**	s	25.206599
	SSA	GWO	**0.001513**	s	25.677421
	GA	ABC	0.763239	-	0.344630
	GA	SSA	0.819775	-	0.259
	GA	GWO	**0.000729**	s	37.024449
	ACO	ABC	**0.000359**	s	-52.795613
	ACO	SSA	**0.000355**	s	-53.041839
	ACO	GWO	**0.000490**	s	-45.164984
	SA	ABC	0.107474	-	-2.798749
	SA	SSA	0.107205	-	-2.802892
	SA	GWO	0.216172	-	-1.785087

Table 7. T-test results of all algorithms on all and large-scale instances for average solution

Instances	Algorithm 1	Algorithm 2	P-value	Significance	T-statistic
All instances	GA	ACO	0.055936	-	-2.115957
	GA	SA	0.078385	-	1.924068
	ACO	SA	0.050874	-	2.169134
	ABC	SSA	0.846396	-	-0.197952
	ABC	GWO	0.399972	-	-0.872663
	SSA	GWO	0.505295	-	-0.686749
	GA	ABC	0.254208	-	1.197562
	GA	SSA	0.289645	-	1.107834
	GA	GWO	0.482505	-	0.724739
	ACO	ABC	0.052600	-	2.150456
	ACO	SSA	0.052810	-	2.148232
	ACO	GWO	0.053769	-	2.138136
	SA	ABC	**0.026046**	s	-2.537748
	SA	SSA	**0.028856**	s	-2.481920
	SA	GWO	**0.023898**	s	-2.584516
Large-scale instances (rat783 & dsj1000)	GA	ACO	**0.046622**	s	-4.467759
	GA	SA	0.066393	-	3.684980
	ACO	SA	**0.044302**	s	4.591675
	ABC	SSA	0.555104	-	-0.702535
	ABC	GWO	0.127446	-	-2.525964
	SSA	GWO	0.192832	-	-1.933704
	GA	ABC	0.123737	-	2.571889
	GA	SSA	0.136871	-	2.417203
	GA	GWO	0.202491	-	1.869440
	ACO	ABC	**0.044954**	s	4.555899
	ACO	SSA	**0.045048**	s	4.550807
	ACO	GWO	**0.045375**	s	4.533231
	SA	ABC	**0.019205**	s	-7.111666
	SA	SSA	**0.015948**	s	-7.823580
	SA	GWO	**0.015742**	s	-7.875739

Table 8. T-test results for old and new two groups on all sizes instances for best opt., avg, std (%) and time

Metric	Instances	Algo. group1	Algo. group2	P-value	Significance	T-statistic
Best opt. solution	All	old	new	0.609777	-	-0.514437
	Small-scale	old	new	0.531586	-	-0.639439
	medium-scale	old	new	0.120642	-	-1.696514
	Large-scale	old	new	0.563546	-	-0.597362
Avg	All	old	new	0.702805	-	-0.384281
	Small-scale	old	new	0.812272	-	-0.241451
	medium-scale	old	new	**0.041796**	s	-2.333548
	Large-scale	old	new	0.667024	-	-0.443244
Std(%)	All	old	new	0.498638	-	0.682849
	Small-scale	old	new	0.626244	-	0.496570
	medium-scale	old	new	0.270391	-	1.166723
	Large-scale	old	new	0.650603	-	0.466863
Time(s)	All	old	new	0.072534	-	1.844435
	Small-scale	old	new	0.226801	-	1.256978
	medium-scale	old	new	0.229646	-	1.279392
	Large-scale	old	new	1.541858	-	0.154134

