Solving the Travelling Salesman Problem Using an Ant Colony System Algorithm

Zakir Hussain Ahmed [1,*], Majid Yousefikhoshbakht [2], Abdul Khader Jilani Saudagar[3], and Shakir Khan[4,5] zaahmed@imamu.edu.sa

[1] Department of Mathematics and Statistics, College of Science, Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh 11432, Saudi Arabia.

[2] Department of Mathematics, Faculty of Sciences, Bu-Ali Sina University, Hamedan 6517838195, Iran.

[3] Information Systems Department, Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh 11432, Saudi Arabia

[4] College of Computer and Information Sciences, Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh 11432, Saudi Arabia.

[5] University Centre for Research and Development, Department of Computer Science and Engineering, Chandigarh University, Mohali 140413, India.

[*] Correspondence: zaahmed@imamu.edu.sa

Abstract

The travelling salesman problem (TSP) is an important combinatorial optimization problem that is used in several engineering science branches and has drawn interest to several researchers and scientists. In this problem, a salesman from an arbitrary node, called the warehouse, starts moving and returns to the warehouse after visiting *n* clients, given that each client is visited only once. The objective in this problem is to find the route with the least cost to the salesman. In this study, a meta-based ant colony system algorithm (ACSA) is suggested to find solution to the TSP that does not use local pheromone update. This algorithm uses the global pheromone update and new heuristic information. Further, pheromone evaporation coefficients are used in search space of the problem as diversification. This modification allows the algorithm to escape local optimization points as much as possible. In addition, 3-opt local search is used as an intensification mechanism for more quality. The effectiveness of the suggested algorithm is assessed on a several standard problem instances. The results show the power of the suggested algorithm which could find quality solutions with a small gap, between obtained solution and optimal solution, of 1%. Additionally, the results in contrast with other algorithms show the appropriate quality of competitiveness of our proposed ACSA.

Keywords: *Travelling Salesman Problem; Ant Colony System Algorithm; Global Updating; NP-hard Problems.*

1. Introduction

The travelling salesman problem (TSP) is an important combinatorial optimization problem (COP) in services and industry because many problems can be converted into a version of the problem and hence, the algorithms for this problem can be applied to solve them [1]. Also, the problem is considered as a standard criterion for new discrete algorithms, and thus, the effectiveness of the algorithms can be evaluated in contrast with other algorithms. In this important COP, the two-dimensional specifications of a node called a warehouse and *n* other nodes called customers are presented as input data. Therefore, the Euclidean distance between the two nodes can be obtained. Now, a salesman starts moving from the warehouse node and returns to it at the end after visiting each customer once. The purpose is to discover a Hamiltonian cycle having the least Euclidean distance (Fig. 1) [2].

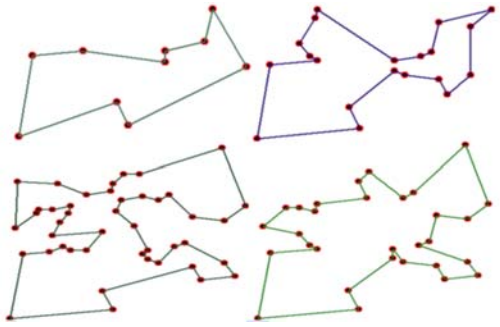

Fig. 1. Examples of sub-optimum solutions for different TSPs

Exact and heuristic algorithms are two classes of methods that can be used to solve this problem as well as other COPs [3]. Using basic exact methods to solve this problem, one can make a list of all possible permutations of nodes, evaluate their costs, and select the best amongst them. However, to use these basic methods, one has to perform $(n-1)!/2$ computations for an undirected graph and $(n-1)!$ computations for a directed graph. Thus, as the number of nodes grows, problem size becomes bigger, computations grow rapidly, and thus efficiency of the methods decline drastically so that after some certain problem size, the methods cannot find solution. That means, these methods require very long computational time for solving these problems, and since these problems need to be solved within short time, these methods are not

Manuscript received February 5, 2023
Manuscript revised February 20, 2023
https://doi.org/10.22937/IJCSNS.2023.23.2.6

recommended to use. Branch and cut algorithm [4], branch and cut algorithm [5], lexisearch algorithm [6], and Lagrangian algorithm [7] are some examples of this type of algorithms.

Since the TSP, as mentioned above, has many applications, finding solution to this problem is very important in reducing the cost of industrial and service companies and hence, has attracted a lot of attention. In recent years, many methods were suggested to solve this problem that have been dealt with seriously for almost three decades, the solution to this problem is not achieved in most cases, but the method can achieve a quality solution at an acceptable time [8]. Heuristic methods are methods that can provide solutions to any COP in a limited time and are mainly based on counting methods, except that they use additional information to guide the search. These methods are completely general in terms of application and can solve complex problems. However, in heuristic methods, unlike metaheuristic methods (another version of heuristic methods) a constant solution is obtained for the problem in each repetition of the method. So, it is unjustified to repeat the method to find a better solution and this is an advantage of the method, but in contrast, these methods cannot escape from the local optimal points (Fig. 2) and thus, easily get stuck in these points [9].

