International Journal of
Intelligent Systems and Applications in Engineering

ISSN:2147-6799 www.ijisae.org Original Research Paper

Solution of Multiple Travelling Salesman Problem using Particle Swarm Optimization based Algorithms

Sevda Dayioglu Gulcu[1] Humar Kahramanli Ornek[2,*]

Submitted: 02.04.2019 Accepted : 07/05/2019

Abstract: Nowadays, the systems that are inspired by biological structures have gained importance and attracted the attention of researchers. The Multiple Travelling Salesman Problem (MTSP) is an extended version of the TSP. The aim in the MTSP is to find the tours for *m* salesmen, who all start and end at the depot, such that each intermediate node is visited exactly once and the total cost of visiting nodes is minimized. The Particle Swarm Optimization (PSO) algorithm which is a meta-heuristic algorithm based on the social behaviour of birds. In this article, 2 algorithms based on PSO, called APSO and HAPSO, were proposed to solve the MTSP. The APSO algorithm is based on the PSO and 2-opt algorithms, the path-relink and swap operators. While the HAPSO algorithm is based on the GRASP, PSO and 2-opt algorithms, the path-relink and swap operators. In the experiments, 5 TSP instances are used and the algorithms are compared with the GA and ACO algorithms. According to the results, the HAPSO algorithm has the better performance than the other algorithms on the most instances. Moreover the HAPSO algorithm produces more stable results than the APSO algorithm and the performance of the HAPSO algorithm is better in all the MTSP instances. Therefore, the HAPSO algorithm is more robust than the APSO algorithm.

Keywords: *2-opt algorithm, grasp algorithm, multiple travelling salesman problem, particle swarm optimization*

1. Introduction

Nowadays, the systems that are inspired by biological structures have gained importance and attracted the attention of researchers. Some social systems in nature exhibit collective intelligence, although they are created by simple individuals with limited abilities. Intelligent solutions applied to the problems arise from the organizations and their indirect communication among these individuals. These systems are the source of important techniques in the development of the artificial intelligence systems [1].

The Travelling Salesman Problem (TSP) is one of the most studied optimization problems. The mathematical definition of optimization refers to the process of systematically analysing or solving a problem by selecting and assigning real or integer values to a function to minimize or maximize the function. Optimization problems are often NP-hard, complex and time-consuming. In TSP, there is a salesman and the nodes that the distance between them are known. All nodes must be visited only once and the salesman must pass all points at least cost and return to the starting point again [2]. Easy formulation of TSP, its difficult solution and having a large number of application areas have caused many studies on this problem. There are many types and generalizations of the TSP in the literature [3]. One of them is the multiple TSP (MTSP). In MTSP there are *n* cities and *m* salesmen. Each of the cities is assigned to a salesman and thus *m* subtours are created. MTSP is the more complex problem than TSP. For the solution, it is necessary to determine which cities must be assigned to which salesmen and to determine the optimal arrangement of the cities in the tours of the salesmen. The most common application of the TSP is in the field of scheduling. Scheduling the work on a production line is usually modelled as TSP. In the case where production expands on multiple parallel lines to be allocated to the works, the problem is modelled as MTSP. Some problems in daily life appear as a MTSP. Placing advertisements in newspapers and media [4] and printing scheduling problem [5] were seen as a MTSP and solved. School bus routing problem [6], planning of rescue units in natural disaster management [7], scheduling of staffs in the photo studio [8] and scheduling of services in the field of home care services [9] were seen as a MTSP and solved. As can be seen, in many different areas, MTSP is emerging and therefore it is very important to solve the MTSP.

Various heuristic and meta-heuristic techniques have been developed for solving optimization problems and new techniques have been also recommended. Here are a few of them: 2-opt algorithm [10], particle swarm optimization (PSO) algorithm [11], genetic algorithm (GA) [12], greedy randomized adaptive search procedure (GRASP) [13], artificial bee colony (ABC) algorithm [14] and ant colony optimization (ACO) algorithm. [15]. Parallel [16] and hybrid methods [17] have been also proposed in recent years.

In the literature, there are many studies conducted on TSP in detail. Unfortunately, a detailed study is rarely found in the case of MTSP. The following are the literature review about MTSP.

Carter [18] proposed a new method for the MTSP solution using the genetic algorithm. The chromosome used in GA consists of two parts. In the first part of the chromosome, there are cities where salesmen visit. The length of the second part is equal to the number of salesmen and the number of cities which each salesman visits is located in each gene. Thus, it is clear which salesman visited which cities. The proposed two-part chromosome method were compared with the two-chromosome technique and one-chromosome technique. It is shown that the proposed method is more advantage than the others according to the experiments. The proposed method has also been used to solve industrial problems such as production planning problem, quality control planning problem.

Junjie and Dingwei [19] proposed a new method to solve the

[1]*Department of Computer Eng., Selcuk University, Konya-42130,Turkey*
ORCID ID: 0000-0002-9993-5222
[1]*Department of Computer Eng., Selcuk University, Konya-42130, Turkey,*
ORCID ID: 0000-0003-2336-7924
**Corresponding Author Email: hkahramanli@selcuk.edu.tr*

MTSP with the ability constraint using the ACO algorithm. They used the TSPLIB instances as the test problems in the experiments.
Dang et al. [20] proposed a new method based GA that simulates the evolution of Darwin to solve the MTSP. They transfer the MTSP to TSP and solve the TSP.
Bektas [21] presented a review article about the definition and formulation of MTSP, its variations and solver methods. In this study, MTSP is considered in terms of single and multi depot types. In addition, some examples about the application of the MTSP in the daily life are given. Information about the exact algorithms and heuristic methods which solve the MTSP is given. In his another study [22], he presented the new models and exact algorithms that solve the multi-depot MTSP.
Ponraj and Amalanathan [23] proposed a new method for solving the MTSP with road capacity. They aimed to minimize the tour times of the salesmen depending on the road capacity constraint. Furthermore, the proposed method was run in parallel on 85 processors in order to improve the performance of the algorithm. Thus, the computational time of the algorithm was minimized.
Yuan et al. [24] proposed a new crossover operator (TCX) to solve MTSP using GA. They used a two-part chromosome structure as a chromosome. They compared the TCX method with three different crossover methods.
Hou and Liu [25] analysed the general characteristics of the MTSP and solved the MTSP by converting it into a simple graph. They also improved the tours obtained using the 2-opt local search algorithm.
Király and Abonyi [26] proposed a GA-based multi-chromosome technique to solve the MTSP. The developed method was integrated with Google Maps and applied in a real logistics problem (a mobile mechanical procurement in one of Hungary's largest energy providers). In the developed method, the number of salesmen was limited to the lower and upper limits. In addition, the shifting, local search, crossover, swap and reverse operations are used as the mutation operator.
Necula et al. [27] handled the MTSP as a multi-optimization problem and solved the problem with ACO algorithm. They aimed both to reduce the longest tour costs and to create balanced subtours. For this purpose, they applied the Pareto method.
Zhou et al. [28] improved the GA with the roulette selection method and the elite selection method to solve MTSP and proposed two GA-based methods. Besides, they proposed four new mutation operators.
Recently, hybrid heuristic methods have been used to solve optimization problems. Yu et al. [29] proposed a hybrid method based on the GA, the k-means algorithm, the Lin-Kernighan algorithm and the branch and cut algorithm to solve the MTSP. The cities are exchanged among salesmen using the GA and the results are clustered with the k-means algorithm. At the lower level, the Lin-Kernighan and branch and cut algorithms are applied to solve the subproblems of each salesman. Thus, this method has the global optimization capability thanks to GA and has local optimization capability thanks to branch and cut algorithm. Shabanpour et al. [30] proposed a hybrid algorithm by combining the GA and the clustering technique to solve the MTSP.
In our study, two new PSO based algorithms are proposed to solve the MTSP.

