Hindawi
Complexity
Volume 2021, Article ID 6668345, 13 pages
https://doi.org/10.1155/2021/6668345

WILEY | Hindawi

Research Article

Solving the Traveling Salesman Problem: A Modified Metaheuristic Algorithm

Majid Yousefikhoshbakht

Department of Mathematics, Faculty of Sciences, Bu-Ali Sina University, Hamedan, Iran

Correspondence should be addressed to Majid Yousefikhoshbakht; yousefikhoshbakht@gmail.com

Received 20 December 2020; Revised 5 February 2021; Accepted 10 February 2021; Published 19 February 2021

Academic Editor: Dimitri Volchenkov

The traveling salesman problem (TSP) is one of the most important issues in combinatorial optimization problems that are used in many engineering sciences and has attracted the attention of many scientists and researchers. In this issue, a salesman starts to move from a desired node called warehouse and returns to the starting place after meeting n customers provided that each customer is only met once. The aim of this issue is to determine a cycle with a minimum cost for this salesman. One of the major weaknesses of the PSO algorithm in the classical version is that it gets stuck in local optimizations. Therefore, in the proposed algorithm, called MPSO, the best solution in the current iteration is also used in the movement step. In addition, a variety of local search algorithms are provided that are used when better answers are generated than before. Also, a new method for moving the particle towards the best particle is presented, which, in addition to probably increasing the quality of the new answer, prevents the premature convergence of the algorithm due to consideration of the concept of random. The results evaluated with the results of several metaheuristic algorithms in the literature show the efficiency of the MPSO algorithm because it has been able to achieve excellent solutions in most of these instances.

1. Introduction

The aim of optimization is to find the best acceptable answer, considering the limitations and needs of the problem. For a problem, there may be different answers that are defined as the objective function to compare them and select the optimal solution. The choice of this function depends on the nature of the problem. For example, travel time or cost is one of the common goals of optimization of the transportation network. However, choosing the appropriate objective function is one of the most important optimization steps. Sometimes, in optimization of multiple objectives simultaneously, such optimization problems, which involve multiple objective functions, are called multiobjective problems. The easiest way to deal with these problems is to form a new objective function in the form of a line composition of the main objective functions, in which the effect of each function is determined by the weight assigned to it. Each optimization problem has a number of independent variables that are called design variables that are represented by the vector $\boldsymbol{x}$ with size n. The purpose of optimization is to determine the design variables in such a way that the objective function is quantitative or optimal. Different optimization problems are divided into two categories:

(a) Optimization problems without constraints: in these problems, the goal is to make the objective function the most or least without any limitations on the design variables

(b) Optimization problems with constraints: optimization, in most applied problems, is carried out according to limitations, and regarding the behavior and performance of a system, behavioral limitations and limitations in the physics and geometry of the problem are called geometric or lateral limitations

Equations representing limitations may be equal or unequal, which in each case, the optimization method is different. However, the limitations define an acceptable area

in the design. Traveling salesman problem (TSP) is highly considered due to its different applications in testing the efficiency of new algorithms and conversion of other problems to this algorithm. In this seemingly simple issue, a salesman moves from the warehouse and must return to the warehouse after meeting all customers once. The objective is to reduce the mileage as much as possible. The TSP issue can be examined from different perspectives. For example, this problem is investigated to measure the efficiency of new algorithms because although this problem has a hard structure in complexity theory, most algorithms are easily used on it, and thus, the efficiency of algorithms can be investigated compared to other algorithms. Also, other issues can be turned into problems and solved. Thus, the steps to solve some problems can be shortened by this procedure, and using highly efficient algorithms to solve the TSP problem, the main problem can be solved with more quality. For example, these issues include vehicle routing problem [1], aircraft crew timing [2], hybrid Chinese postman issue [3], workshop arrangement [4], mission design for autonomous mobile robots [5], and computer connectivity [6]. However, the applications of this issue are not limited to those mentioned and can be referred to [7] for full information.

The methods for solving this problem like other operation research can be classified as exact and heuristic algorithms. One of the obvious and exact methods for solving this problem is finding all the permutations of the nodes and their associated tours and considering the best result in all instances as the optimum solution. To achieve this goal, we need $(n-1)!/2$ operations for undirected graphs and $(n-1)!$ operations for the directed graph. Therefore, if the number of problem data increases in these problems and the size of the problem becomes larger, then the efficiency of these algorithms for solving such problems decreases so much that after a certain amount of data, the algorithm cannot achieve the optimal solution. In other words, these algorithms need unacceptable times to solve these problems, and because the solution of these practical problems must be done in an acceptable time, the use of these algorithms in these problems is not recommended.

Due to the weaknesses in exact, heuristic, and metaheuristic algorithms, in this paper, a modified PSO algorithm for TSP, called PPSO, is presented. In addition, since the combination of heuristic and metaheuristic algorithms in literature has had good results, the proposed algorithm is combined with several local search algorithms. On the contrary, several standard examples of the problem have been considered to test the efficiency of the algorithm, and for each example, the algorithm has been tested ten times, and the worst, average, and best solutions have been presented. Also, the quality of the best solutions obtained by this method has been compared with other metaheuristic algorithms for a suitable range of examples. The results show that the algorithm has a very good efficiency for solving problems and can obtain near-optimal solutions at the acceptable time.

In the next sections of this article, firstly, the related works and the classic PSO are presented in Section 2. In Section 3, the proposed method is described and explains how the use of modifications in this algorithm and some local search methods are effective in increasing the efficiency of the algorithm. Then, in Section 4, the results of the proposed algorithm are compared with other metaheuristic algorithms, and finally, the conclusions and future works are described in Section 5.

2. The Related Work and the Classic PSO

Like most combinatorial optimization problems, TSP solution methods are divided into two categories: exact and heuristic algorithms. In exact algorithms such as Lagrangian relaxation [8], branch and bound [9], and pseudoallocation [10], which are mostly used for almost small size problems, the optimal solution to the problem is obtained, despite the fact that a lot of time has to be spent to get this answer. In this issue, the number of solutions possible for n customers, regardless of the warehouse or starting place, is equal to the number of permutations of 1 to n. In other words, the objective of the problem is to find a Hamiltonian cycle with the lowest cost because the number of solutions to this issue is equal to n!. By increasing the customers of the problem, the number of solutions can grow sharply, and it is no longer possible to easily, at an acceptable time, compare all the solutions and achieve the optimal solution. But if the dynamic programming method is used to solve this problem, then the time order will be algorithmic, which is an exponential order. It should be noted that despite the fact that the exponential order is a very bad time, it is still much better than the factorial order. For this reason, several decades ago, heuristic algorithms were proposed for TSP which can reach the solutions in a short time.

These algorithms were able to solve the unacceptable CPU time of exact algorithms because today's time solving is the most important factor for solving a routing problem. In the application of these problems in daily life used for the delivery of goods, it is better that, in some issues, especially in perishable goods, they reach at a specified time, and the cost of routing can be withered. The use of these algorithms in those years was much higher, but over time the results of these algorithms showed that despite the very good time of solving these algorithms, the quality of these algorithms is not of acceptable quality, especially when the size of the problem grows, which occurs in most real problems [11–15]. In other words, the quality of the obtained solution in these algorithms is the victim of time, and as such, these algorithms are trapped in local optimizations and cannot get a better solution due to lack of time and proper solution [16]. So over the last three decades, scientists and researchers have tried to come up with methods that have an efficient solution to find high-quality solutions; however, it needs much more time to be solved. These efforts led to the use of the concept of random in methods, which, for example, is known in the simulated annealing algorithm compared to the classical local search algorithm because in this algorithm, a worse solution is accepted than the best solution obtained by the probable. This solution was used in other new algorithms called metaheuristic and caused the quality of the obtained solutions in these methods to increase compared to heuristic

algorithms. So, these algorithms, despite taking more time to solve, are needed to produce better solutions than heuristic algorithms with appropriate solutions and make proper use of the concept of accident. Of course, it should be noted that the very good property of metaheuristic algorithms is that the time of implementation of the algorithm is user-dependent, and based on the acceptable time in each problem, the algorithm is implemented [17–20]. Therefore, there is a direct relationship in these algorithms for the run time and the quality of the obtained solutions, which can change according to the user's diagnosis.