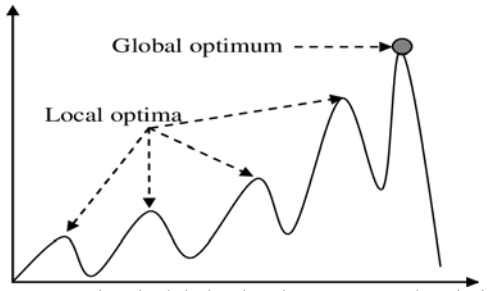

Fig. 2. Local and Global points in a compound optimization problem

One of the most important and promising versions of heuristic methods that do not have problems with complete counting methods are metaheuristic methods that have recently focused more on them. These methods have similarities with social or natural systems and their application is derived from continuous methods of research which has had very good results in solving the complex problems. In these methods where the performance of the method, different from heuristic methods, controlled by the client, solutions are found within a reasonable time. Even though obtained solutions are better than the solutions using heuristic methods and the methods possibly do not get stuck in local optimizations, but there are numerous parameters which must be set by the user, may be experimentally. However, the parameters setting is also a difficult job. Different values of the parameters cause the methods to obtain different solutions in similar iterations. Therefore, these methods do not have a rigid process for finding the solutions and random parameter setting play an important role in these methods. Tabu search (TS) [10], genetic algorithm (GA) [11], scatter algorithm (SA) [12], ant colony system (ACS) [13] bee colony optimization (BCO) [14], memetic algorithm (MA) [15], and neural network (NN) [16] are examples of metaheuristic algorithms.

Hybrid metaheuristic and heuristic methods are new methods that are highly considered by researchers and scientists today, because the combination of these methods makes the advantages of each method as best as possible and obtains appropriate and good quality solutions within very short time [17-20]. The combination of such methods also includes the benefits of heuristic method, which obtains a good solution within a shorter time and has good efficiency to escape the optimal local points based on metaheuristic methods [21]. In other words, such algorithms have a more suitable structure for searching in the space of solutions and have a more regular procedure for finding elite solutions. In these algorithms, first, the space is searched as much as possible at an acceptable time, and the susceptible areas with elite solutions are obtained, then the search process is pushed from the global to the local state, and the algorithm searches for the neighborhoods of elite solutions with a more exact procedure [22].

The TSP is NP-hard, meaning that no polynomial algorithm is available to obtain optimal solution, and the computing time to find solution grows exponentially through its size. Therefore, these kinds of problems are often solved with heuristic and metaheuristic techniques. Also, because the TSP has numerous industrial and service applications, an efficient metaheuristic algorithm equipped with diversification and intensification mechanisms is suggested. We propose an efficient ant colony system algorithm (ACSA) that uses 3-opt local search as intensification mechanism. Several sets of benchmark instances are considered in this paper for comparing results by our algorithm with the results by several other metaheuristic algorithms. The experimental results indicate that our algorithm can provide better solutions for almost all instances within an acceptable computational time.

This paper is arranged as follows. The components of the proposed ACSA to solve the TSP are described in section 2 and the parameters setting are considered in section 3. Section 4 reports comparative study of our suggested algorithm with other metaheuristic algorithms tested on some standard instances from the literature. Finally, the conclusions are summarized in Section 5.

2. The Proposed ACS Algorithm

ACS is one of the oldest ant colony optimization (ACO) methods which can find sub-optimum solutions within a short time. This algorithm has been an improvement over ant system (AS) due to the introduction of new mechanisms. In this method, the ant that is capable of finding the best solution, releases pheromone on its path in every iteration [23]. Alternatively, the best solution in the current iteration is encouraged to improve its usefulness. After some time, all ants in AS build similar paths. If the path is not found good, means that the algorithm fails to find better solutions, then the searching and extracting procedure is terminated. Thus, the algorithm fails to find efficient solutions for large sized problem instances.

In this paper, we propose to use an algorithm to examine the space areas more accurately. So, a modified ACS algorithm is used here. The ACS, introduced in [24], was largely inspired by the ant system method. In addition, the algorithm has attained some improvements due to the introduction of new mechanisms in terms of efficiency in comparison to other types of ACO algorithms. Here, due to the weaknesses in the ACS algorithm, some modifications are made to the algorithm.

2.1. Heuristic problem information

One of the coefficients that is of great importance in ACS and has a very important role in selecting nodes, is the heuristic information coefficient of the problem which is shown by $\eta_{ij}(t)$. This coefficient, in the beginning of the algorithm's iterations where the pheromone's impact is less important, has a key role and decides the position and side of the search areas. In the conventional ACS algorithm, this value is considered equal to the inverse of the cost between the two nodes, like the nearest neighborhood method, the lower the cost the more likely it is to be chosen. As the nearest neighborhood algorithm is not obtaining well good quality solutions, it can be replaced with some efficient heuristic algorithms. For this reason, it is proposed to use the idea of a saving algorithm, which is an efficient innovative method that has more power to find good quality solutions within lesser time. Therefore, this coefficient is considered equal to $\eta_{ij}(t) = a c_{i0} + b c_{0j} - c c_{ij} + d \left| c_{i0} - c_{0j} \right|$, where *a, b, c* and *d* are fixed numbers which are adjusted by the user and the cost is between 'node *i*' and 'node *j*'. Table 1 shows the tested values and the best values found for these parameters. In this table, by considering three parameters b, c and d equal to 1, the best value *a* is obtained 1.5 by testing instances and comparing the solutions. Then, by considering value a, c and d to 1.5, 1 and 1 respectively, the best value b is obtained. Finally, the best values c and d are similarly found.

2.2. Pheromone evaporation coefficient

This parameter is used at the bottom of each iteration of the ACS algorithm, which is shown as a constant value. This coefficient decreases the pheromone quantity in all edges to a certain value. In this study, this coefficient changes due to the correct analysis of the ACS algorithm and converts from constant coefficient to variable coefficient. Therefore, by changing the state of the algorithm, this coefficient changes and makes the algorithm more efficient. The fixed coefficient of the pheromone does not make any difference when implementing the algorithm for any solution quality, in spite of the reality that the solutions are less accurate at the starting of the algorithm and the heuristic information of the instance has more power to orientation of the algorithm. However, the number of iterations is added, and the appropriate usage of the pheromone quantity poured on the edges, the solutions will reach a higher quality. So, it is well to consider this coefficient as an ascending function from the starting of the algorithm. Also, since the stopping condition for the algorithm is defined by the multiple number of nodes of each problem, this functional coefficient of iterations is considered.