3.4. Discussions

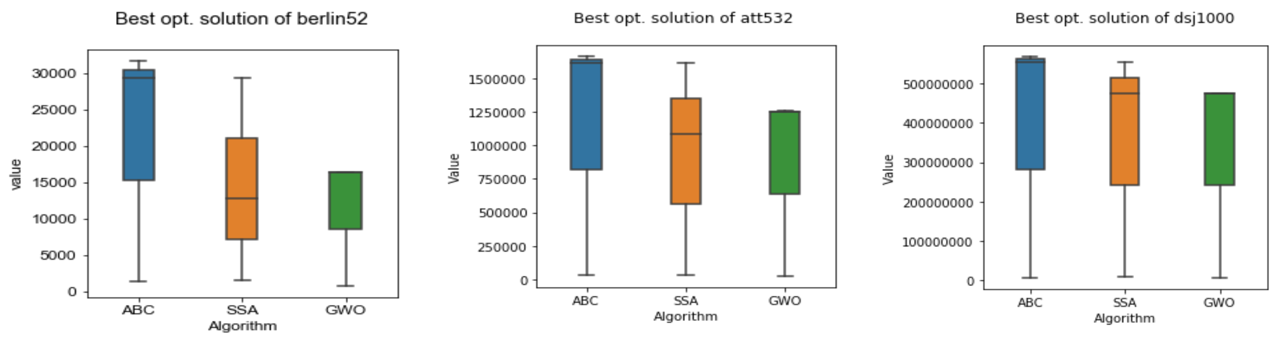
This part initiates a look into consulting the tables and figures that illustrate algorithmic contrasts in the TSP solution. It next explores the differences in algorithmic efficiency that are found among various subgroups of instances. The effects of these divergences are analyzed in order to shed light on the applicability of algorithms with regard to TSP issue sizes and complications.

According to Table 3, as the size of the TSP instances increases, GWO tends to outperform ABC and SSA in terms of finding the best optimal solution, which is depicted in the box plot in Figure 2 for three different sizes of instances. Larger cases show this tendency due to GWO's boosted exploration and exploitation in complicated and broad search environments. GWO's techniques may help it explore wider solution landscapes and find superior optimal solutions in larger TSP cases than ABC and SSA. GWO's adaptability and robustness in scaling up to increasingly complicated problem sizes in the TSP domain may explain its efficiency in handling larger instances. This suggests that GWO is more effective at handling larger TSP instances.

The standard deviation percentage metric normalizes the standard deviation relative to the best optimal solution, indicating how close the solutions are to the best. ABC and GWO consistently achieve low standard deviations because, with their specific exploration and exploitation tactics, they can converge on optimal solutions and preserve solution consistency across TSP examples. Additionally, the time taken by each algorithm to solve the TSP instances increases as the size of the instances increases. GWO is the fastest algorithm, followed by ABC and SSA. This indicates that GWO is efficient in terms of computation time, making it a favorable choice for solving larger TSP instances. All of these convergences are illustrated in Figure 3 via the line plot approach and in Figure 4 via the box plot approach.

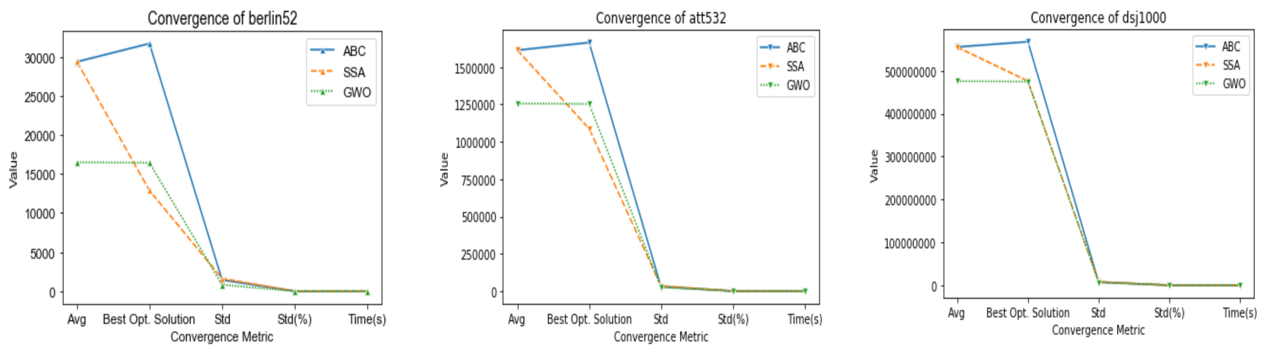
As shown in Figure 5 and Figure 6, ACO excels over GA and SA in various TSP scenarios due to its effective optimization approaches and adaptability. In small-scale instances, ACO surpasses GA and SA with competitive optimal solutions with lower values. ACO outperforms GA and SA in optimal and average medium-scale solutions like ts225 and att532. GA outperforms ACO and SA in optimal solutions in large-scale examples like rat783 and dsj1000, while ACO remains competitive with extended computation times, demonstrating its TSP endurance. ACO's strength to balance exploration and exploitation in slightly complicated solution domains makes it better than GA and SA in medium-scale cases. ACO's pheromone-based exploration and local search techniques usually handle medium-scale situations with diligent optimization. In these TSP situations, ACO gets to high-quality solutions faster than GA and SA because it quickly looks at a lot of different solutions and takes advantage of potential areas.