A modified elite genetic algorithm (GA), called SWAP_GATSP, is proposed for efficiently solving the TSP issue [21]. In this algorithm, novel deterministic operation injected in the typical structure of elitist GA, called knowledge-based multiple inversion are used. This kind of variation helps in exploring the search space efficiently and prevents the GA from getting stuck in the local optima. Note that this is an upgraded version of simple inversion mutation. But, it cannot be called the mutation operator, as it is a decisive process not a random one. Also, modified order crossover and knowledge-based neighborhood swapping are used for increasing quality of GA. If k, m, and n are the number of generations, number of the population size, and the number of customers in the considered instance, the time complexity of the algorithm is O (k•m•n).

In paper [22], a modified PSO algorithm called C3DPSO was presented to efficiently solve the TSP problem, which achieved better results on standard examples than the previous three modifications of the PSO algorithm. This algorithm changed the formula for updating particles using the mutation coefficient called c3 so that causing this algorithm could create a perfect balance between intensification and diversification mechanisms. Also, the modifications made in the algorithm had an effective impact on both the run time and the quality of the answers. Also, they cause the algorithm not to have premature convergence and to adequately search for the problem space. On the contrary, the parameters of this algorithm were not adjusted by any method, and in addition to the three improved PSO algorithms, the algorithm was compared with an ant colony optimization. The results showed the efficiency of the proposed algorithm on the examples taken from the TSPLIB library.

For the first time, a bee optimization algorithm based on their foraging food for the TSP problem was presented in the paper [23], and the results were compared with previous versions of the bee optimization algorithm. It should be noted that this algorithm had some differences compared to previous versions of the bee method. For example, the bee hive in the problem had the same distances as all customers, and no local search algorithm was used, and the bees did not have the memory to store the number of bees seen in each edge, and to build the answer, the bees used the arc fitness and the distance between the nodes. These modifications led to a good efficiency algorithm for solving 12 standard problems from 48 nodes to 200 nodes. In addition, in comparison with other metaheuristic algorithms, excellent solutions were obtained by this algorithm.

The study in [24] uses a multiagent reinforcement learning method to solve the TSP problem as one of the most important routing issues. This multiagents' algorithm works on the basis of learning enhancement so that despite the independency of each factor and memory, it causes optimization problems to be solved. It should be noted that the lack of communication of factors does not reduce the quality of the answers, but ultimately their hidden communication leads to the final solution of the problem. Due to the inefficiency of this algorithm in the intensification mechanism, an iterated local search algorithm is used to further improve the answers in this algorithm in order to investigate more neighbors in susceptible areas. The use of several TSP problems from the TSPLIB library shows the efficiency of the proposed algorithm compared to other algorithms.

In [25], an elite hybrid algorithm of ACO is presented, that uses an innovative process with an external memory structure. This memory is composed of a variety of high-quality answers, causing a balance between intensification and diversification mechanisms. These modifications increase the efficiency of the algorithm in order to escape from local optimal points and achieve high-quality solutions. To test the efficiency of this new algorithm, an NP-hard problem, such as TSP, which has a simple concept, is considered to determine the compatibility and efficiency of the algorithm along with its effectiveness. Due to the multifactor structure of this algorithm and the weakness in the local search of the answers, a repetitive local search algorithm is used to intensify the search around high-quality answers in order to obtain better answers. Finally, as a case study, two problems of finding the best tour in Jordan are considered, and the results showed that the algorithm has high quality of performance, for example, for these methods, memetic algorithm [26], discrete tree-seed algorithm [27], simulated annealing [28], ant colony optimization [29], and genetic algorithm [30]. Although the stochastic search of metaheuristic algorithms causes the problem to be investigated in different directions and in case of falling in the local optimal, the algorithm can escape from the local optimal by using these structures, but on the contrary, this structure causes the time of these algorithms to increase compared to heuristic methods. It should be added that, in metaheuristic algorithms, the accuracy of the obtained solutions and the speed of reaching have an inverse relationship with each other.

As it is known from this form, the speed of accuracy of the solutions at the beginning of the algorithm is very good, and in a short time, the quality of the solutions is very desirable, but whatever passes from the beginning of the algorithm, this desirability decreases and causes the quality of the solutions to remain almost constant. Therefore, due to the high volume of calculations, variables, and parameters, the number of algorithm repetitions increases sharply, and the algorithm cannot improve the quality of the solutions at the right speed.

In hybrid algorithms composed of metaheuristic and heuristic algorithms, it has been tried to use the advantages of each algorithm appropriately. For example, metaheuristic algorithms have a good ability to search globally and can

search for the problem space at an acceptable time and identify susceptible areas. Therefore, in hybrid algorithms, these methods can be used to search globally and find good solutions. On the contrary, heuristic methods have good local search efficiency and can properly examine the neighborhoods of a solution. Therefore, in some hybrid algorithms, these methods can be used when the algorithm has been able to identify high-quality solutions, but it requires the local search to find the better solution in the neighborhoods of these solutions. As an example of these methods, which have attracted a lot of attention today, we can refer to the hybrid algorithms' sweep, genetic algorithm (GA) and the nearest neighborhood [31], ant system algorithm and scattered search [32], and particle swarm optimization (PSO) and local search [33].

The particle swarm optimization (PSO) algorithm is a population-based random optimization method developed by Kennedy and Eberhart in 1995 [34], inspired by the social behavior of bird overcrowding and fish farming. Of course, this was just the beginning, and extensive research was conducted to improve this method, which led to stronger versions of this method provided by many authors [35–37] that the reader could see a summary of the development, improvement, and applications of this algorithm in [38]. To get a proper understanding of this method, consider a group of birds looking for food in an environment. It is notable that none of them know the place of food, but at each stage, they know their distance to the place of food. Accordingly, the best approach to finding food is to follow the nearest bird to food. PSO tries to use this behavior to solve optimization problems [39] and offers a reliable way to solve these problems. In general, the PSO algorithm has simple concepts and is rooted in artificial life and computational intelligence. Also, this algorithm is similar to evolutionary computational techniques such as the genetic algorithm (GA), but compared to GA, it does not have intersection and mutation operators [40]. Because the PSO method has few parameters, its implementation is simple and effective and applicable to solve various problems. Finally, its main application is for solving unconstraint problems, but it can also be used for constraint problems using the fines' method.

As mentioned earlier, the PSO algorithm simulates bird swarm behaviors. Imagine the following scenario: a group of birds are accidentally exposed to food in one area. There is only one piece of food in the search area. Not all birds know where the food is, but they know how much food per iteration. So, what is the best strategy for finding food? The solution is to follow a bird that is closer to food. PSO designers adapted this scenario and used it to solve optimization problems. In PSO, each solution is a "particle" in the search space. All particles have proportional values that are evaluated by the proportionality function for optimization and have speeds that drive particle flight. Particles flow through the problem space with optimal particles.

The PSO algorithm begins with a group of random particles (solutions) and then searches by updating generations. In each iteration, each particle is updated with two "best" values: the first is the best solution (objective function) that has been achieved so far (the value of the objective is also stored). This value is called *pbest*. Another is the "best" value ever achieved by every particle in the population. This is the best global value of the best and is called *gbest*. In other words, *gbest* can be considered the best *pbest* in the whole group. This process continues until the algorithm stops (Figure 1). It should be noted that the condition of stopping in this algorithm is to rate birds to zero or reach the number of repetitions considered. Now, the speed and position are updated with the equation as follows:

$$v[t+1] = v[t] + c1 * \text{rand1}(t) * ((\text{pbest}[t] - \text{present}[t]) + c2 * \text{rand2}(t) * (\text{gbest}[t] - \text{present}[t])), \quad (1)$$

$$\text{present}[t+1] = \text{present}[t] + v[t+1], \quad (2)$$

where $c1$ and $c2$ are the constant learning parameters that determine the effect of *gbest* and *pbest* in the next position, *rand1* and *rand2* are random numbers in the range [0, 1], *present* is the present position of each bird, and *v[t]* is the movement speed of each bird in stage *t*.