Therefore, it is suggested that the pheromone evaporation rate at the starting of the algorithm is low and with increasing iterations this value increases. This method triggers the algorithm to use a logical process to search and find better solutions locally and globally initially and finally by the algorithm respectively. Because the accuracy of the solutions at the starting is low and the low evaporation pheromone quantity causes the difference in the pheromone on the met and unsealed manes in the algorithm's iteration to increase until the chances of being selected by the unsealed edges increase. This also triggers the algorithm for searching globally at the starting of the process, but when iterations increase and more high-quality solutions are produced, searching is better to tend from global to local to search for the same as elite solutions. Therefore, increasing the evaporation of the fermion with an ascending function causes this goal to be realized because by increasing the pheromone coefficient, the difference between the met and unsealed mane is increased and the chances of unsealed edges being selected in previous iterations are close to zero. Therefore, the algorithm is more likely to consider previously met edges along with their neighbors.

Therefore, the objective of this study is to present a function that is ascending and always has a range [0,1] based on the conditions of the problem parameters (formula (1)). In this process and in the searches performed, an

ascending trigonometric function instead of constant coefficient is suggested in formula (1), which makes it possible to obtain better results for the algorithm, which is dealt with in the next section.

$$\rho_t = 1 - \rho \cos(\frac{\pi t}{3loop}) \quad (1)$$

loop: The total number of algorithm loop repetitions which is considered as the final condition of the algorithm.

t: Represents the repeat number of the algorithm, which is at least equal to 1 and maximum.

ρ_t: The evaporation rate is new, which increases with time.

ρ: Fixed evaporation rate for algorithms that is always considered from the interval [0,1].

Since *t* changes from 1 to n, the value $(\pi t/3n)$ changes between zero and $(\pi/3)$ consequently has a descending function between [0.5, 1] and $[\ 1 - \rho\, cos(\pi t/3n)]$ ultimately is based on ρ, the ascending functional coefficient in [0, 1].

2.3. The 3-Opt algorithm

In obtaining higher quality solutions, our proposed ACSA algorithm is merged with the 3-opt local search improving algorithm. This local search method is enabled once a better quality solution is obtained for the algorithm than in the previous iterations. The reason why local search is activated in the situation is that once a better quality solution is obtained than in the previous iterations, there may be another better quality solution in the neighbor, which can be achieved by more searches in the neighbor. This method, as presented in Fig. 3, works based on removing three edges from the solution and re-connecting those three edges in a different way. There are several ways to re-create the tour, but only if it applies to the problem constraints and the recent tour gets a better quality solution than in the previous iteration. The operation of removing three edges and re-connecting again continues continuously to the point that no new improver movements are found for the algorithm.

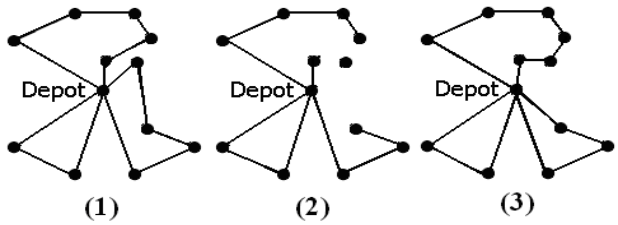

Fig. 3. The 3-Opt local search

3. Parameters Analysis

Premature convergence causes a search to be concentrated prematurely around sub-optimum solutions, which will bring about the abandonment of the search for new solutions in the algorithm. Search stagnation refers to a situation in which all ants follow a single path and construct a single solution a few times. In such cases the system will stop discovering new and good routes and cannot generate better solutions. In other words, premature convergence occurs when the algorithm prematurely focuses only on one part of the solution space and its neighbors instead of doing an adequate global search all over the search space. In contrast, most successful meta-heuristic methods first conduct an adequate global search in order to identify potential points for good solution and then move toward these points for finding near-optimum solutions. One of the best ways to prevent premature convergence is to use parameter setting for the algorithm, which allows the algorithm to perform appropriate global and local searches in the problem space with the appropriate number of parameters.

Parameter setting is very important for meta-heuristic algorithms because it causes the algorithm to achieve values for its parameters, which makes it likely to obtain its best values for the test instances. However, due to time limits, it cannot investigate all the possible values of the parameters. For this reason, in addition to using efficient methods such as Taguchi, one should also use the experience of others. In other words, it is better to get the minimum and maximum efficiency of these parameters from other articles and based on these values, consider some of them as candidates and then use other methods to find their best values. Note that due to time constraints, there is no guarantee that all of these parameters will be optimal, but it is likely that the parameter values will be among the best values found for them.

A number of parameters of our algorithm, which have a significant impact on our algorithm's convergence, are considered and analyzed in this section. For this purpose, different values are tested for each parameter in order to find values close to their normal values. Then, Taguchi method is utilized for finding the best parameter values and the final parameter values are shown in the Table 1. Note that for this setting the parameter instance eli76 is considered and for each candidate value in Taguchi method, each problem is examined ten times and the best of these values are reported. Also, for each parameter test, the other parameters are considered constant so that the best value can be found for it. Four parameters of this algorithm including initial pheromone on edges, α, β and *loop* are considered separately to investigate the effect of parameters of the ACSA on the solution. It should be noted that α *and* β are the ability of the effect of the pheromone and the heuristic information of the problem in the next mane

by ants and loop is the repetition rate of the algorithm, respectively.

Table 1: Parameter setting for our proposed ACSA.