Overall, the comparison highlights the varying performance of the algorithms based on the problem instance size. GA shows promise for small and large-scale instances, while ACO consistently performs well across different scales. SA demonstrates effectiveness for small-scale instances but struggles to scale effectively for larger problems. Researchers should consider these performance differences and choose the appropriate algorithm based on the problem size and optimization requirements. All of these convergences of small, medium, and large-scale instances are illustrated in Figure 6 via the line plot approach as well as in Figure 7 via the box plot approach



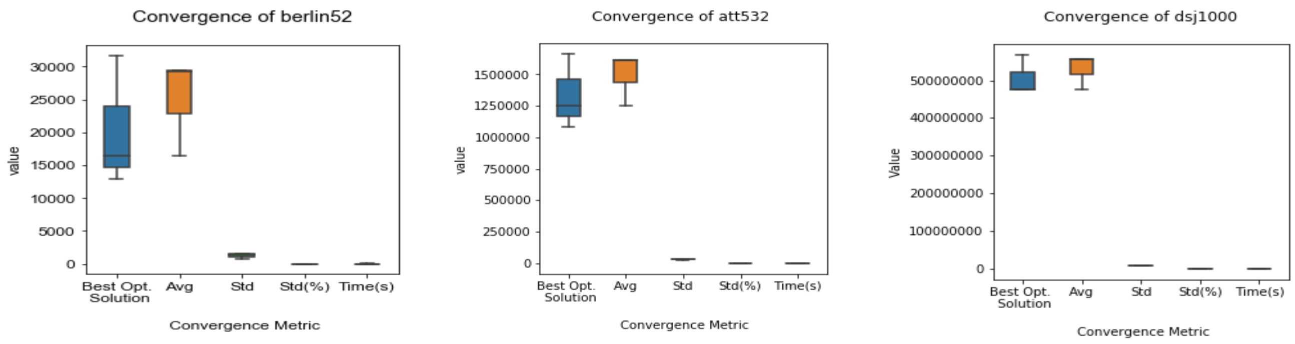
a) berlin52 b) att532 c) dsj1000

Figure 2. Box plot convergence of new algorithms on three different sizes instances for best opt. Solution.



a) berlin52 b) att532 c) dsj1000

Figure 3. Line plot convergence of new algorithms on three different sizes instances for all performance metrics.



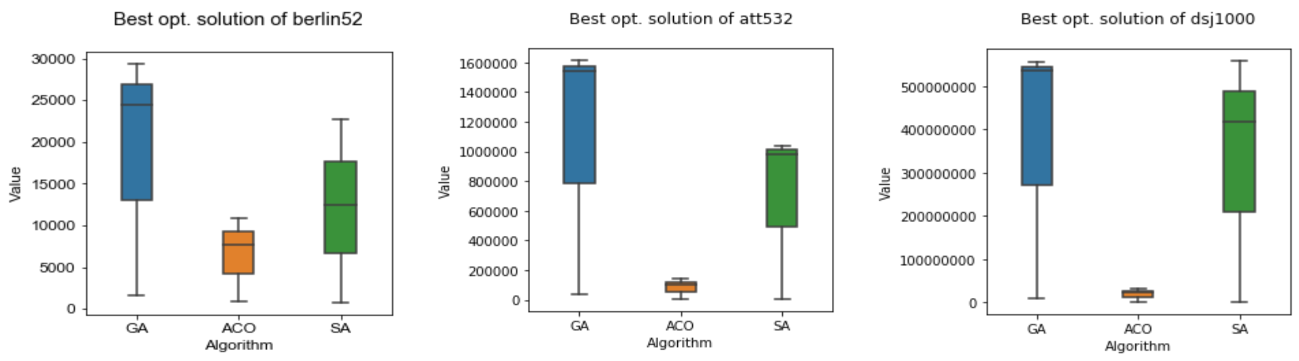
a) berlin52 b) att532 c) dsj1000

Figure 4. Box plot convergence of new algorithms on three different sizes instances for all performance metrics.