2. Materials and Methods

2.1. Travelling Salesman Problem

TSP is an optimization problem. It was first formulated in 1930. The goal in the TSP is to find the shortest tour where a salesman starts in a certain city, visits each city only once and returns to the start city [15]. An example of a TSP and its solution is shown in Fig. 1. Considering the nodes on a graph and the costs between them, TSP can also be defined as traversing all the nodes in the graph at the most cost effective and each node must be visited only once [31]. As the number of nodes increases, it becomes very difficult to determine the most cost-effective tour in the graph. Therefore, TSP is classified as a NP-hard problem [32].

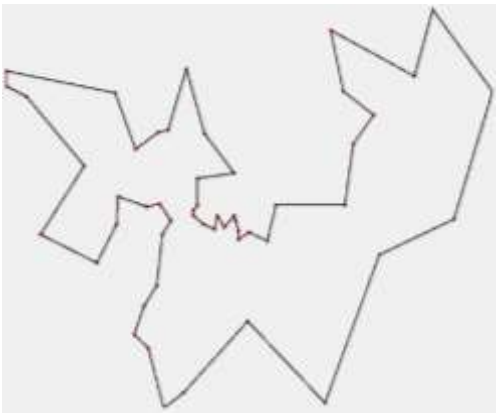

Fig. 1. A TSP instance (berlin52).

The TSP can be expressed mathematically as follows: Given the set of cities as $\{c_1, c_2, \ldots, c_n\}$ and the distance between two different cities as $d(c_i, c_j)$. The aim is to find the order of cities (π) that minimizes the tour length. The formulation of the TSP is shown in (1).

$$\min \sum_{i=1}^{n-1} d(c_{\pi(i)}, c_{\pi(j)}) + d(c_{\pi(n)}, c_{\pi(1)}) \quad (1)$$

2.2. Multiple Travelling Salesman Problem

The MTSP can generally be defined as: Given *n* cities and *m* salesmen. The *m* salesmen are located in a single depot. The other cities to be visited are called as intermediate cities. The aim in the MTSP is to find the tours for all *m* salesmen, who all start and end at the depot, such that each intermediate node is visited exactly once and the total cost of visiting all nodes is minimized [21]. Fig. 2 shows an example of the MTSP instance. In this example, there are 4 salesmen. Each salesman starts its tour from the depot city and completes its tour at the depot city.

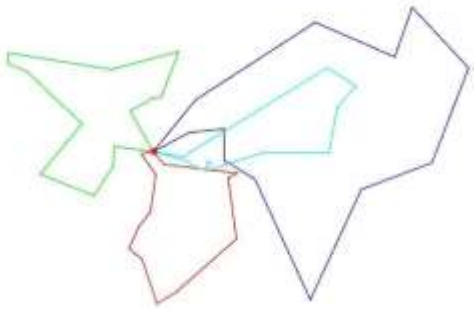

Fig. 2. An example of a single depot MTSP for 4 salesmen (berlin52).

In this article, the single depot MTSP is studied. However, in the literature there are various versions of MTSP [21]:
Multi-depot: The multi-depot MTSP has many depots and there is a salesman in each depot. After completing the tour, the salesman returns to the depot where he started.
Time windows: Certain cities must be visited during a certain

period of time.

Number of salesmen: In this version of the MTSP, the number of salesmen may be pre-determined or may be limited in certain intervals.

There is not any standard MTSP data set in the literature. Therefore, this is the biggest deficiency in testing the performance of the algorithms which solve the MTSP. Hence, the TSP data set is frequently used in the experiments. In this study, the symmetric TSP data set in TSPLIB library [33, 34] was used. The TSP data set used in the experiments is presented in Table 1. The "optimum" column in the table shows the best result of the TSP when the TSP is solved by a single salesman. Since the TSP instance is considered as a single depot MTSP in this study, more than one salesman solve this problem and the total tour length is naturally more.

Table 1. The TSP data set used in the experiments

#	Instance	Number of cities	Optimum
1	att48	48	10628
2	berlin52	52	7542
3	pr76	76	108159
4	rat99	99	1211
5	bier127	127	118282

2.3. Particle Swarm Optimization Algorithm (PSO)

The PSO algorithm which is a meta heuristic algorithm based on the social behaviour of birds striving to achieve a goal was developed by Kennedy and Eberhart [11, 35]. PSO algorithm has been used in different fields such as industry engineering [36], civil engineering [37], energy systems engineering [38], electrical engineering [39] and geology engineering [40] because of its successful performance. Qu and Lou [41] used the PSO algorithm for the optimal allocation of regional water resources. Balci and Valenzuela [42] developed a PSO-based method to solve the energy production scheduling problem. Gaur et al. [43] used the PSO algorithm and artificial neural network for the management of groundwater resources. Zhang et al. [44] applied the PSO algorithm to the multi-optimization problem and used it for feature selection in the field of data mining. Izquierdo [45] applied the PSO algorithm for the optimization of the wastewater collection network.

The PSO algorithm simulates the behaviour of bird flocks. The real-life bird is called a particle in the PSO algorithm and each particle represents a solution in the problem space. Each particle has a speed information and this speed information affects the next position of the particle. Also, the other important factors affecting the new position of the particle are the best position of the particle's history (*pBest*) and the best particle in the swarm (*gBest*). Fig. 3 shows the flowchart of the PSO algorithm.