Parameter	Candidate values	The best value
Initial pheromone on edges	10, 20, …, 100	20
α	1-5	1
β	1-5	4
Number of iterations based on number of instance nodes (loop)	1-5	2

4. Computational Experiments

First, various standard problem instances are studied, and the solutions by our proposed ACSA algorithm are evaluated against the solutions by elitist-ant system (EAS) and common ACS. These metaheuristic algorithms are executed ten times for each instance due to weaving different solutions in each execution. Then the obtained best solutions are displayed in Table 2. Further, the results of ACSA are compared with other metaheuristic algorithms with the same conditions as Table 1 which are also shown in Table 2. In other words, their best solutions is compared in this table. The instances are provided on the website: http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp.

Table 2: Results by the three algorithms for some instances.

	Instance	n	ACSA				EAS	ACS	BKS
			Tbest	WS	AS	BS			
1	GR48	48	10	**5046**	**5046**	**5046**	5075	**5046**	5046
2	ATT48	48	10	**10628**	**10628**	**10628**	10701	10641	10628
3	Eil51	51	10	**426**	**426**	**426**	429	429	426
4	Berlin52	52	10	**7542**	**7542**	**7542**	7591	**7542**	7542
5	ST70	70	10	690	682	**675**	684	684	675
6	Eil76	76	10	**538**	**538**	**538**	545	**538**	538
7	KroA100	100	10	21982	21462	**21282**	21456	21362	21282
8	KroB100	100	10	**22141**	**22141**	**22141**	22304	22301	22141
9	Eil101	101	10	662	636	**629**	642	642	629
10	Lin105	105	10	14401	14384	**14379**	14703	14421	14379
11	KroA150	150	10	28563	26726	**26524**	26821	26821	26524
12	KroB150	150	10	27602	26352	**26130**	26439	26439	26130
13	KroA200	200	10	31028	29883	29451	30098	29743	29368
14	KroB200	200	10	31651	29802	29506	30672	29761	29437

In Table 2, the number of nodes in each instance (n), number of execution required to find the best solution (Tbest), worst solution (WS), average solution (AS) and best solution (BS) found by our ACSA, and the best solutions obtained by common EAS and ACS, and the best known solution (BKS) are reported. According to the results reported in the table, the weakest algorithm among the three presented algorithms is the EAS algorithm, which could not achieve BKS for any of the 14 instances in the table. Therefore, this algorithm does not have the required performance for these instances. Alternatively, the conventional ACS algorithm that is a modification to EAS, is able to find better solution than the EAS algorithm. To be precise, the ACS algorithm is able to find BKS for three instances and so, for nine out of fourteen instances it obtained better quality solution than by the EAS algorithm. However, the results by these two algorithms are equal for the remaining five instances and the EAS algorithm has not performed well than the ACS algorithm for any instance. Therefore, in general, the ACS algorithm has shown better performance than the EAS algorithm in every respect. By comparing our proposed ACSA with other two algorithms, it is found that our algorithm has a great performance to solve the instances and is able to find the BKS for twelve out of fourteen instances. In addition, for the remaining two instances, the algorithm has obtained better solutions than the previous two algorithms. These results find that the revisions made in our proposed algorithm caused the

algorithm to avoid local optimizations and found better solutions.

We have calculated the solution gap between the obtained solution and the BKS by the formula: *Gap=100*(d1-d2)/d2*, where d1 and d2 are the solution by an algorithm and BKS, respectively. Fig. 4 compares the worst, best and average solution *Gap*s for the instances by our proposed algorithm based on the results reported in Table 1. In this figure, X-axis represents the instances and Y-axis represents the *Gap*s. It is found in the figure that among the ten repetitions of the algorithm for the six instances - 1, 2, 3, 4, 6 and 8, there is no difference among three results. In addition, it is showed that for the instance 10, the *Gap* is very low. In general, since the *Gap* for the remaining instances is less than 8%, it can be concluded that for this group of instances, our algorithm has a worthy ability to find quality solutions and various execution of the algorithm do not produce very different solutions. In other words, the algorithm retains the ability to find quality solutions in different executions for almost every instance.

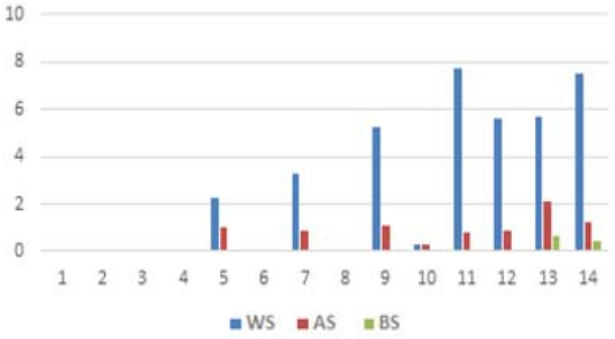

Fig. 4. The Gap comparison among WS, AS and BS by our proposed ACSA.

In order to compare our ACSA and simple ACS, instance Eil51 has been used. Fig. 5 shows the solutions obtained by the algorithms throughout their executions. In the figure, X- axis denotes iteration numbers and Y-axis denoted best solutions in an iteration throughout the execution of an algorithm. In this test, the stopping condition of an algorithm is equal to the number nodes of the instance. The figure shows that at the beginning of the iterations, ACS shows very good improvement in the solution, however, after 18 iterations, there is no improvement. So, there is a premature convergence of solution. On the other hand, ACSA is not only capable of improving the solution quickly in every iteration, but it is also capable of escaping the local optima and is achieving the BKS of 426 within 20 iterations.