Table 6 reveals that GA and GWO outperform other algorithms in medium-sized optimization problems like TSP due to their convergence, exploration, and exploitation capabilities. GA, known for its population-based search and crossover-mutation mechanisms, is effective in exploring diverse solution spaces but less efficient in convergence. GWO, inspired by grey wolves' hunting behaviors, utilizes promising search space areas more efficiently, improving convergence and solution quality for medium-scale instances. The t-test results for Table 7 provide some interesting findings for two groups of instances. GA, SA, and ABC exhibit identical outcomes in the "All instances" segment, with insignificant variations in the average solution values. On the other hand, ACO performs differently from GA, SA, and ABC, showing substantive variations. Interestingly, ACO and SA show significant variances when contrasted with several algorithms in the "Large-scale instances," indicating varying effectiveness in handling larger-scale TSP situations. In both segments, ABC, SSA, and GWO show stability with each other as well as with other algorithms.

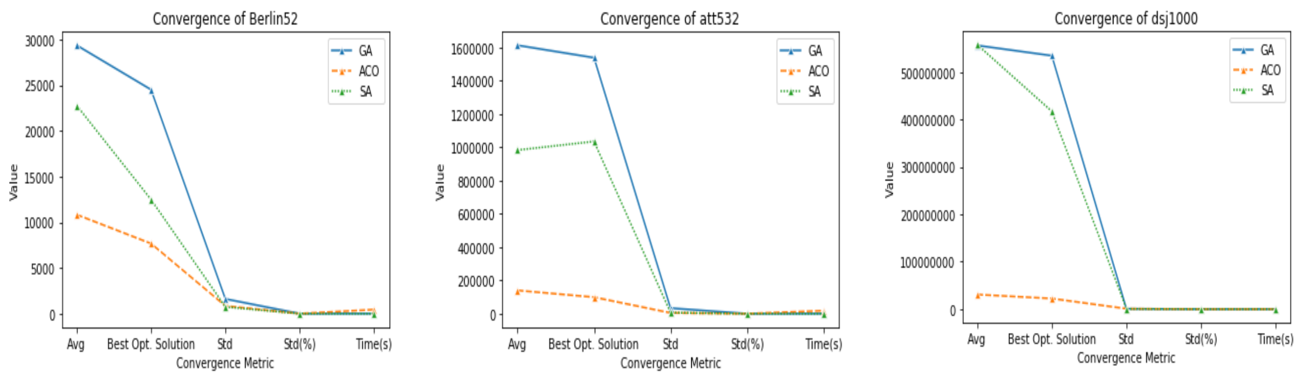
In the most significant evaluations from Table 8 for the metric "best opt. solution," there is no significant difference between the old and new algorithm groups across all instances, including small-scale, medium-scale, and large-scale cases. In terms of average performance (Avg), there is no significant difference between the old and new algorithm groups for all instances and small-scale instances. However, for medium-scale instances, the new algorithm group shows a statistically significant improvement with a lower T-statistic value. Regarding the metric "Std(%)", which represents the standard deviation, there is no significant difference between the old and new algorithm groups for all instances and small-scale instances. However, for medium-scale instances, the new algorithm group exhibits a statistically significant improvement with a higher T-statistic value.

Lastly, in terms of the "Time(s)" metric, which measures the execution time, there is no significant difference between the old and new algorithm groups for all instances and small-scale instances. However, for medium-scale instances, the new algorithm group shows a statistically significant improvement with a higher T-statistic value. Overall, the results suggest that the new algorithm group shows promising improvements in terms of average performance, standard deviation, and execution time for medium-scale instances, and there is no significant difference between the two groups for large-scale instances. while no significant differences are observed in other instances. The overall convergence graph of all algorithms on att532 as a medium-scale instance is shown in Figure 8.