In Figure 2, the movement of the particle, in the possible space, is shown.

3. The Proposed PSO

Although metaheuristic algorithms, such as PSO, have excellent performance than heuristic algorithms, they fall in premature convergence when used for complex problems in daily life. In other words, this concept causes particles, which show the role of feasible solutions to the problem, to be stuck in local optimal points, and spend a long time without improvement. Therefore, the quality of the obtained answers cannot grow continuously and achieve quality solutions. For this reason and to overcome premature convergence, several modifications are presented to improve the PSO algorithm in the literature [41–45]. In the PSO algorithm, *pbest* and *gbest* are considered, while in the proposed algorithm, called MPSO, the best current answer is also considered as *gcbest*. The method is that the solutions *gbest* and *gcbest* have two coefficients of *a1* and *a2*, respectively, in which the sum of these two variables is equal to 100%. The values of these two coefficients change over time as the algorithm runs, as described in the following. Finally, three local search algorithms are considered for intensification to further improve the answers if *gbest* or *pbest* is updated. The steps of the proposed algorithm are described as follows.

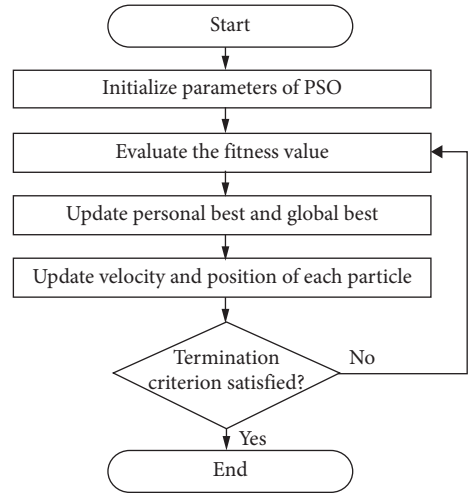

Figure 1: The steps of the classic version of PSO.

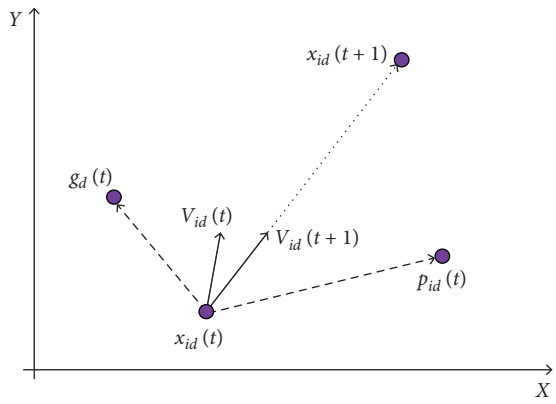

Figure 2: Moving particle in the PSO algorithm.

3.1. Initialization. In the TSP problem, each particle has a possible solution and indicates a permutation of the TSP customers. At first, the algorithm starts with m random solutions. Then, according to the objective function, the objective value is calculated for each particle based on the sum of all arcs traveled by the salesman. Besides, the $m+2$ variables are considered with the objective function's values of infinite positive, whose m is considered to be the best known locations for particles. Also, the other two variables are *gbest* and *gcbest* in each iteration.

3.2. Diversification and Intensification Mechanisms. Although the PSO algorithm had better performance than heuristic and some metaheuristic algorithms at the time it is presented, with the expansion of urbanization and creating more problematic problems in industry and services, this classic version of the algorithm was unable to solve these problems optimally. Therefore, it is necessary to make modifications to the algorithm to increase the efficiency. For example, one of the weaknesses is the premature convergence, which causes the algorithm to be limited to a range in the problem space without paying attention to the global search and so-called falling in the local optimum point [46, 47]. In other words, in the classical mode, there is no proper balance between a global search of the problem space and a suitable local search when finding susceptible areas. On the contrary, the factors that have caused the metaheuristic algorithms to be highly considered nowadays are as follows: the large size of everyday problems that managers face today, the high importance of solving problems that customers or users pay attention to using these algorithms, and the stability of the answers obtained for various problems considered for both managers and customers. Therefore, the aim is to provide sustainable algorithms that can achieve large-sized problems in industry and services at a reasonable time. In other words, the answers obtained by different performances do not have much tolerance and can be solved with a maximum error.

For balancing between global and local search mechanisms in the proposed algorithm, the best current solution at each iteration, called *gcbest* is also considered. This algorithm works by considering two variables $0 <= a1$ and $a2 \leq 1$ for gbest and *gcbest*, respectively, so that %alpha $<= a1 \leq$ %beta and $a2 = 1 - a1$. At the beginning of the algorithm, which requires the algorithm to pay more attention to the global search, $a1 =$ %alpha is considered, resulting in $a2 =$ %beta. Now, during the algorithm and based on the number of implementations of the main body of the algorithm, the value $a1$ gradually reaches from %alpha to %beta. Therefore, a random number between 0 and 1 is generated, and if this value is smaller or equal to $a1$, *gbest* will be considered, and otherwise, *gcbest* will be selected. Now, *pbest* and one of these two particles are considered, and the corresponding particle moves as a linear combination of them. The method presented in this paper is described by using an example. Suppose that the random number will be created so that *gcbest* is selected, *pbest* = 0 3 2 1 5 6 8 7 4 8 0, *gcbest* = 0 4 5 2 1 3 6 8 7 0, and particle = 0 3 1 4 5 8 7 2 6 0. Then, two *pbest* and *gcbest* answers are searched so that the customers in a joint burst are found between them and transferred to the new particle answer. In this example, the values of "2 1" and "6 8 7" are common between the two *pbest* and *gcbest* solutions. As a result, the particle response is modified using the insert algorithm in which customers can be created. So, particle = 0 3 2 1 4 5 6 8 7 2 0 is generated. The reason for using this method is that because two *pbest* and *gcbest* answers are among the best answers in the population, so during the algorithm, pathways are created that may not be the best, but the same routes of a few customers are most likely the best routes between those customers, and it is better to create in the particle as well.

3.3. Updating gbest and pbest. After all the particles have moved using the previous step, *pbest* of each particle and *gbest* may be changed. In other words, for each particle, if the new answer has a better value than the corresponding *pbest*, this value is updated. In addition, after changing all *pbests*, if a value is found in them that has a better value than *gbest*, this value will also be updated. It should be noted that, at the end of the algorithm, the value of *gbest* and its objective

function are presented as the best answer and value of the problem.

3.4. Local Search Algorithms. In this step, four iterated local search algorithms are described, as shown in Figure 3. These methods are used when *pbest* or *gbest* is updated. It is important to note that the reason for using these algorithms at this stage is that when a better solution is found, there may likely be better solutions around it, which the algorithm can achieve them [14, 48]. In the inset move, a specific customer moves to another location on the same path, and in the exchange move, two customers are selected and change their positions. Besides, the 2-Opt move works by removing two edges from the cycle and re-connecting the two arcs in another way. Finally, in 3-inverse move, three customers are selected, and their order is reversed in the desired answer. It is necessary to note that each movement is accepted by the algorithm when it leads to a better solution to the relevant method.

3.5. The Stop Condition. Like other metaheuristic algorithms, the stop criterion for the end of the algorithm is investigated in this step. If this condition has been achieved, *gbest* and its objective function will be considered as the best solution and value of the problem. Otherwise, the algorithm process will be repeated again until the stop condition of the algorithm is established. The pseudocode of the algorithm is shown in Figure 4.

3.6. Computational Complexity of PPSO. In order to obtain the computational complexity of the proposed algorithm, it should be assumed that the number of iterations of the main body of the algorithm is n, and the initial population of particles is equal to m. Now, it is enough to consider the part of the code where most of the operations are performed, which is why the rest of the code can be ignored. This part is the using local search algorithms. The complexity of the main loop is O (n). Also, all particles must be checked to find out which particle's *pbest* has changed, so there is O (m) in this section. Now, the 3-inverse algorithm is more complex than other local search algorithms, and because it has to choose 3 customers from m, the complexity of this part is O (m^3). As a result, the complexity order of the proposed algorithm is of O (nm^4).