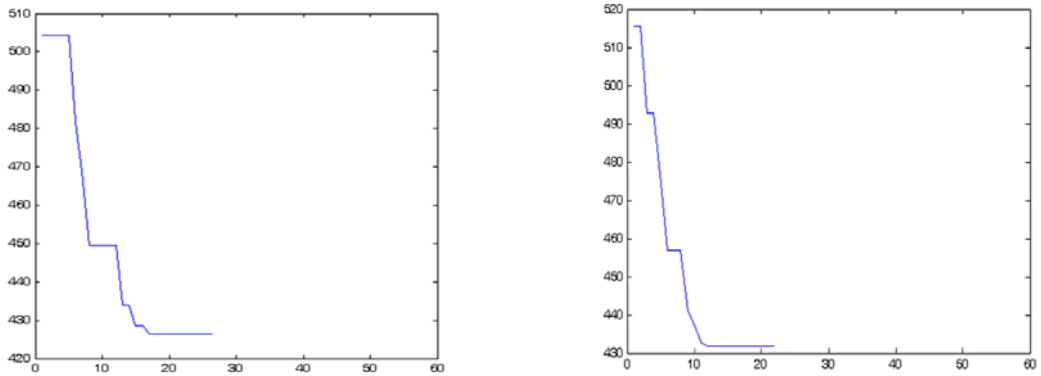

Fig. 5. Comparison between ACS (left) and ACSA (right) for Eil51.

We further, picked solutions of six instances obtained by the proposed algorithm that are reported in Table 2, and showed in Fig. 6. As previously mentioned in these instances, the two-dimensional coordinates of the depot and customers are presented as data, which is specified in these figures on the X and Y axes. As seen in these figures, for each instance, a tour is found in which each customer is met exactly once, and the salesman returns to the depot at the end. For these six instances, our algorithm could obtain excellent solutions and hence, achieved BKS values.

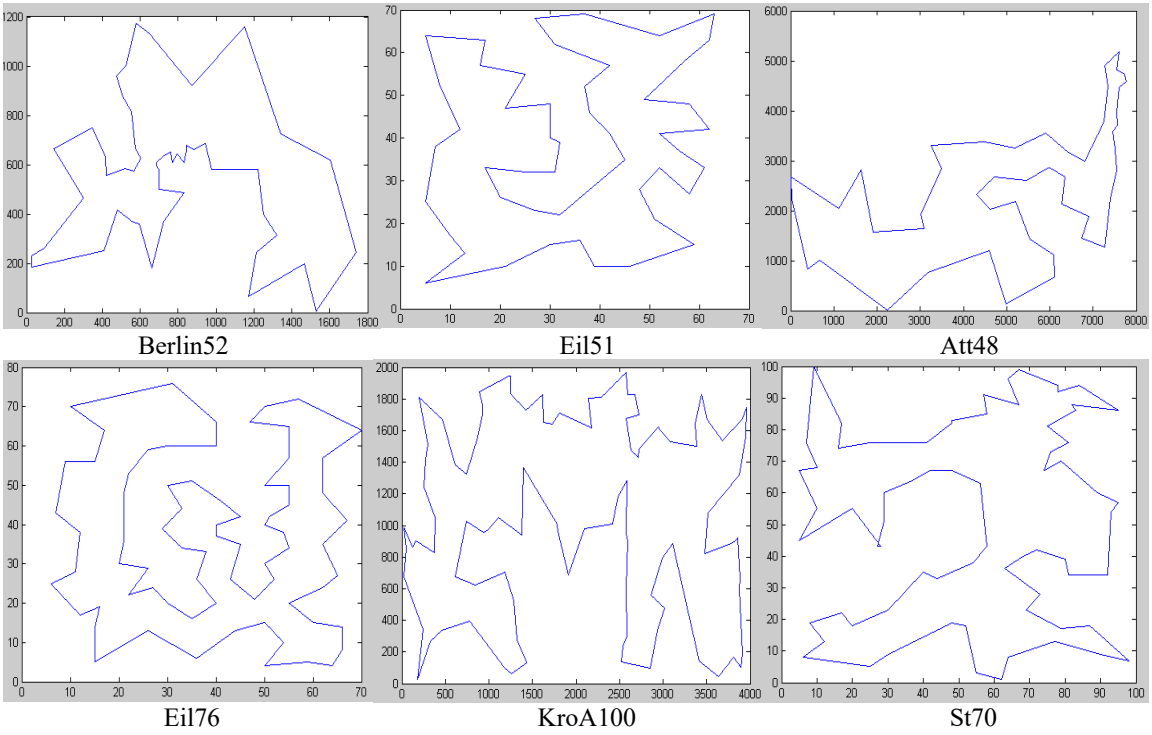

Fig. 6. Solutions obtained by our ACSA for some instances.

Table 3 presents the best solution found for 14 instances by four metaheuristic algorithms. The left column for each algorithm shows the best solution by the algorithm(s) and the right column indicates its percentage of improvement compared to the BKS (Gap). Columns 2 and 3 in the table present the results by BCO [25]. The algorithm belonging to particle swarm optimization (PSO) [26] is presented in columns 4 and 5. The effective ACO (EACO) has been presented in columns 6 and 7 [27]. Columns 8 and 9 contain the solutions of the ACSA. Further, the BKS has been reported in column 10 to compare solutions obtained by the algorithms. Finally, N/A means that the result is not reported for the instance.

Table 3: Comparing four algorithms for standard TSP instances.