At the beginning of the algorithm, the parameters of the algorithm are determined and the population is generated randomly. Then, the position of each particle is evaluated by the fitness function. The *pBest* and *gBest* information is calculated. The speed information of each particle is calculated using (2). The position information of each particle is updated using (3). This process continues until a stopping criterion is met and *gBest* information is reported as the output. The stop criterion is usually the maximum number of iterations.

$$V_i^d = V_i^d + c_1 * rand1_i^d * (pbest_i^d - X_i^d) + c_2 * rand2_i^d * (gbest^d - X_i^d) \quad (2)$$

$$X_i^d = X_i^d + V_i^d \quad (3)$$

where V_i^d represents the speed information of the particle i on the dimension d. X_i^d represents the position information the particle i on the dimension d. c_1 and c_2 parameters are acceleration coefficients and they controls the movement of the particle depending on *pBest* and *gBest* towards to the optimum. *rand1* and *rand2* are the random numbers generated between 0 and 1.

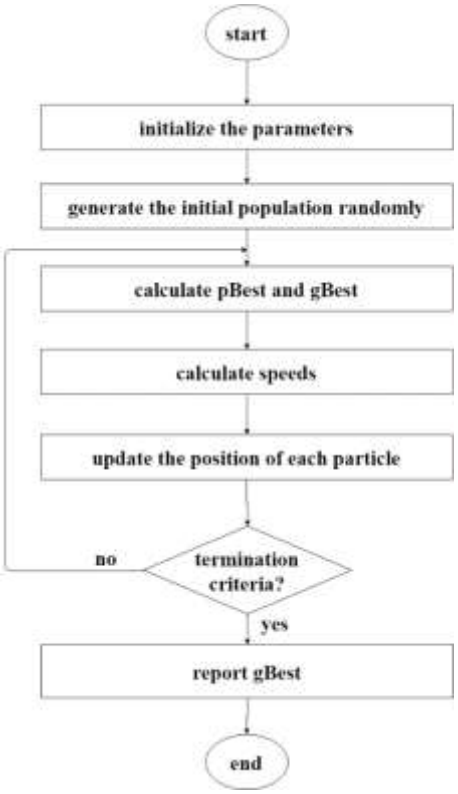

Fig. 3. The flowchart of the PSO algorithm.

2.4. Greedy Randomized Adaptive Search Procedure (GRASP)

The GRASP algorithm developed by Feo and Resende [13] is a meta-heuristic algorithm generally applied to combination optimization problems. The GRASP algorithm is a single-solution meta-heuristic method. In the single-solution meta-heuristic method, there is only one solution, and this solution can be improved through neighbourhoods [46]. Each iteration in the GRASP algorithm consists of 2 phases: the construction phase and the local search phase. The construction phase creates a feasible solution. Then, the local search phase starts to improve the solution generated. At the end of the algorithm, the best solution is reported as a result [47].

The pseudo-code of the algorithm is presented in Fig. 4, Fig. 5 and Fig. 6 for the minimization problem. The maximum number of iterations defined by the user T and the threshold value α are sent to the algorithm as the parameters. The threshold value α is set between 0 and 1. The algorithm calls the *Greedy_Randomized_Construction* method and the *Local_Search* method in each iteration until it reaches the maximum number of iterations. The *solution* found by the *Greedy_Randomized_Construction* method is tried to be improved by the *Local_Search* method. If the *solution* obtained by the *Local_Search* method is better than the *best_solution* up to this stage, the *best_solution* value is updated by assigning the *solution* value to the *best_solution* value.

```
procedure GRASP(Max_Iterations,Seed)
1    Read_Input();
2    for k = 1,...,Max_Iterations do
3        Solution ← Greedy_Randomized_Construction(Seed);
4        Solution ← Local_Search(Solution);
5        Update_Solution(Solution,Best_Solution);
6    end;
7    return Best_Solution;
end GRASP.
```

Fig. 4. The pseudo code of the GRASP algorithm [47].

The construction phase, the first phase of the GRASP algorithm, is a constructive heuristic. The constructive heuristics add a solution to the solution that is initially empty and produce a partial solution. This process continues until the solution is complete. [48]. The pseudo-code of this phase is presented in Fig. 5. The description of the abbreviation in the algorithm is as follows: *e* is the problem element and *E* is a finite set of the problem elements. *C* is the candidate set. *c*(*e*) represents the cost. The restricted candidate list (RCL) contains elements with low cost. For example for the TSP, *e* represents the path that connects the two cities, and *E* is all paths connecting the cities. *C* represents all cities that can be visited from a city. *c*(*e*) is the length of the path connecting the two cities. c_{min} shows the cost of the nearest city to go and c_{max} shows the cost of the farthest city to go. The restricted candidate list (RCL) is the list of the nearest cities that can be reached.

The construction phase of the algorithm works as follows: Initially, the *solution* is empty. The candidate set *C* is generated and the cost *c*(*e*) of each element in the candidate set *C* is calculated. The following steps continue as long as the candidate set *C* has an element and the *solution* obtained at the end of the construction phase is returned:

- The minimum cost c_{min} and the maximum cost c_{max} from the elements in the candidate set are calculated.
- The restricted candidate list is created using c_{min}, c_{max} and α threshold value.
- A random element *e* is selected from the restricted candidate list and added to the *solution*.
- The candidate set *C* is updated.
- The costs *c*(*e*) are recalculated.

```
procedure Greedy_Randomized_Construction(α, Seed)
1    Solution ← ∅;
2    Initialize the candidate set: C ← E;
3    Evaluate the incremental cost c(e) for all e ∈ C;
4    while C ≠ ∅ do
5        c^min ← min{c(e) | e ∈ C};
6        c^max ← max{c(e) | e ∈ C};
7        RCL ← {e ∈ C | c(e) ≤ c^min + α(c^max − c^min)};
8        Select an element s from the RCL at random;
9        Solution ← Solution ∪ {s};
10       Update the candidate set C;
11       Reevaluate the incremental costs c(e) for all e ∈ C;
12   end;
13   return Solution;
end Greedy_Randomized_Construction.
```

Fig. 5. The pseudo code of the construction phase [47].