4. Computational Results

In this section, the results of the proposed algorithm are presented for two categories of examples. Also, the results obtained by the proposed algorithm are compared with some of the most famous results of metaheuristic algorithms in the second set of instances. It should be added that all the code of the proposed algorithm is written in Matlab 7, and the computer on which these programs have been implemented is a laptop with Intel(R) Core(TM) i3 CPU 2.53 GHz and 4 GB of memory. In this section, the parameter settings are discussed, and then, our results are presented.

4.1. Parameter Settings. The solutions produced by the MPSO, like every metaheuristic algorithm, were dependent on the seed used to generate the sequence of pseudorandom numbers and on the different values of the parameters of the algorithm. Only four parameters exist in our proposed algorithm, and the values of these directly or indirectly affect the quality of the final solution. A parameter setting procedure is necessary to reach the best balance between the quality of the solutions obtained and the required computational attempt. We should mention that there is no way of defining the most effective values of the parameters. Therefore, in this paper, similar to many others, they are selected after thorough testing; we understand that the most influential experimental parameters in MPSO that directly affect the quality of the final solution are the number of populations and the stop condition.

Thus, all of the parameters' values have been determined on the Ch150 instance. To find the best values of the parameters in the proposed algorithm, 15 combinations of parameters are selected, and for each test, the Ch150 example is tested ten times, and the average of the answers is considered. Now, considering the answers to these examples and using the Taguchi method, the best answers for the parameters are obtained. The results on the next section confirm that our parameter setting worked well. The parameters of the MPSO and their values in different configurations are congregated in Table 1.

4.2. Our Results. Table 2 shows 40 standard instances for comparing the classical PSO algorithm with the proposed MPSO method, which has a suitable range of customers from 29 to 575. In this table, n represents the number of clients in each instance in column 2, and the best results obtained by all algorithms (BKS) in the literature are given in the last column. In addition, the best solution (BS), worst solution (WS), and the average solution (AS) for both algorithms are shown in other columns for 10 times the execution of each instance. Also, for each algorithm, the value of the objective function (cost) and the relative difference to BKS (Gap) for each example are given. These instances are available at the URL http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.

If the best solution found by one of the two algorithms is shown with $c(s^*)$, Gap compared to BKS is obtained from formula (3). It should be noted that the zero value in this formula means that the algorithm has been able to achieve the BKS answer in the literature:

$$100 \times [c(s^*) - \text{BKS}]/\text{BKS}. \quad (3)$$

It should be noted that the a Gap value can be a negative, zero, or positive number so that obtaining a negative Gap number for an example shows that the algorithm was able to produce high quality answers and increase the quality value of BKS. In addition, a value of zero indicates that the algorithm was able to obtain an answer similar to the best algorithms for that example, and a positive value indicates a worse response than BKS, which occurs in most algorithms.

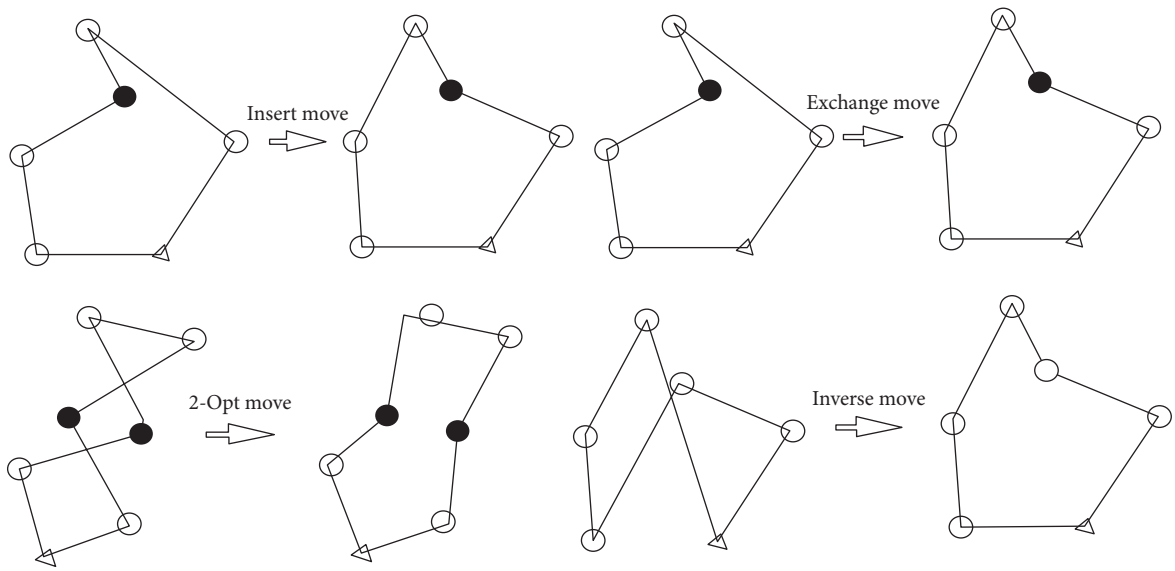

Figure 3: The insert, exchange, 2-Opt, and 3-inverse moves.

Input = n, m, and customer characteristics.
1- Generate m random solutions for the TSP problem and find their objective functions.
2- Find the *pbest*, *gcbest* and gbest solutions.
3- Repeat
(i) Move all particles toward to the *pbest*, *gcbest* or *gbest* solutions according to the procedure.
(ii) For each particle, update *pbest*, then update *gbest*.
(iii) If the *pbest* or *gbest* is updated, all local search algorithms are used.
4- Until the stop condition is satisfied.
Output = The *gbest* with its objective function.

Figure 4: Steps of the proposed algorithm.

Computational results on 40 examples of Table 2 show that the classic PSO algorithm has been able to achieve BKS in WS, AS, and BS modes in only 7 small instances, but when the number of nodes increases, it cannot achieve BKS solutions with the same quality and loses its efficiency. Also, in the three examples Arr48, Eil51, and KroA100, although the algorithm in the BS mode has been able to achieve the BKS solutions, the BKS solutions have not been obtained in WS and AS modes. On the contrary, in the remaining 30 examples, which have a customer number of 76 and above, the algorithm in the WS mode has minimum and maximum errors from 3.34 to 19 percent. Besides, for BS and AS modes, it has better solutions so that, for the first criterion, it has minimum and maximum errors of 1.56 to 8 percent, and for the second criterion, it has errors between 0.67 and 14.06 percent. Therefore, it can be concluded that the algorithm in the classical mode is not of good quality and does not have the necessary stability to use routing problems. Also, the results of the proposed MPSO algorithm show that the algorithm has a very excellent performance, and in 16 examples, the last of which was Bier127, it has been able to have a very good stability and achieve BKS solutions in all three criteria. In other words, BKS answers in 14 other examples have been obtained by this algorithm for the BS mode, and in this case, in only 10 of the 40 examples, it has not been able to achieves to the BKS solutions that for a metaheuristic algorithm compared to other algorithms, and as you will see in the next table, it is a very high-quality solution. In this case, the Gaps change between 0.22 and 4.34, while for average AS answers, its values change between 0.13 and 8.59. Also, in the WS mode, the algorithm has the minimum and maximum Gaps from 0.33 to 9.19.

To compare the answers obtained in Table 2, Figures 5 and 6 have been used so that in Figure 5, the average of Gaps is shown for 40 instances, while in Figure 7, the average of the results obtained by the two algorithms for the AS mode is presented to compare their stabilities. It should be noted that, in Figure 5, the average Gaps for BS, AS, and WS are ascending, but usually because the best solutions of metaheuristic algorithms are compared in the BS mode, it can be seen that the average for PSO and MPSO algorithms is 3.28 and 0.55, respectively, which shows that the modifications of the classic version have increased the quality of the algorithm and obtained excellent answers. On the contrary, Figure 6 shows that only the stability of the solutions for the first four examples is the same for the two algorithms, both of which were able to achieve the best answers, but for the next 36 examples, the MPSO algorithm achieved the better solutions.

Table 1: Parameter settings for the MPSO method.