Instance	BCO		PSO		EACO		ACSA		BKS
	Gap	Sol	Gap	Sol	Gap	Sol	Gap	Sol	
GR48	N/A	N/A	N/A	N/A	5072	0.52	**5046**	0.00	5046
ATT48	10661	0.31	N/A	N/A	10701	0.69	**10628**	0.00	10628
Eil51	428	0.47	427	0.23	429	0.70	**426**	0.00	426
Berlin52	N/A	N/A	**7542**	0.00	7591	0.65	**7542**	0.00	7542
ST70	N/A	N/A	N/A	N/A	686	1.63	**675**	0.00	675
Eil76	539	0.19	540	0.37	545	1.30	**538**	0.00	538
KroA100	21763	2.26	21296	0.07	21456	0.82	**21282**	0.00	21282
KroB100	22637	2.24	N/A	N/A	22304	0.74	**22141**	0.00	22141
Eil101	635	0.95	N/A	N/A	643	2.23	**629**	0.00	629
Lin105	15288	6.32	N/A	N/A	14703	2.25	**14379**	0.00	14379
KroA150	27858	5.03	N/A	N/A	26829	1.15	**26524**	0.00	26524
KroB150	26535	1.55	N/A	N/A	26439	1.18	**26130**	0.00	26130
KroA200	29961	2.02	29563	0.66	30111	2.53	29451	0.28	29368
KroB200	30350	3.10	N/A	N/A	30771	4.53	29506	0.23	29437
Average		2.22		0.27		1.49		0.04	

Comparing BCO with PSO, it is seen that out of four instances whose results are reported, PSO obtained better solutions than the solutions obtained by BCO for three instances whereas BCO obtained better solution for only one instance. So, PSO is better than BCO. Comparing PSO with EACO, it is seen that out of five instances whose results are reported, PSO obtained better solutions than the solutions obtained by BCO for all instances. So, PSO is better than EACO also. Comparing ACSA with PSO, it is seen that out of five instances whose results are reported, ACSA obtained better solutions than the solutions obtained by PSO for four instances whereas both obtained same solution for only one instance. For these instances, PSO could find solutions with less than 1% *Gap*. Despite being able to obtain the BKS for Berlin52, it failed to achieve these results for the remaining four instances. One can conclude that ACSA is capable to obtain better quality solutions compared to PSO and is capable to escape from the local optima. Comparing BCO with ACSA, BCO has produced close solution only for Eil76 and Eil101, while ACSA has obtained better solutions for the remaining nine instances. In general, BCO showed low performance and could not obtain best solution for any instance. Comparing ACSA with EACO, one can conclude that the ACSA can find better solutions that shows an acceptable enhancement of our algorithm. EACO is the weakest algorithm like BCO could not converge to best solution for any of fourteen instances. According to the performance of the algorithms from worst to best, one can list as: BCO, EACO, PSO and ACSA.

Table 4 provides additional comparison between our proposed ACSA and other seven algorithms. A total of twelve instances are considered in this table, ten of which are medium-sized and the other two are large-sized. Each algorithm has two columns best Gap (BG) and mean Gap (MG). In addition, each algorithm is executed ten times for every instance, and the best solution Gap of these ten solutions are shown in the first column and their average in the second column. The seven algorithms used for comparison with ACSA are as follows: PSO [28], multi-agent reinforcement learning algorithm (MARLA) [29], BCO [25], neural network (NN) [30], improved ACO with pheromone correction strategy (ACO-SEE) [31], generalized chromosome GA (GCGA) [32] and hybrid EAS (HEAS) [33].

Table 4: Comparing eight algorithms for standard TSP instances.

Instance	MARLA		PSO		BCO		NN		ACO-SEE		GCGA		HEAS		ACSA	
	BG	MG	BG	MG	BG	MG	BG	MG	BG	MG	BG	MG	BG	MG	BG	MG
Eil51	0.23	1.07	0.23	2.58	0.47	0.85	0.24	2.69	0.23	0.23	0.23	0.94	0.00	0.70	0.00	0.00
Berlin52	0.00	0.71	0.00	3.85	N/A	N/A	0.00	5.18	0.00	0.13	N/A	N/A	0.00	0.11	0.00	0.00
St70	0.15	1.80	0.00	3.42	N/A	N/A	N/A	N/A	0.00	1.36	0.00	0.44	0.00	0.00	0.00	1.04
Eil76	1.12	3.09	1.49	4.17	0.19	2.01	0.56	3.41	1.49	1.19	2.23	2.42	0.00	2.42	0.00	0.00
Pr76	0.86	2.38	0.11	3.82	N/A	N/A	N/A	N/A	0.11	2.62	0.14	0.72	0.00	3.18	0.00	0.00
KroA100	0.00	0.98	N/A	N/A	2.26	3.43	0.24	1.13	0.00	0.72	0.50	1.23	0.00	0.11	0.00	0.80
KroB100	0.43	1.27	N/A	N/A	2.24	3.10	0.91	2.35	N/A	N/A	0.24	1.81	N/A	N/A	0.00	0.00
Eil101	0.64	4.17	N/A	N/A	0.95	2.29	1.43	3.12	N/A	N/A	1.59	2.70	N/A	N/A	0.00	1.11
KroA150	1.27	3.10	N/A	N/A	5.03	6.39	0.58	3.14	N/A	N/A	1.40	2.92	N/A	N/A	0.00	0.76
KroB150	2.28	2.86	N/A	N/A	1.55	3.68	0.51	1.92	N/A	N/A	1.63	2.11	0.21	1.63	0.00	0.85
Rl1323	5.51	6.40	N/A	N/A	N/A	N/A	11.31	13.00	N/A	N/A	N/A	N/A	N/A	N/A	1.85	3.55
Fl1400	2.59	3.85	N/A	N/A	N/A	N/A	3.60	4.88	N/A	N/A	N/A	N/A	N/A	N/A	1.75	2.98

Comparing the best solutions achieved by the algorithms, one can conclude that BCO, NN and GCGA algorithms have the worst solutions, and they could obtain optimal solutions for only zero, one and one instances, respectively. Besides, the three algorithms MARLA, PSO and ACO-SEE have obtained better solutions in this view, and they have achieved 2, 2 and 3 optimal solutions respectively. Finally, HEAS and ACSA algorithms have obtained high-quality solutions compared to the other algorithms and have reached the BKS for more than half of the instances. Alternatively, HEAS algorithm is able to find the optimal solutions for six instances while our proposed ACSA algorithm could achieve the optimal solutions for ten instances. Therefore, one can conclude that ACSA algorithm is the best algorithm among these eight algorithms and is capable to obtain high quality solutions.