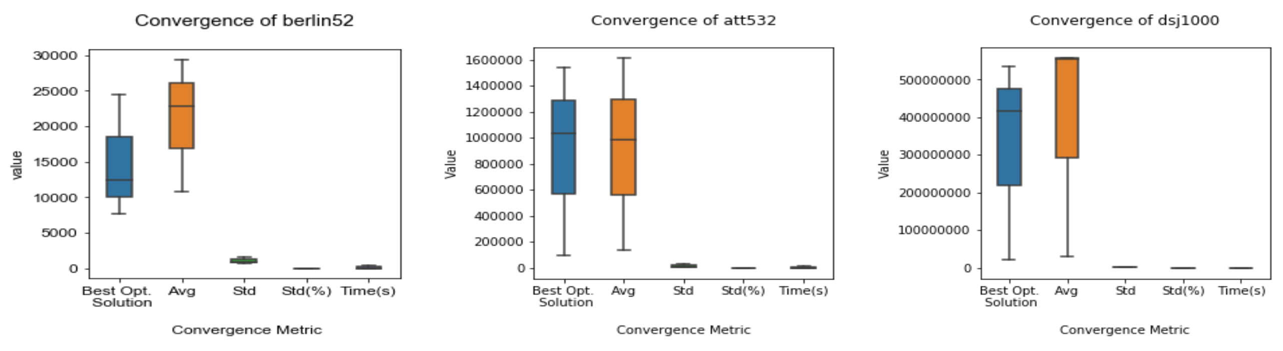
a) berlin52 b) att532 c) dsj1000

Figure 5. Box plot convergence of old algorithms on three different sizes instances for best opt. Solution.



a) berlin52 b) att532 c) dsj1000

Figure 6. Plot convergence of old algorithms on three different sizes instances for all performance metrics.



a) berlin52 b) att532 c) dsj1000

Figure 7. Box plot convergence of old algorithms on three different sizes instances for all performance metrics

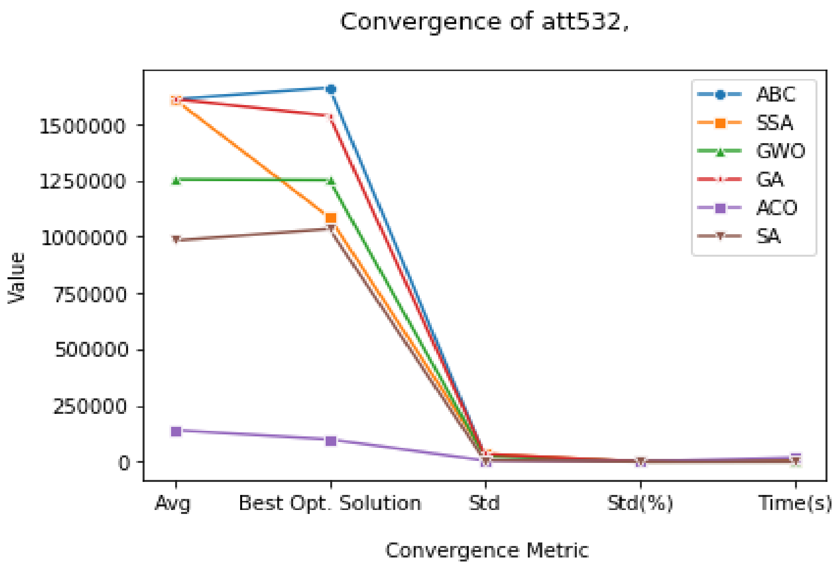


Figure 8. Overall convergence of all algorithms on att532 instances for all performance metric.

5. Conclusion

The study compared TSP optimization techniques and found significant improvements in convergence and scalability for medium-scale instances using new algorithms (ABC, SSA, and GWO). However, small and large-scale incidents did not differ significantly. The new algorithms also showed significant computational time enhancements for medium-scale instances. These findings can help researchers and practitioners choose TSP optimization techniques, but the study's limitations include a restricted collection of benchmark instances and the use of a few optimization strategies.

Future TSP optimization research should include more benchmark examples of small, medium, and large-scale scenarios, including real-world examples. There's an opportunity to refine and customize optimization algorithms for small and large-scale instances to improve efficiency and address the performance gap identified in this study. Looking into hybrid techniques that take the best parts of several algorithms and combine them may help find better solutions that take less time to compute. This is especially true for large-scale scalability.