Parameter	Description	Candidate value	Value
m	The number of population	$[n/2]$, n, $[3n/2]$, $2n$, $[5n/2]$, and $3n$	$2n$
Alpha (percent)	The at least value of using *gcbest*	0-5-10-15-20–25	10
Beta (percent)	The at most value of using *gcbest*	75-80-85–90–95–100	90
SC	The number repetition of *gbest* as the stop condition	$n/10$, $n/9$, $n/8$, $n/7$, $n/6$, $n/5$, $n/4$, and $n/3$	$n/4$

Table 2: The results of the proposed algorithm for the TSP instances.

	Instance	n	WS (PSO)		AS (PSO)		BS (PSO)		WS (MPSO)		AS (MPSO)		BS (MPSO)		BKS
			Cost	Gap	Cost	Gap	Cost	Gap	Cost	Gap	Cost	Gap	Cost	Gap	
1	Burma14	14	3323	0.00	3323	0.00	3323	0.00	3323	0.00	3323	0.00	3323	0.00	3323
2	Gr17	17	2085	0.00	2085	0.00	2085	0.00	2085	0.00	2085	0.00	2085	0.00	2085
3	Gr21	21	2707	0.00	2707	0.00	2707	0.00	2707	0.00	2707	0.00	2707	0.00	2707
4	Gr24	24	1272	0.00	1272	0.00	1272	0.00	1272	0.00	1272	0.00	1272	0.00	1272
5	Fri26	26	937	0.00	937	0.00	937	0.00	937	0.00	937	0.00	937	0.00	937
6	Bayg29	29	1610	0.00	1610	0.00	1610	0.00	1610	0.00	1610	0.00	1610	0.00	1610
7	Bays29	29	2020	0.00	2020	0.00	2020	0.00	2020	0.00	2020	0.00	2020	0.00	2020
8	Arr48	48	10983	3.34	10699	0.67	10628	0.00	10628	0.00	10628	0.00	10628	0.00	10628
9	Eil51	51	454	6.57	429	0.70	426	0.00	426	0.00	426	0.00	426	0.00	426
10	Pr76	76	113545	4.98	115003	6.33	109870	1.58	108159	0.00	108159	0.00	108159	0.00	108159
11	Rat99	99	1397	15.36	1373	13.38	1294	6.85	1211	0.00	1211	0.00	1211	0.00	1211
12	KroA100	100	23342	9.68	21843	2.64	21282	0.00	21352	0.33	21292	0.05	21282	0.00	21282
13	KroB100	100	23424	5.79	22964	3.72	22534	1.77	22141	0.00	22141	0.00	22141	0.00	22141
14	Rd100	100	8601	8.74	8285	4.74	8047	1.73	8135	2.84	8054	1.82	7910	0.00	7910
15	Eil101	101	699	11.13	657	4.45	643	2.23	629	0.00	629	0.00	629	0.00	629
16	Lin105	105	15897	10.56	15142	5.31	14910	3.69	14379	0.00	14379	0.00	14379	0.00	14379
17	Pr107	107	48933	10.45	46761	5.55	45602	2.93	44303	0.00	44303	0.00	44303	0.00	44303
18	Pr124	124	65911	11.66	63660	7.84	61801	4.69	59555	0.89	59113	0.14	59030	0.00	59030
19	Bier127	127	130110	10.00	129379	9.38	127830	8.07	118282	0.00	118282	0.00	118282	0.00	118282
20	Ch130	130	6779	10.95	6454	5.63	6299	3.09	6255	2.37	6176	1.08	6110	0.00	6110
21	Pr136	136	105839	9.37	102183	5.59	99575	2.90	97972	1.24	97543	0.80	96772	0.00	96772
22	Pr144	144	64869	10.82	62129	6.14	60293	3.00	58972	0.74	58612	0.13	58537	0.00	58537
23	Ch150	150	7325	12.21	6974	6.83	6724	3.00	6659	2.01	6579	0.78	6528	0.00	6528
24	KroA150	150	29488	11.17	28328	6.80	27720	4.51	26952	1.61	26711	0.71	26524	0.00	26524
25	Pr152	152	81378	10.44	77350	4.98	75492	2.46	74162	0.65	73915	0.32	73682	0.00	73682
26	Rat195	195	2686	15.63	2498	7.53	2383	2.58	2401	3.36	2351	1.21	2323	0.00	2323
27	D198	198	17828	12.98	16849	6.77	16653	5.53	16222	2.80	16024	1.55	15780	0.00	15780
28	KroA200	200	32799	11.68	31295	6.56	30249	3.00	31052	5.73	29818	1.53	29533	0.56	29368
29	Ts225	225	138929	9.70	133210	5.19	129442	2.21	129157	1.99	128385	1.38	126643	0.00	126643
30	Pr226	226	89207	11.00	85649	6.57	82761	2.98	81279	1.13	80990	0.77	80545	0.22	80369
31	Pr264	264	56140	14.26	53289	8.45	50205	2.18	50945	3.68	50178	2.12	49325	0.39	49135
32	A280	280	2883	11.79	2814	9.11	2658	3.06	2795	8.38	2632	2.06	2598	0.74	2579
33	Pr299	299	53016	10.01	51404	6.67	49582	2.89	49105	1.90	48589	0.83	48332	0.29	48191
34	Lin318	318	50215	19.48	47172	12.24	45311	7.81	45812	9.00	45391	8.00	43710	4.00	42029
35	Linhp318	318	49201	19.00	46356	12.12	44653	8.00	45066	9.00	44653	8.00	41345	0.00	41345
36	Rd400	400	18084	18.34	17215	12.66	16503	8.00	16656	9.00	16503	8.00	15892	4.00	15281
37	Pr439	439	127588	19.00	120083	12.00	115794	8.00	116967	9.09	115994	8.19	111875	4.34	107217
38	Si535	535	57256	18.18	55264	14.06	52326	8.00	52811	9.00	52326	8.00	50388	4.00	48450
39	U574	574	43917	19.00	41134	11.46	39857	8.00	40296	9.19	39757	7.73	36905	0.00	36905
40	Rat575	575	8060	19.00	7586	12.00	7215	6.53	7383	9.01	7355	8.59	7014	3.56	6773

In Table 3, 18 examples are intended to compare between the MPSO and the following seven algorithms, which have 24 to 200 nodes. In this table, the names of examples are shown in the second column and the best results obtained (BKS) in the last column. Also, the best results obtained by the mentioned algorithms are mentioned in other columns. It should be noted that since there are many TSP examples in literature, it is hard to find examples where all the algorithms have been implemented, so these 18 examples are considered in this table. In addition, the results of GA, BSO, and ACO + SEE algorithms are presented for four, five, and five instances, respectively, while the rest of the algorithms have been tested in a larger number of examples.

The genetic algorithm (GA) [16]

The particle swarm optimization (PSO) [17]

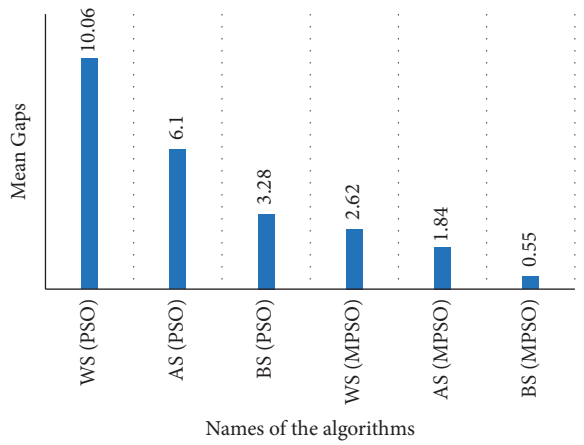

Figure 5: The mean Gaps of the PSO and MPSO algorithms.