Another criterion based on which these eight algorithms can be compared is the average of the solutions in ten repetitions, so the proposed algorithm has found the best solutions. With more details, five algorithms - MARLA, PSO, BCO, NN, ACO-SEE and GCGA are different despite the number of best results, but in this regard, they were the same and none of them could achieve the optimal solutions. As in the previous comparison, HEAS and ACSA

algorithms have produced the best solutions. The HEAS algorithm could find optimal solution for only one instance, but our proposed ACSA, in this comparison, by obtaining the five best solutions, could find the best solutions compared to other algorithms. Therefore, in general, from all these comparisons reported in tables 2, 3 and 4, one can conclude that the improvements done in our proposed algorithm have produced the algorithm to produce better quality solutions and hence, our proposed ACSA has become one of the best algorithms.

5. Conclusion and Future Works

In this paper, an efficient ACS algorithm, called ACSA, is developed for solving the TSP. In our proposed algorithm, several important improvements were done. According to the obtained results in comparison against other algorithms, one can conclude that the productivity of the proposed algorithm is very good, and the algorithm has obtained high quality solutions. In addition, this efficient algorithm can be combined with other innovative constructive algorithms due to random solutions obtained at the beginning of the execution to start working with better solutions. To further increase the productivity of the algorithm, it can be combined with local search or some heuristic algorithms such as genetic algorithm, simulated annealing or tabu search. Finally, this algorithm can be used in other versions of the TSP, such as the general TSP or time-dependent TSP. The execution of these ideas would be postponed to our future articles. Further, we shall study this problem using other metaheuristic algorithms, like genetic algorithms [34], tabu search [35], etc.

Data Availability

The data used to support the findings of this study is available at the website: http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgement

The authors extend their appreciation to the Deanship of Scientific Research, Imam Mohammad Ibn Saud Islamic University (IMSIU), Saudi Arabia, for funding this research work through Grant No. (221412020).

References

[1] Sedighpour, M., Ahmadi, V., Yousefikhoshbakht, M., Didehvar, F., and Rahmati, F. (2014). Solving the open vehicle routing problem by a hybrid ant colony optimization, *Kuwait Journal of Science*, *41*(3), 139-162.

[2] Yousefikhoshbakht, M., Didehvar, F., and Rahmati, F. (2015). A mixed integer programming formulation for the heterogeneous fixed fleet open vehicle routing problem, *Journal of optimization in Industrial Engineering*, *8*(18), 37-46.

[3] Ahmed, Z.H. (2018). A hybrid algorithm combining lexisearch and genetic algorithms for the quadratic assignment problem, *Cogent Engineering*, *5(1)*, 1423743. doi: 10.1080/23311916.2018.1423743.

[4] Germs, R., Goldengorin, B., and Turkensteen, M. (2012). Lower tolerance-based branch and bound algorithms for the ATSP, *Computers and Operations Research, 39(2),* 291-298.

[5] Cordeau, J. F., Dell'Amico, M., Iori, M. (2010). Branch-and-cut for the pickup and delivery traveling salesman problem with FIFO loading, *Computers and Operations Research*, 37(5), 970-980.

[6] Ahmed, Z.H. (2011). A data-guided lexisearch algorithm for the asymmetric traveling salesman problem, *Mathematical Problems in Engineering. 2011*, 750968. doi: 10.1155/2011/750968.

[7] Mak, V., and Boland, N. (2007). Polyhedral results and exact algorithms for the asymmetric travelling salesman problem with replenishment arcs, *Discrete Applied Mathematics, 155(16),* 2093-2110. doi: 10.1016/j.dam.2007.05.014.

[8] Yousefikhoshbakht, M., Dolatnejad, A., Didehvar, F., and Rahmati, F. (2016). A modified column generation to solve the heterogeneous fixed fleet open vehicle routing problem, *Journal of Engineering, 2016,* 5692792. doi: 10.1155/2016/5692792.

[9] Ahmed, Z.H. (2010). Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator. *International Journal of Biometrics and Bioinformatics*, 6(3), 96-105.

[10] Niu, Y., Yang, Z., Chen, P., and Xiao, J. (2018). A hybrid tabu search algorithm for a real-world open vehicle routing problem involving fuel consumption constraints, *Complexity*, *2018*, 5754908, 12 pages. doi: 10.1155/2018/5754908.

[11] Ahmed, Z.H. (2014). A Simple genetic algorithm using sequential constructive crossover for the quadratic assignment problem, *Journal of Scientific and Industrial Research*, 73(12), 763-766. http://nopr.niscpr.res.in/handle/123456789/30018.

[12] Liu, Y.-H. (2008). Diversified local search strategy under scatter search framework for the probabilistic traveling salesman problem. *European Journal of Operational Research*, 191(2), 332-346. doi: 10.1016/j.ejor.2007.08.023.

[13] Yousefikhoshbakht, M., Malekzadeh, N., and Sedighpour, M. (2016). Solving the traveling salesman problem based on the genetic reactive bone route algorithm whit ant colony system. *International Journal of Production Management and Engineering*, *4*(2), 65-73. doi:10.4995/ijpme.2016.4618.

[14] Marinakis, Y., Marinaki, M., and Dounias, G. (2011). Honey bees mating optimization algorithm for the Euclidean traveling salesman problem. *Information*

Sciences, 181(20), 4684-4698. doi: 10.1016/j.ins.2010.06.032.