The pseudo code of the local search phase of the algorithm is presented in Fig. 6. *f*(.) represents the fitness function. The *new solution* is obtained by making local changes on the existing *solution*. If the *new solution* is better than the existing *solution*, the algorithm continues with the *new solution*. The improved *solution* obtained at the end of the local search phase is returned.

```
procedure Local_Search(Solution)
1    while Solution is not locally optimal do
2        Find s' ∈ N(Solution) with f(s') < f(Solution);
3        Solution ← s';
4    end;
5    return Solution;
end Local_Search.
```

Fig. 6. The pseudo code of the local search phase [47].

2.5. 2-opt Algorithm

The 2-opt algorithm is a local search algorithm developed by Croes [10]. The 2-opt algorithm deletes the two edges in a TSP tour. In this way the tour is divided into two parts. Then, these two parts are reunited by experimenting with different possibilities to be a TSP tour. If the new tour is better than the old tour, then the new tour is the solution. The 2-Opt algorithm continues to erase the two edges and regenerate the tour until it cannot find improvement [3, 49]. A sample application of the 2-Opt local search algorithm is shown in Fig. 7. As shown in the figure, the tour is divided into 2 parts and these 2 parts are reassembled to form a different tour for a possible solution.

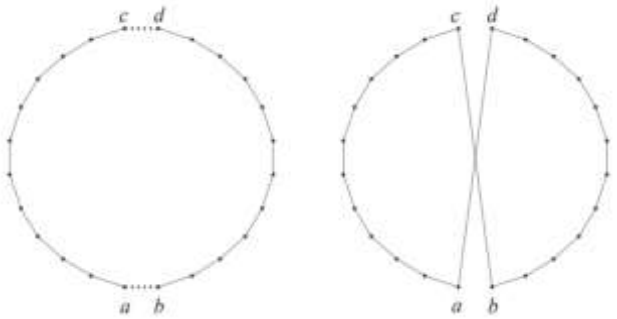

Fig. 7. A sample application of the 2-opt algorithm [49].

2.6. APSO

This section describes in detail the PSO based meta-heuristic algorithm (APSO) developed to solve the MTSP. The developed method solves the MTSP which is a discrete optimization problem using the PSO and 2-opt algorithms, the path-relink and swap operators. The initial population of the APSO algorithm is randomly generated. Each of these particles represents the solution of the MTSP. The APSO algorithm finds the new solutions in the search space to solve the MTSP. The 2-opt algorithm, the path-relink and swap operators improve the solutions obtained by the APSO algorithm. The flowchart of the developed APSO algorithm is shown in Fig. 8.

It is shown in Equation (2), there are three main factors affecting the movement of a particle in the PSO: the previous velocity information of the particle, the best position of the particle so far (*pBest*) and the best position of the particles in the swarm so far (*gBest*). Since the PSO algorithm was developed for the continuous optimization problems, the pure form of the algorithm cannot be applied directly to the discrete optimization problems. In this study, some changes have been made in the algorithm in order to use the PSO algorithm in the solution of the MTSP. These changes are the determining the new position of a particle using the 2-opt local search algorithm, the path-relink operator and the swap operator.

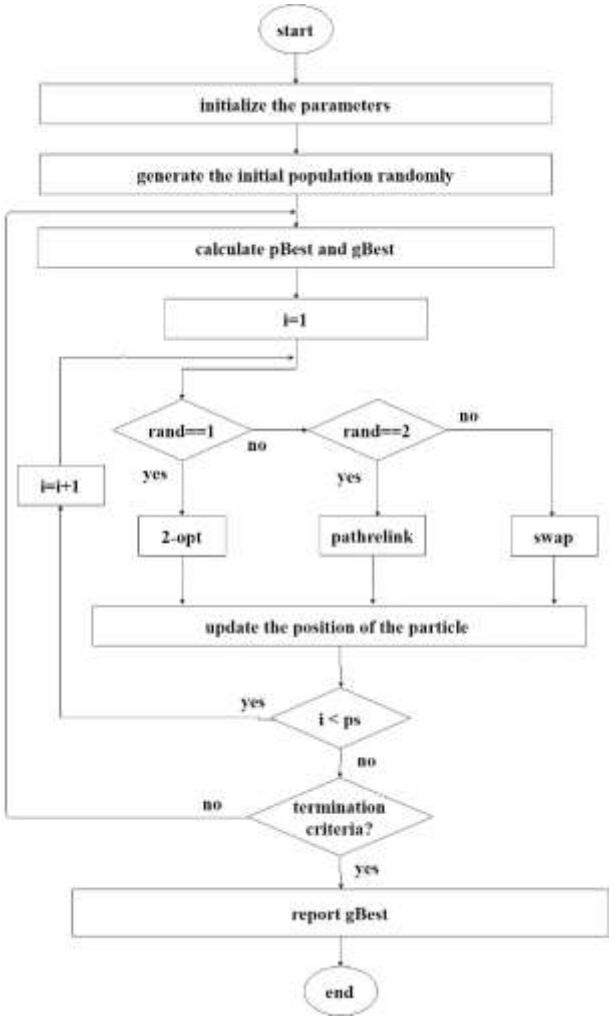

Fig. 8. The flowchart of the APSO algorithm (ps: population size, i: particle index).

In the developed algorithm, the number of travelling salesman *m* is determined by the user. Besides, in order to create a balanced tour, there is an upper limit *K* of the number of cities to be visited per salesman. This upper limit *K* is calculated using (4). The *K* value is calculated when the TSP instance to be solved in the TSPLIB library is transferred to the algorithm. When the berlin52 problem is asked to be solved with 5 salesmen, the *K* value is set 12. In other words, a salesman cannot visit more than 12 cities for the berlin52 problem.

$$K = \left((d-1+m)/m\right)+1 \tag{4}$$

where *K* represents the upper limit, *d* represents the dimension of the problem namely the number of the cities and *m* represents the number of the salesmen.

In the developed algorithm, a particle represents a solution of the MTSP. Each particle contains *m* subtours. A subtour is created by only one salesman. Each city has an id information and this information is the order of the cities in the TSP instance file. The first city in the file has an id value of 0 and this city is a depot city. Table 2 shows a sample of a particle subtour data. Each subtour starts with the depot city (id value 0). After the salesman visited the last city in the tour, the salesman returns to the depot city. For example in Table 2, the salesman with the index 1 returns to the city 0 to complete the tour after the salesman visited the city 41.