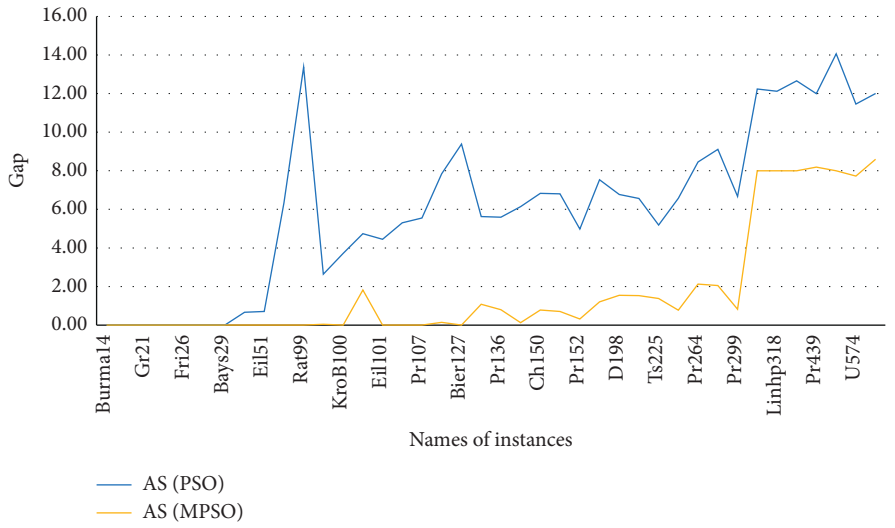

Figure 6: Comparing the mean Gaps in the AS mode for PSO and MPSO.

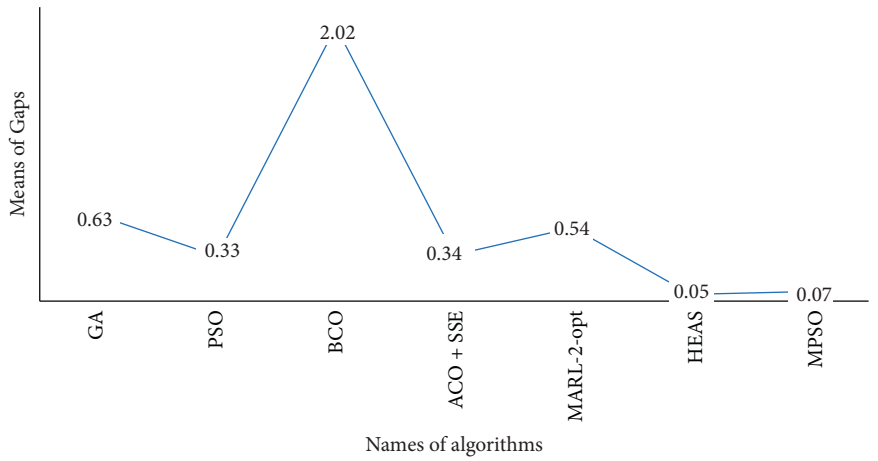

Figure 7: The mean Gaps of the algorithms.

Table 3: Comparing the proposed PSO algorithm with other algorithms.

	Instance	n	GA	PSO	BCO	ACO + SEE	MARL-2-opt	HEAS	MPSO	BKS
1	GR24	24	1272	—	—	—	—	—	1272	1272
2	GR48	48	5046	—	—	—	—	—	5046	5046
3	ATT48	48	—	—	10661	—	—	10628	10628	10628
4	Eil51	51	—	427	428	427	427	426	426	426
5	Berlin52	52	—	7542	—	7542	7542	7542	7542	7542
6	ST70	70	685	—	—	675	676	675	675	675
7	Eil76	76	—	540	539	546	544	538	538	538
8	KroA100	100	21504	21296	21763	21282	21282	21282	21282	21282
9	KroB100	100	—	—	22637	—	22237	—	22141	22141
10	KroC100	100	—	—	20853	—	—	—	20749	20749
11	KroD100	100	—	—	21643	—	—	—	21294	21294
12	KroE100	100	—	—	22450	—	—	—	22068	22068
13	Eil101	101	—	—	635	—	633	—	629	629
14	Lin105	105	—	—	15288	—	14412	—	14379	14379
15	KroA150	150	—	—	27858	—	26862	—	26524	26524
16	KroB150	150	—	—	26535	—	26464	26185	26254	26130
17	KroA200	200	—	29563	29961	—	29538	29420	29533	29368
18	KroB200	200	—	—	30350	—	—	—	29501	29437

The bee colony optimization (BCO) [18]

The improved ACO with pheromone correction strategy (ACO + SEE) [19]

The multi-agent reinforcement learning algorithm (MARL-2-opt) [20]

The hybrid elitist-ant system (HEAS) [21]

By comparing the results obtained in this table, it can be concluded that the proposed algorithm has been able to achieve more quality solutions than GA so that in two examples, St70 and KroA100 have increased the quality of the answers. On the contrary, in GR24 and GR48, the proposed algorithm has obtained the same solutions. Therefore, it can be concluded that although the GA algorithm has perfect performance for examples with small-size customer's numbers, when the number of customers increases, the GA algorithm, unlike the MPSO algorithm, cannot maintain its performance and achieves lower quality solutions.

The second algorithm that the results are compared with the proposed method is the PSO algorithm. This algorithm has been stable in the five examples that have been tested and has been able to obtain the solutions with less than one percent error. Although this algorithm has achieved the best solution so far in Berlin52 example, it has failed to respond with this accuracy in the remaining four examples. Also, in these four examples, the proposed algorithm has been able to obtain better solutions than this algorithm. Therefore, it can be concluded that the MPSO algorithm is more powerful than the PSO algorithm and has been able to escape from local optimal points. Comparing the proposed algorithm with the BCO algorithm, it can be seen that the BCO algorithm has only been able to obtain close solutions to the MPSO algorithm in Eil51 and Eil76 examples, and in the remaining 17 examples, the proposed algorithm has obtained much better solutions than this algorithm. It should be noted that the BCO algorithm in general is not efficient compared to other algorithms presented in this table, and in none of the examples, it has been able to achieve the BKS solutions.

The proposed algorithm is completely superior to the ACO-SEE algorithm because compared to the five problems which the solutions of this algorithm are presented, the solutions of the two algorithms have the same only in two examples. Also, in three other examples, the proposed MPSO algorithm has been able to improve the quality of the solutions of this algorithm and achieve better solutions. Also, Marl-2-opt's solutions are presented in 11 examples, which compared to the proposed algorithm solutions, have been able to achieve the same solutions in two examples. However, in nine other instances, this algorithm has not been able to achieve the quality of the MPSO solutions and has obtained worse solutions. The last algorithm in this table is compared to MPSO results is the HEAS algorithm, which has very good solutions compared to other metaheuristic algorithms. By comparing the results of this algorithm with the MPSO algorithm, it can be seen that this algorithm has been able to achieve a better solution than the proposed algorithm, although in another example, it has obtained a worse solution. Besides, these two algorithms have found the same solutions to the MPSO algorithm in six other examples.

In Figure 7, the average Gap for the results of the seven algorithms presented in Table 3 is shown in which the horizontal axis presents the names of the algorithms, and the average Gaps are shown in the vertical axis. In this figure, the weakest method is BCO, which has the average of 2.02 and has poor-quality solutions compared to the results of GA and MARL-2-opt algorithms because the average Gaps for these two algorithms are 0.63 and 0.54, respectively. On the contrary, the two algorithms PSO and ACO + SEE have very close answers and have an average of 0.33 and 0.34, respectively, which have obtained better answers than the previous three algorithms. Finally, HEAS and MPSO algorithms have the best solutions compared to the previous five algorithms that were able to achieve values of 0.05 and 0.07, which are outstanding results. It should be noted that the HEAS algorithm is presented in only eight examples in