[15] Wang, Y.-T., Li, J.-Q., Gao, K.-Z., and Pan, Q.-K. (2011). Memetic algorithm based on improved Inver–over operator and Lin–Kernighan local search for the Euclidean traveling salesman problem. *Computers and Mathematics with Applications*, 62(7), 2743-2754. doi: 10.1016/j.camwa.2011.06.063.

[16] Cochrane, E.M., and Beasley J.E. (2003). The co-adaptive neural network approach to the Euclidean travelling salesman problem, *Neural Networks, 16(10),* 1499-1525. doi: 10.1016/S0893-6080(03)00056-X.

[17] Ahmed, Z.H., Hameed, A.S., and Mutar, M.L. (2022). Hybrid genetic algorithms for the asymmetric distance-constrained vehicle routing problem, *Mathematical Problems in Engineering*, *2022*, 2435002, 20 pages. doi: 10.1155/2022/2435002.

[18] Hameed, A.S., Mutar, M.L., Alrikabi, H.M.B, Ahmed, Z.H., Abdul–Razaq, A.A., and Nasser, H.K. (2021). A hybrid method integrating a discrete differential evolution algorithm with tabu search algorithm for the quadratic assignment problem: a new approach for locating hospital departments, *Mathematical Problems in Engineering*, *2021*, 6653056, 21 pages. doi: 10.1155/2021/6653056.

[19] Ahmed, Z.H., and Yousefikhoshbakht, M. (2023). A hybrid algorithm for the heterogeneous fixed fleet open vehicle routing problem with time windows, *Symmetry*, 15(2), 486.

[20] Al-Furhud, M.A., and Ahmed, Z.H. (2020). Experimental study of a hybrid genetic algorithm for the multiple travelling salesman problem, *Mathematical Problems in Engineering, 2020*, 3431420, 13 pages. doi: 10.1155/2020/3431420.

[21] Yousefikhoshbakht, M. (2021). Solving the traveling salesman problem: a modified metaheuristic algorithm. *Complexity*, *2021*, 6668345, 13 pages. doi: 10.1155/2021/6668345.

[22] Ahmed, Z. H. (2013). A hybrid genetic algorithm for the bottleneck traveling salesman problem, *ACM Transactions on Embedded Computing Systems (TECS)*, 12(1), 1-10. doi: 10.1145/2406336.2406345.

[23] Baltierra, S., Valdebenito, J., and Mora, M. (2022). A proposal of edge detection in images with multiplicative noise using the ant colony system algorithm, *Engineering Applications of Artificial Intelligence*, 110, 104715. doi: 10.1016/j.engappai.2022.104715.

[24] Dorigo, M., and Gambardella, L. M. (1997). Ant colony system: A cooperative learning approach to the traveling salesman problem, *IEEE Transactions on Evolutionary Computation, 1(1),* 53–66.

[25] Wong, L.-P., Low, M.Y.H., and Chong, C.S. (2008). A bee colony optimization algorithm for traveling salesman problem, *2008 Second Asia International Conference on Modelling & Simulation (AMS),* 818-823. doi: 10.1109/AMS.2008.27.

[26] Zhong, W., Zhang, J., and Chen, W. (2007). A novel discrete particle swarm optimization to solve traveling salesman problem, *2007 IEEE Congress on Evolutionary Computation, 3283-3287*. doi: 10.1109/CEC.2007.4424894.

[27] Yousefikhoshbakht, M., and Darani, N.M. (2019). A combined metaheuristic algorithm for the vehicle routing problem and its open version, *Journal of AI and Data Mining, 7(1),* 169-179. doi: 10.22044/jadm.2018.7116.1840.

[28] Geem, Z.W., Kim, J.H., and Loganathan, G.V. (2001). A new heuristic optimization algorithm: harmony search, *Simulation, 76(2),* 60–68. doi: 10.1177/003754970107600201.

[29] Alipour, M.M., and Razavi, S.N. (2015). A new multiagent reinforcement learning algorithm to solve the symmetric traveling salesman problem, *Multiagent and Grid Systems, 11(2)*, 107-119. doi: 10.3233/MGS-150232.

[30] Masutti, T.A.S., and de Castro, L.N. (2009). A self-organizing neural network using ideas from the immune system to solve the traveling salesman problem, *Information Sciences, 179(10)*, 1454–1468. doi: 10.1016/j.ins.2008.12.016.

[31] Tuba, M., and Jovanovic, R. (2013). Improved ACO algorithm with pheromone correction strategy for the traveling salesman problem, *International Journal of Computers, Communications and Control, 8(3),* 477-485.

[32] Wu, C., Liang, Y., Lee, H.P., and Lu, C. (2004). Generalized chromosome genetic algorithm for generalized traveling salesman problems and its applications for machining, *Physical Review E, 70(1),* 016701. doi: 10.1103/PhysRevE.70.016701.

[33] Jaradat G.M. (2018). Hybrid elitist-ant system for a symmetric traveling salesman problem: case of Jordan, *Neural Computing and Applications, 29,* 565-578. doi: 10.1007/s00521-016-2469-3.

[34] Ahmed, Z.H., Al-Otaibi, N., Al-Tameem, A., and Saudagar, A.K.J. (2023). Genetic crossover operators for the capacitated vehicle routing problem, *Computers, Materials & Continua*, 74(1), 1575-1605. doi: 10.32604/cmc.2023.031325.

[35] Ahmed, Z.H., and Yousefikhoshbakht, M. (2023). An improved tabu search algorithm for solving heterogeneous fixed fleet open vehicle routing problem with time windows, *Alexandria Engineering Journal, 64(1),* 349-363. doi: 10.1016/j.aej.2022.09.008.