Contributions of the authors

M.A.A.H. conducted the core research, devised the experimental setup, wrote the paper, and was vital in analyzing the findings and formulating the conclusions. Z.Y.A. contributed to the literature review, edited the paper, and provided theoretical guidance.

Conflict of Interest Statement

There is no conflict of interest between the authors.

References

[1] H. Cheng, H. Zheng, Y. Cong, W. Jiang, and S. Pu, “Select and Optimize : Learning to Solve Large-Scale Traveling Salesman Problem,” in *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, 2022, vol. 206, pp. 1219–1231.

[2] Y. Wu, T. Weise, and R. Chiong, “Local Search for the Traveling Salesman Problem: A Comparative Study,” in *Proceedings of 2015 IEEE 14th International Conference on Cognitive Informatics and Cognitive Computing, ICCI*CC 2015*, 2015, pp. 213–220. doi: 10.1109/ICCI-CC.2015.7259388.

[3] M. DİRİK, “Optimization: A comparison of recent meta-heuristic optimization algorithms using benchmark function,” *J. Math. Sci. Model.*, vol. 5, no. 3, pp. 113–124, 2022, doi: 10.33187/jmsm.1115792.

[4] M. ŞAHİN, “Improvement of the Bees Algorithm for Solving the Traveling Salesman Problems,” *Bilişim Teknol. Derg.*, vol. 15, no. 1, pp. 65–74, 2022, doi: 10.17671/gazibtd.991866.

[5] W. Li, C. Wang, Y. Huang, and Y. ming Cheung, “Heuristic smoothing ant colony optimization with differential information for the traveling salesman problem,” *Appl. Soft Comput.*, vol. 133, p. 109943, 2023, doi: 10.1016/j.asoc.2022.109943.

[6] B. A. Ajayi, M. A. Magaji, S. Musa, R. F. Olanrewaju, and A. A. Salihu, “A Comparative Analysis of Optimization Heuristics Algorithms as Optimal Solution for Travelling Salesman Problem,” in *Proceedings of the 5th International Conference on Information Technology for Education and Development (ITED) 2022*, 2022, pp. 3–10. doi: 10.1109/ITED56637.2022.10051627.

[7] M. Mondal and D. Srivastava, “A Genetic Algorithm-Based Approach to Solve a New Time-Limited Travelling Salesman Problem,” *Int. J. Distrib. Syst. Technol.*, vol. 14, no. 2, pp. 1–14, 2023, doi: 10.4018/IJDST.317377.

[8] L. S. Hasan, “Artificial Bee Colony Algorithm and Bat Algorithm for Solving Travel Salesman Problem,” *Webology*, vol. 19, no. 1, pp. 4185–4193, 2022, doi: 10.14704/web/v19i1/web19276.

[9] M. Khajehzadeh, A. Iraji, A. Majdi, S. Keawsawasvong, and M. L. Nehdi, “Adaptive Salp Swarm Algorithm for Optimization of Geotechnical Structures,” *Appl. Sci.*, vol. 12, no. 13, 2022, doi: 10.3390/app12136749.

[10] Y. Liu, L. Xu, Y. Han, X. Zeng, G. G. Yen, and H. Ishibuchi, “Evolutionary Multimodal Multiobjective Optimization for Traveling Salesman Problems,” *IEEE Trans. Evol. Comput.*, pp. 1–1, 2023, doi: 10.1109/tevc.2023.3239546.

[11] I. Khan, K. Basuli, and M. K. Maiti, “Multi-objective covering salesman problem: a decomposition approach using grey wolf optimization,” *Knowl. Inf. Syst.*, vol. 65, no. 1, pp. 281–339, 2023, doi: 10.1007/s10115-022-01752-y.

[12] K. Panwar and K. Deep, “Discrete Salp Swarm Algorithm for Euclidean Travelling Salesman Problem,” *Appl. Intell.*, 2022, doi: 10.1007/s10489-022-03976-5.

[13] I. Khan and M. K. Maiti, “A swap sequence based Artificial Bee Colony algorithm for Traveling Salesman Problem,” *Swarm Evol. Comput.*, vol. 44, no. November 2016, pp. 428–438, 2019, doi: 10.1016/j.swevo.2018.05.006.