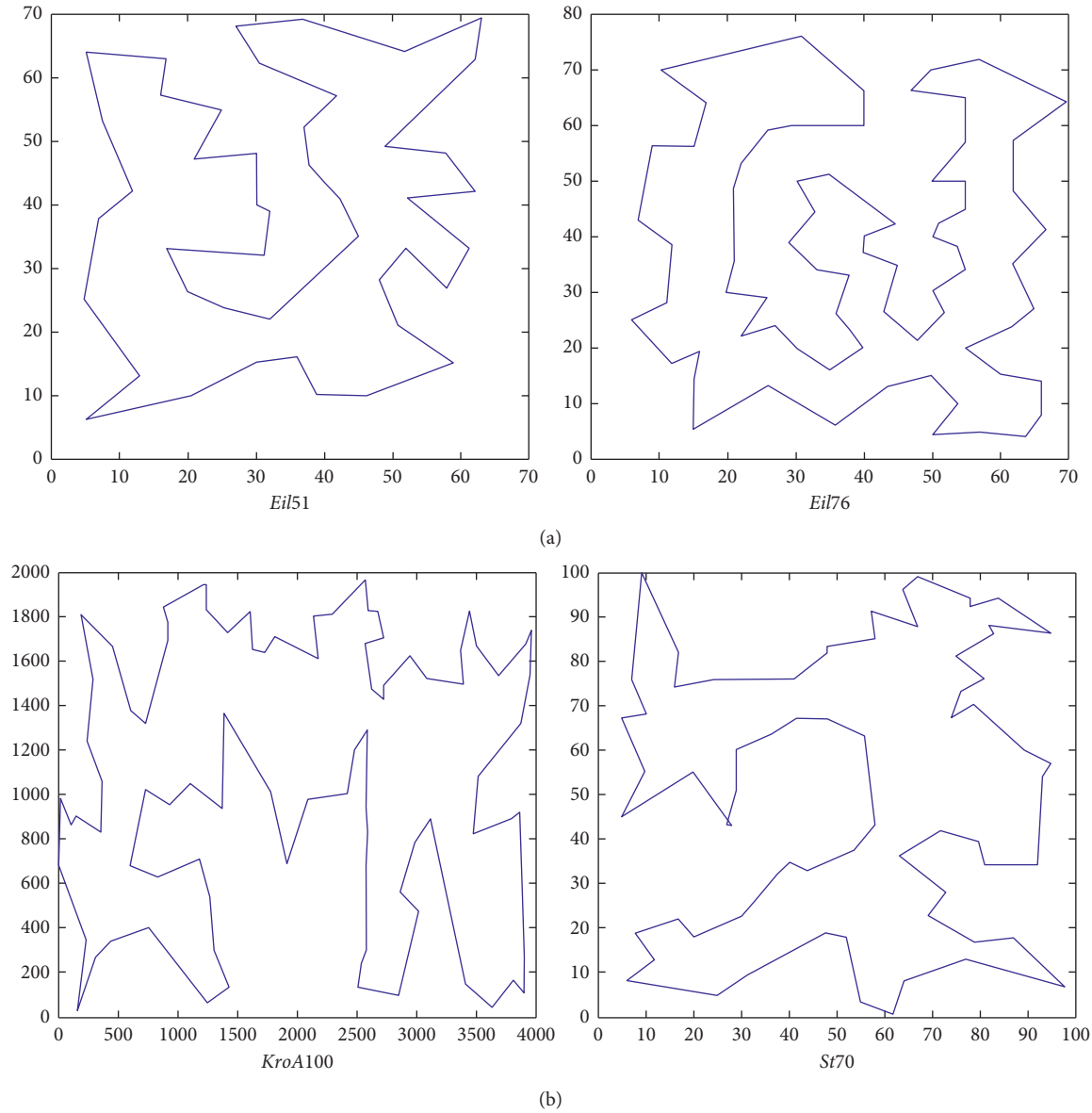

Figure 8: Some solutions obtained by MPSO.

this table, and compared to the proposed algorithm, its results are not presented in ten examples. Therefore, it is not possible to provide a more accurate comparison between these two algorithms, and it can only be concluded that these two algorithms have found the best results between the seven algorithms in 18 examples of Table 3.

In addition, the six solutions of the proposed algorithm are shown in Figure 8. In these solutions, the algorithm has been able to achieve the best-known solutions.

5. Conclusion and Future Works

A modified PSO algorithm called MPSO was used for solving the TSP problem in this paper. Also, to test the stability of the method, the worst, average, and best solutions are compared to the classic PSO in the number of standard problems which have a good range of customers. Since multiagent algorithms usually do not have a good performance in the intensification mechanism, the insert, exchange, 2-Opt, and inverse moves were used. Besides, some modifications were made to the PSO algorithm to better search the problem space and escape local optimum points. The results show the efficiency of the algorithm compared to other methods presented for the standard problems. For future tasks, this algorithm can be used to the vehicle routing and allocation problems because the efficiency of the algorithm grows when the number of customers of examples is increased, and the algorithm cannot be stable. Therefore, it is better to use the tabu search algorithm to increase the efficiency of the intensification mechanism. Also, finding a more efficient state for moving particles towards *gbest* and *pbest* causes to increase the global search of the problem space. Finally, some of the most representative computational

intelligence algorithms can be used to solve the TSP problem, like monarch butterfly optimization, earthworm optimization algorithm, elephant herding optimization, moth search algorithm, slime mould algorithm, and harris hawks optimization. The application of these ideas is postponed to other articles.

Data Availability

http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.

Conflicts of Interest

The author declares that there are no conflicts of interest.

References

[1] M. Yousefikhoshbakht, F. Didehvar, and F. Rahmati, "An effective rank based ant system algorithm for solving the balanced vehicle routing problem," *International Journal of Industrial Engineering*, vol. 23, no. 1, pp. 13–25, 2016.

[2] H. A. Saleh and R. Chelouah, "The design of the global navigation satellite system surveying networks using genetic algorithms," *Engineering Applications of Artificial Intelligence*, vol. 17, pp. 111–122, 2004.

[3] D. Chan and D. Mercier, "IC insertion: an application of the travelling salesman problem," *International Journal of Production Research*, vol. 27, no. 10, pp. 1837–1841, 1989.

[4] E. L. Lawler, J. K. Lenstra, and D. B. Shmoys, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Wiley, New York, NY, USA, 1985.

[5] B. Brummit and A. Stentz, "Dynamic mission planning for multiple mobile robots," *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 3, pp. 2396–2401, 1996.

[6] W. Zhang, "Truncated branch-and-bound: a case study on the asymmetric tsp," in *Proceedings of the Working Note of AAAI 1993 Spring Symposium: AI and NP-Hard Problems*, pp. 160–166, Stanford, CA, USA, 1993.

[7] T. Bektas, "The multiple traveling salesman problem: an overview of formulations and solution procedures," *Omega*, vol. 34, no. 3, pp. 209–219, 2006.

[8] S. Yadlapalli, W. A. Malik, S. Darbha, and M. Pachter, "A lagrangian-based algorithm for a multiple depot, multiple traveling salesmen problem," *Nonlinear Analysis: Real World Applications*, vol. 10, no. 4, pp. 1990–1999, 2009.

[9] S. Poikonen, B. Golden, and E. A. Wasil, "A branch-and-bound approach to the traveling salesman problem with a drone," *Informs Journal on Computing*, vol. 31, no. 2, pp. 335–346, 2019.

[10] B. Gavish and K. Srikanth, "An optimal solution method for large-scale multiple traveling salesmen problems," *Operations Research*, vol. 34, no. 5, pp. 698–717, 1986.

[11] F. Nakhaei, M. Irannajad, and M. Yousefikhoshbakht, "Flotation column performance optimisation based on imperialist competitive algorithm," *International Journal of Mining and Mineral Engineering*, vol. 7, no. 1, pp. 1–17, 2016.

[12] A. Rahmani and M. Yousefikhoshbakht, "Capacitated facility location problem in random fuzzy environment: using (α, β)-cost minimization model under the Hurwicz criterion," *Journal of Intelligent & Fuzzy Systems*, vol. 25, no. 4, pp. 953–964, 2013.

[13] A. Zafari, S. Tashakori, and M. Yousefi Khoshbakht, "A hybrid effective genetic algorithm for solving the vehicle routing problem," *International Journal of Industrial Engineering & Production Research*, vol. 21, no. 2, pp. 63–76, 2010.

[14] M. Yousefikhoshbakht, F. Didehvar, and F. Rahmati, "A mixed integer programming formulation for the heterogeneous fixed fleet open vehicle routing problem," *Journal of Optimization in Industrial Engineering*, vol. 8, no. 18, pp. 37–46, 2015.

[15] M. YousefiKhoshbakht and N. Mahmoodi Darani, "A combined metaheuristic algorithm for the vehicle routing problem and its open version," *Journal of AI and Data Mining*, vol. 7, no. 1, pp. 169–179, 2019.