Table 2. A sample of a particle subtour data (5 salesmen)

Salesmen index	Subtour	Number of visited cities
1	[0, 48, 35, 33, 45, 24, 11, 27, 28, 1, 6, 41]	12
2	[0, 25, 46, 13, 51, 12, 26, 14, 23, 47, 36, 34]	12
3	[0, 31, 42, 32, 10, 50, 3, 5, 4, 37, 39, 38]	12
4	[0, 21, 30, 17, 16, 2, 40, 7, 8, 9, 18, 44]	12
5	[0, 43, 15, 49, 19, 22, 29, 20]	8

The flowchart of the developed APSO algorithm is shown in Fig. 8. The developed algorithm works as follows: First, the parameters of the algorithm are determined. These are the number of the particles, the maximum number of iterations, the number of the salesmen, the upper limit *K* of the number of cities to be visited per salesman. Then, the initial population of the PSO algorithm is randomly generated. The point to note here is that a city should be visited by only one salesman. Namely, any city except the depot city has to be in only one subtour. Thus, each particle has a solution consisting of *m* subtours for MTSP. Then the iteration cycle begins and the following steps repeat until the maximum iteration is reached. First, *pBest* and *gBest* information is calculated. Then, the following steps are applied for each particle: a random number *rand* is generated. *rand* $\in \{1, 2, 3\}$. If the *rand* value is 1, the 2-opt algorithm is applied to the subtours in the particle. If the *rand* value is 2, the path-relink operator is applied using *pBest* information to the subtours in the particle. If the *rand* value is 3, the swap operator is applied to the subtours in the particle. At the end of the algorithm, the *gBest* information is reported as the output. The path-relink operator and the swap operator used in the algorithm are described below.

The path-relink operator was developed by Glover [50] as a concentration strategy to guide the search towards elite results. [51]. The path-relink operator creates a chain of solutions between the two solutions. In this study, the source solution is the tour formed by the particle. The target solution is the *pBest* information of the particle. The intermediate solutions are produced between the source solution and the target solution and the quality of these intermediate solutions are evaluated. An example of the usage of the path-relink operator is given in Table 3. In the table, the source represents the subtour of the particle and the target represents the subtour in the *pBest* information. 4 solutions are produced between the source solution and the target solution. The fitness values of these solutions are calculated. If these solutions are better than *pBest* or *gBest*, the *pBest* and *gBest* information is updated.

Table 3. Path-relink operator

Step	Subtour
source	[0, 48, 35, 33, 45, 24, 11, 27, 28, 1, 6, 41]
1	[0, 48, 33, 35, 45, 24, 11, 27, 28, 1, 6, 41]
2	[0, 33, 48, 35, 45, 24, 11, 27, 28, 1, 6, 41]
3	[0, 33, 48, 35, 24, 45, 11, 27, 28, 1, 6, 41]
4	[0, 33, 48, 24, 35, 45, 11, 27, 28, 1, 6, 41]
target	[0, 33, 24, 48, 35, 45, 11, 27, 28, 1, 6, 41]

The swap operator is the process of exchanging a randomly selected city of 2 subtours randomly selected in a particle. Table 4 provides an example of the usage of the swap operator. First, two random salesmen are selected. In Table 4, these salesmen are 1 and 2. Then, a random city is selected from these salesmen and these cities are exchanged. In the example, the 45th and 36th cities are selected and exchanged.

Table 4. Swap operator

Salesman index	Subtour
1	[0, 48, 35, 33, **45**, 24, 11, 27, 28, 1, 6, 41]
2	[0, 25, 46, 13, 51, 12, 26, 14, 23, 47, **36**, 34]
	Swap operator ⇩
1	[0, 48, 35, 33, **36**, 24, 11, 27, 28, 1, 6, 41]
2	[0, 25, 46, 13, 51, 12, 26, 14, 23, 47, **45**, 34]

2.7. HAPSO

This chapter describes a hybrid meta-heuristic algorithm (HAPSO) based on the PSO and GRASP algorithms to solve MTSP. The developed method uses the GRASP, PSO and 2-opt algorithms and the path-relink and swap operators to solve MTSP. The GRASP algorithm is used to generate the initial population in the PSO algorithm. The PSO algorithm finds new solutions in the search space to solve the MTSP. The 2-opt algorithm improves the solutions of the PSO algorithm. The flowchart of the developed algorithm is shown in Fig. 9. The HAPSO algorithm an improved form of the APSO algorithm. In the APSO algorithm, the initial population of the algorithm is randomly generated. However, the initial population in the HAPSO algorithm is generated by the GRASP algorithm.

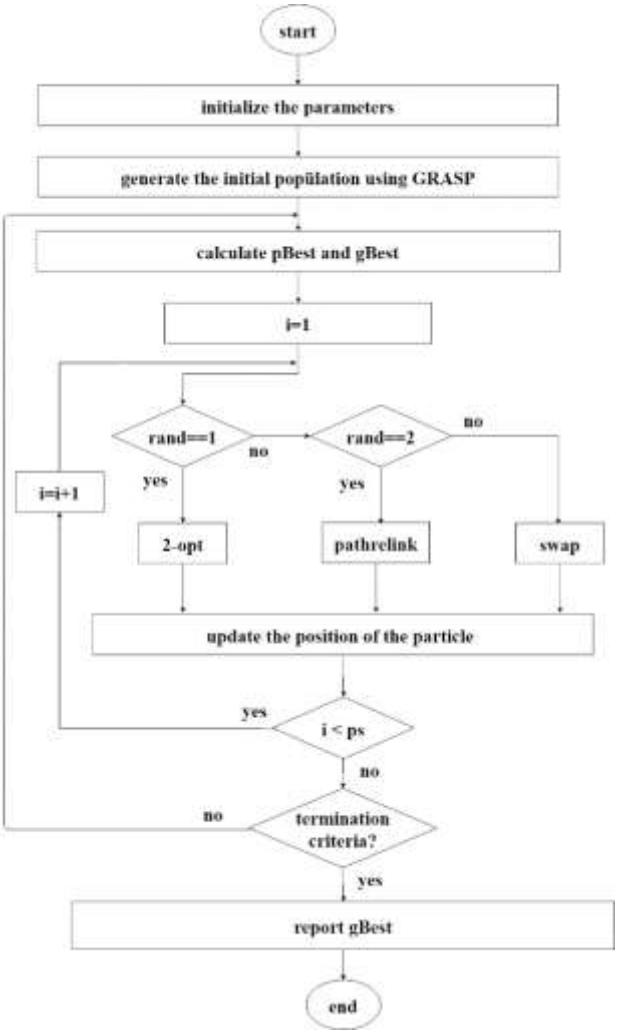

Fig. 9. The flowchart of the HAPSO algorithm (ps: population size, i: particle index).