[14] S. Sobti and P. Singla, “Solving Travelling Salesman Problem Using Artificial Bee Colony Based Approach,” *Int. J. Eng. Res. Technol.*, vol. 2, no. 6, pp. 186–189, 2013, [Online]. Available: http://www.ijert.org/browse/volume-2-2013/june-2013-edition?download=3722:solving-travelling-salesman-problem-using-artificial-bee-colony-based-approach&start=20

[15] S. Sharma and V. Jain, “A Novel Approach for Solving TSP Problem Using Genetic Algorithm Problem,” *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 1116, no. 1, p. 012194, 2021, doi: 10.1088/1757-899x/1116/1/012194.

[16] W. Li, L. Xia, Y. Huang, and S. Mahmoodi, “An ant colony optimization algorithm with adaptive greedy strategy to optimize path problems,” *J. Ambient Intell. Humaniz. Comput.*, vol. 13, no. 3, pp. 1557–1571, 2022, doi: 10.1007/s12652-021-03120-0.

[17] E. Chandomi-Castellanos *et al.*, “Modified Simulated Annealing Hybrid Algorithm to Solve the Traveling Salesman Problem,” in *2022 8th International Conference on Control, Decision and Information Technologies, CoDIT 2022*, 2022, pp. 1536–1541. doi: 10.1109/CoDIT55151.2022.9804145.

[18] T. Tlili and S. Krichen, “A simulated annealing-based recommender system for solving the tourist trip

design problem," *Expert Syst. Appl.*, vol. 186, no. October 2020, p. 115723, 2021, doi: 10.1016/j.eswa.2021.115723.

[19] T. Ye, H. Wang, W. Wang, T. Zeng, L. Zhang, and Z. Huang, "Artificial bee colony algorithm with an adaptive search manner and dimension perturbation," *Neural Comput. Appl.*, vol. 34, no. 19, pp. 16239–16253, 2022, doi: 10.1007/s00521-022-06981-4.

[20] A. Ebrahimnejad, M. Enayattabr, H. Motameni, and H. Garg, "Modified artificial bee colony algorithm for solving mixed interval-valued fuzzy shortest path problem," *Complex Intell. Syst.*, vol. 7, no. 3, pp. 1527–1545, 2021, doi: 10.1007/s40747-021-00278-0.

[21] P. Chen, M. Liu, and S. Zhou, "Discrete Salp Swarm Algorithm for symmetric traveling salesman problem," *Math. Biosci. Eng.*, vol. 20, no. 5, pp. 8856–8874, 2023, doi: 10.3934/mbe.2023389.

[22] S. Kassaymeh, S. Abdullah, M. A. Al-Betar, M. Alweshah, M. Al-Laham, and Z. Othman, "Self-adaptive salp swarm algorithm for optimization problems," *Soft Comput.*, vol. 26, no. 18, pp. 9349–9368, 2022, doi: 10.1007/s00500-022-07280-9.

[23] F. Boualem, S.M., Meftah, B., Debbat, "Solving Travelling Salesman Problem Using a Modified Grey Wolf Optimizer," in *International Conference on Artificial Intelligence in Renewable Energetic Systems*, 2022, pp. 708–716. doi: https://doi.org/10.1007/978-3-030-92038-8_71.

[24] H. Liu, B. Lei, W. Wang, G. Zhong, and H. Chai, "An improved grey wolf optimization for solving TSP," in *International Conference on Computer Science and Communication Technology (ICCSCT 2022)*, 2022, no. December 2022, p. 29. doi: 10.1117/12.2661782.

[25] R. G. Revision, "TSPLIB 95 — TSPLIB 95 0.7.1 documentation," *Sphinx*, 2018. https://tsplib95.readthedocs.io/en/stable/pages/readme.html (accessed Feb. 24, 2023).

[26] N. Sultana, J. Chan, T. Sarwar, and A. K. Qin, "Learning to Optimise General TSP Instances," *Int. J. Mach. Learn. Cybern. Vol.*, vol. 13, pp. 2213–2228, 2022, doi: https://doi.org/10.1007/s13042-022-01516-8.