[16] M. Ashouri and M. Yousefikhoshbakht, "A combination of meta-heuristic and heuristic algorithms for the VRP, OVRP and VRP with simultaneous pickup and delivery," *Brain Broad Research in Artificial Intelligence and Neuroscience*, vol. 8, no. 2, pp. 81–95, 2017.

[17] Z. S. Eskandaria and M. Yousefikhoshbakhtb, "Solving the vehicle routing problem by an effective reactive bone route algorithm," *Transportation Research*, vol. 2, no. 1, p. 39, 2012.

[18] M. Yousefikhoshbakht, F. Didehvar, and F. Rahmati, "An efficient solution for the VRP by using a hybrid elite ant system," *International Journal of Computers Communications & Control*, vol. 9, no. 3, pp. 340–347, 2014.

[19] M. Sedighpour, V. Ahmadi, M. Yousefikhoshbakht, F. Didehvar, and F. Rahmati, "Solving the open vehicle routing problem by a hybrid ant colony optimization," *Kuwait Journal of Science*, vol. 41, no. 3, 2014.

[20] F. Nakhaei, M. Irannajad, and M. Yousefikhoshbakht, "Simultaneous optimization of flotation column performance using genetic evolutionary algorithm," *Physicochemical Problems of Mineral Processing*, vol. 52, 2016.

[21] S. S. Ray, S. Bandyopadhyay, and S. K. Pal, "New operators of genetic algorithms for traveling salesman problem," in *Proceedings of the 17th International Conference on Pattern Recognition*, pp. 497–500, Cambridge, UK, 2004.

[22] W. Zhong, J. Zhang, and W. Chen, "A novel discrete particle swarm optimization to solve traveling salesman problem," in *Proceedings of the 2007 IEEE Congress on Evolutionary Computation*, pp. 3283–3287, Singapore, 2007.

[23] L. P. Wong, M. Y. H. Low, and C. S. Chong, "A bee Colony optimization algorithm for traveling salesman problem," in *Proceedings of the 2008 Second Asia International Conference on Modelling & Simulation (AMS)*, pp. 818–823, Kuala Lumpur, Malaysia, 2008.

[24] M. Tuba and R. Jovanovic, "Improved ACO algorithm with pheromone correction strategy for the traveling salesman problem," *International Journal of Computers Communications & Control*, vol. 8, no. 3, pp. 477–485, 2013.

[25] G. M. Jaradat, G. M. Jaradat, "Hybrid elitist-ant system for a symmetric traveling salesman problem: case of Jordan," *Neural Computing and Applications*, vol. 29, no. 2, pp. 565–578, 2018.

[26] T. Huang, Y. J. Gong, S. Kwong, H. Wang, and J. Zhang, "A niching memetic algorithm for multi-solution traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 24, 2019.

[27] A. C. Cinar, S. Korkmaz, and M. S. Kiran, "A discrete tree-seed algorithm for solving symmetric traveling salesman problem," *Engineering Science and Technology, an International Journal*, vol. 23, no. 4, pp. 879–890, 2020.

[28] S. H. Zhan, J. Lin, Z. J. Zhang, and Y. W. Zhong, "List-based simulated annealing algorithm for traveling salesman

problem," *Computational Intelligence and Neuroscience*, vol. 2016, Article ID 1712630, , 2016.

[29] M. Yousefikhoshbakht, E. Mahmoodabadi, and M. Sedighpour, "A modified elite ACO based avoiding premature convergence for traveling salesmen problem," *Journal of Industrial Engineering International*, vol. 7, no. 5, pp. 68–75, 2011.

[30] M. Yousefikhoshbakht, N. Malekzadeh, and M. Sedighpour, "Solving the traveling salesman problem based on the genetic reactive bone route algorithm whit ant colony system," *International Journal of Production Management and Engineering*, vol. 4, no. 2, pp. 65–73, 2016.

[31] C.-H. Wang and J.-Z. Lu, "A hybrid genetic algorithm that optimizes capacitated vehicle routing problems," *Expert Systems with Applications*, vol. 36, no. 2, pp. 2921–2936, 2009.

[32] X. Zhang and L. Tang, "A new hybrid ant colony optimization algorithm for the vehicle routing problem," *Pattern Recognition Letters*, vol. 30, no. 9, pp. 848–855, 2009.

[33] T. J. Ai and V. Kachitvichyanukul, "Particle swarm optimization and two solution representations for solving the capacitated vehicle routing problem," *Computers & Industrial Engineering*, vol. 56, no. 1, pp. 380–387, 2009.

[34] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of IEEE International Conference on Neural Networks*, pp. 1942–1948, Perth, Australia, 1995.

[35] X. Zhang, H. Liu, and L. Tu, "A modified particle swarm optimization for multimodal multi-objective optimization," *Engineering Applications of Artificial Intelligence*, vol. 95, Article ID 103905, 2020.

[36] A. A. Karim, N. A. Mat Isa, and W. H. Lim, "Modified particle swarm optimization with effective guides," *IEEE Access*, vol. 8, pp. 188699–188725, 2020.

[37] H. Liu, X.-W. Zhang, and L.-P. Tu, "A modified particle swarm optimization using adaptive strategy," *Expert Systems with Applications*, vol. 152, Article ID 113353, 2020.

[38] S. Mirjalili, J. S. Dong, A. Lewis, and A. S. Sadiq, "Particle swarm optimization: theory, literature review, and application in airfoil design," in *Nature-Inspired Optimizers*, pp. 167–184, Springer, Cham, Switzerland, 2020.

[39] S.-K. S. Fan and C.-H. Jen, "An enhanced partial search to particle swarm optimization for unconstrained optimization," *Mathematics*, vol. 7, no. 4, p. 357, 2019.

[40] C. Ou and W. Lin, "Comparison between PSO and GA for parameters optimization of PID controller," in *Proceedings of the 2006 International Conference on Mechatronics and Automation*, pp. 2471–2475, IEEE, Luoyang, China, 2006.

[41] X. F. Song, Y. Zhang, D. W. Gong, and X. Y. Sun, "Feature selection using bare-bones particle swarm optimization with mutual information," *Pattern Recognition*, vol. 112, Article ID 107804, 2021.

[42] X.-F. Song, Y. Zhang, Y.-N. Guo, X.-Y. Sun, and Y.-L. Wang, "Variable-size cooperative coevolutionary particle swarm optimization for feature selection on high-dimensional data," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 5, pp. 882–895, 2020.

[43] Z. Yong, Y. Li-Juan, Z. Qian, and S. Xiao-Yan, "Multi-objective optimization of building energy performance using a particle swarm optimizer with less control parameters," *Journal of Building Engineering*, vol. 32, Article ID 101505, 2020.

[44] Y. Zhang, H.-G. Li, Q. Wang, and C. Peng, "A filter-based bare-bone particle swarm optimization algorithm for unsupervised feature selection," *Applied Intelligence*, vol. 49, no. 8, pp. 2889–2898, 2019.

[45] M. Mao, Q. Duan, L. Zhang, H. Chen, B. Hu, and P. Duan, "Maximum power point tracking for cascaded PV-converter modules using two-stage particle swarm optimization," *Scientific Reports*, vol. 7, no. 1, pp. 1–10, 2017.

[46] F. Maleki and M. Yousefikhoshbakht, "A hybrid algorithm for the open vehicle routing problem," *International Journal of Optimization in Civil Engineering*, vol. 9, no. 2, pp. 355–371, 2019.

[47] M. Yousefikhoshbakht, A. Dolatnejad, F. Didehvar, and F. Rahmati, "A modified column generation to solve the heterogeneous fixed Fleet open vehicle routing problem," *Journal of Engineering*, vol. 2016, Article ID 5692792, 2016.

[48] N. Mahmoodi Darani, V. Ahmadi, Z. Saadati Eskandari, and M. Yousefikhoshbakht, "Solving the capacitated clustering problem by a combined meta-heuristic algorithm," *Journal of Advances in Computer Research*, vol. 4, no. 1, pp. 89–100, 2013.