In the HAPSO algorithm, the number of travelling salesman m is determined by the user. Besides, in order to create a balanced tour, there is an upper limit K of the number of cities to be visited per salesman. This upper limit K is calculated using (4).

In the HAPSO algorithm, a particle represents a solution of the MTSP. Each particle contains m subtours. A subtour is created by only one salesman. Each city has an id information and this information is the order of the cities in the TSP instance file. The first city in the file has an id value of 0 and this city is a depot city. Table 2 shows a sample of a particle subtour data.

The threshold value α in the GRASP algorithm determines the degree of the greed and is between 0 and 1. As its value is closer to 0, the greed is increasing. As its value is closer to 1, the randomness is increasing. In this study the threshold value α is taken as 0.4. In addition, the 2-opt heuristic algorithm is used as the local search algorithm in GRASP algorithm.

The flowchart of the HAPSO algorithm is shown in Fig. 9. The developed algorithm works as follows: First, the parameters of the algorithm are determined. These are the number of the particles, the maximum number of iterations, the number of the salesmen, the upper limit K of the number of cities to be visited per salesman and the threshold value α in the GRASP algorithm. Then, the initial population of the PSO algorithm is generated using the GRASP algorithm. Thus, each particle has a solution consisting of m subtours for MTSP. Then the iteration cycle begins and the following steps repeat until the maximum iteration is reached. First, *pBest* and *gBest* information is calculated. Then, the following steps are applied for each particle: a random number *rand* is generated. *rand* $\in \{1, 2, 3\}$. If the *rand* value is 1, the 2-opt algorithm is applied to the subtours in the particle. If the *rand* value is 2, the path-relink operator is applied using *pBest* information to the subtours in the particle. If the *rand* value is 3, the swap operator is applied to the subtours in the particle. At the end of the algorithm, the *gBest* information is reported as the output. The path-relink operator and the swap operator used in the algorithm are described in the previous section.

3. Experimental Results

In this section, the experimental results of the APSO and HAPSO algorithms are presented. The technical features of the computer used in the development of algorithms and experiments are as follows: Windows 10 operating system, Intel i5 3 GHz, 4 GB memory, java SE 8.

The total length of the tour varies according to the selection of the depot city. Therefore, the comparison of the experimental results of the studies in the literature is not appropriate and fair. In this study, the depot city is determined as the first city in the TSPLIB files to be read.

The developed APSO and HAPSO algorithms were compared with the ACO and GA algorithms in the literature [52] and the results are given in Table 5. We used 5 different symmetric TSP instances in comparison: att48, berlin52, bier127, pr76 and rat99. The parameters of the APSO and HAPSO algorithms are set as follows: the number of particles is 50, the maximum number of iterations is 2000 and the number of the salesmen is 2, 3 and 4 respectively. The upper limit K is calculated using (4). Furthermore, each experiment was run 20 times independently.

Table 5 presents the average results of the GA and ACO algorithms and the best, worst and mean results obtained by the developed APSO and HAPSO methods for 2, 3 and 4 salesmen.

As can be seen in the table, the HAPSO algorithm achieved better results than the APSO, GA and ACO algorithms for 2 salesmen. In

addition, the APSO algorithm achieved the second best results.

The HAPSO algorithm yielded better results than the APSO, GA and ACO algorithms for 3 salesmen.

The ACO algorithm yielded better results on the att48 and berlin52 instances for 4 salesmen. The HAPSO algorithm yielded better results than the APSO, GA and ACO algorithms results on the bier127 and pr76 instances. On the rat99 instance, the GA algorithm achieved better result. In addition, the HAPSO algorithm obtained the second best result on berlin52 and rat99 instances. In fact, as seen in the table, the result obtained by the HAPSO algorithm on the rat99 instance is close to the result obtained by the GA algorithm.

The graphs of the best tours found by the APSO and HAPSO algorithms for 2 salesmen are shown in Fig. 10. The tour of each salesman is shown with a different colour. Each salesman starts the tour from the same depot city and returns to the depot city after completing the tour.

Fig. 11, Fig. 12 and Fig. 13 show the boxplots of the results obtained by the APSO and HAPSO algorithms on the MTSP instances for 2, 3 and 4 salesmen. When the boxplots are analyzed, it is seen that the HAPSO algorithm produces more stable results than the APSO algorithm and its performance is better on all the MTSP instances. Therefore, the HAPSO algorithm is more robust than the APSO algorithm.

Table 5. Comparison of the APSO, HAPSO, GA and ACO algorithms for 2, 3, and 4 salesmen

Salesman (m)	Instance	APSO			HAPSO			GA	ACO
		Best	Worst	Mean	Best	Worst	Mean		
2	att48	43424	44929	44121.4	36891	38461	**37690.9**	50725.81	71151.36
	berlin52	9482	10244	9872.9	7994	8377	**8268.4**	11066.69	16354.02
	bier127	155651	165175	161693.8	125480	128649	**127089.9**	282343.86	425016.29
	pr76	143560	151283	148538.8	120490	124412	**123341.7**	184176.1	309514.32
	rat99	1662	1724	1698.6	1427	1463	**1442.3**	2487.64	3980.43
3	att48	51510	55974	54393.1	40637	45965	**43845.9**	49709.78	51032.81
	berlin52	11149	12089	11583.1	8876	9467	**9180.5**	10898.75	11726.73
	bier127	187575	196370	193300.1	129621	143525	**137325.9**	257228.63	349770.9
	pr76	174485	183105	179152.1	138096	148349	**143102.8**	170857.76	265550.49
	rat99	2018	2087	2048.9	1680	1772	**1734.1**	1970.48	3328.02
4	att48	58445	67170	64488.3	50014	55736	52716.6	47083.53	**42169.88**
	berlin52	12952	13602	13204.8	9893	10908	10365	11736.74	**8820.24**
	bier127	207814	219357	214992.2	141496	150427	**147184**.6	233708.3	294273.77
	pr76	201022	214005	207663.5	159964	175331	**168679.15**	168717.69	207333.11
	rat99	2353	2428	2386.7	1963	2097	2033.8	**1945.36**	2814.74

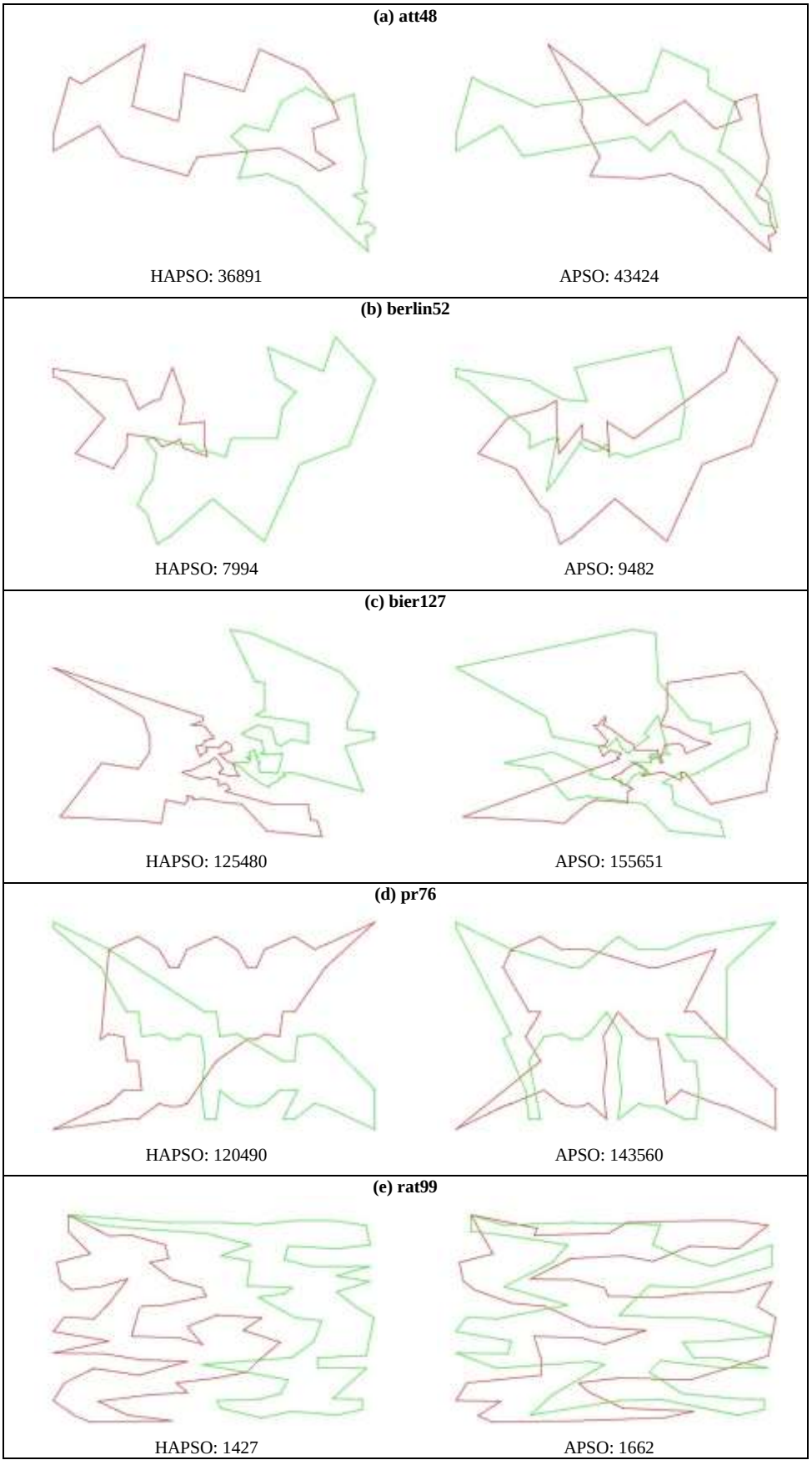

Fig. 10. Best tours of HAPSO and APSO for 2 salesmen on att48, berlin52, bier127, pr76 and rat99 instances.

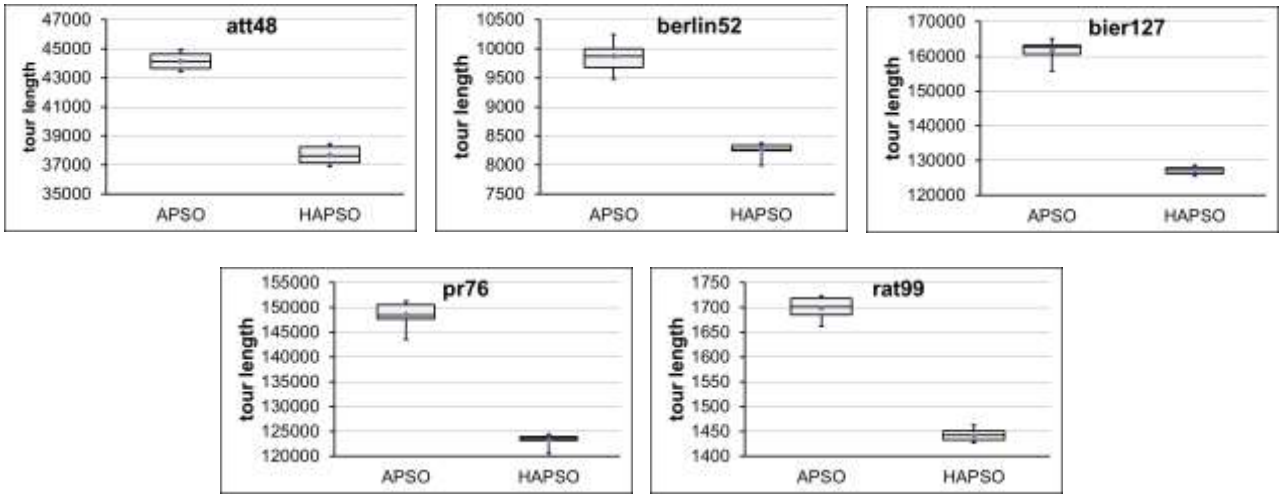

Fig. 11. Boxplot of the MTSP instances for 2 salesmen.

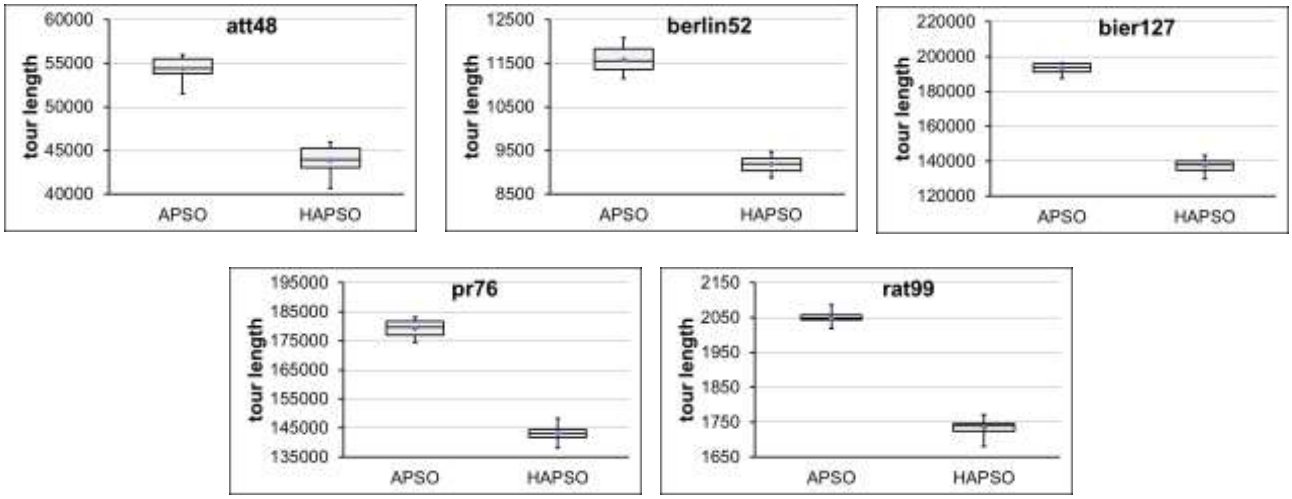

Fig. 12. Boxplot of the MTSP instances for 3 salesmen.

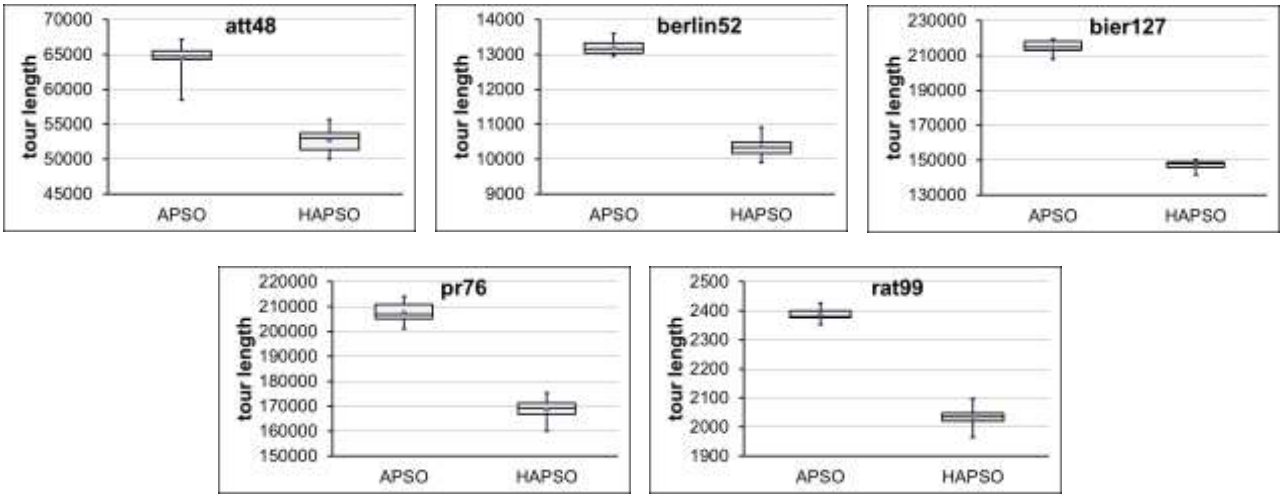

Fig. 13. Boxplot of the MTSP instances for 4 salesmen.

4. Conclusion

This article presents 2 meta-heuristic algorithms, APSO and HAPSO, to solve the MTSP problem. The first algorithm which is the APSO algorithm solves the MTSP which is a discrete optimization problem using the PSO and 2-opt algorithms, the path-relink and swap operators. The initial population of the APSO algorithm is randomly generated. Each of the particles represents the solution of the MTSP and contains a subtour for each salesman. The APSO algorithm finds the new solutions in the search space to solve the MTSP. The 2-opt algorithm, the path-relink and swap operators improve the solutions obtained by the APSO algorithm. The 2-opt algorithm deletes the two edges in a subtour of the particle. In this way the subtour is divided into two parts. Then, these two parts are reunited by experimenting with different possibilities to be a new subtour. The path-relink operator creates a chain of solutions between the two solutions. In this study, the source solution is the tour formed by the particle. The target solution is the *pBest* information of the particle. The intermediate solutions are produced between the source solution and the target solution and the quality of these intermediate solutions are evaluated. The swap operator is the process of exchanging a randomly selected city of 2 subtours randomly selected in a particle.

The second algorithm which is the HAPSO algorithm is based on the GRASP, PSO, 2-opt algorithms and the path-relink and swap operators. The HAPSO algorithm is the improved version of the APSO algorithm. The initial population of the HAPSO algorithm is generated using the GRASP algorithm. The 2-opt algorithm, the path-relink and swap operators improve the solutions obtained by the HAPSO algorithm.

Since the PSO algorithm was developed for the continuous optimization problems, the pure form of the algorithm cannot be applied directly to the discrete optimization problems. The contribution of this article is that the APSO and HAPSO algorithm based on PSO have the ability to solve the discrete optimization problem.

The aim in the MTSP is to find the tours for all *m* salesmen, who all start and end at the depot, such that each intermediate node is visited exactly once and the total cost of visiting all nodes is minimized. There is not any standard MTSP data set in the literature. Therefore, this is the biggest deficiency in testing the performance of the algorithms which solve the MTSP. Hence, the TSP data set is frequently used in the experiments. In this study, the symmetric TSP data set in TSPLIB library was used.

The APSO and HAPSO algorithms were compared with the ACO and GA algorithms in the literature. The HAPSO algorithm has the better results on the 13 instances. The ACO algorithm has the better results on the 2 instances. The GA algorithm has a better result on the one instance. In addition, the HAPSO algorithm produces more stable results than the APSO algorithm and its performance is better in all the MTSP instances. Therefore, the HAPSO algorithm is more robust than the APSO algorithm.

As the future work, we plan to develop the parallel versions of the APSO and HAPSO algorithms.

References

[1] V.V. Nabiyev, Yapay zeka: insan-bilgisayar etkileşimi, Seçkin Yayıncılık, 2012.

[2] M. Dorigo, T. Stützle, Ant Colony Optimization, Bradford Book, 2004.

[3] G. Gutin, A.P. Punnen, The traveling salesman problem and its variations, Springer Science & Business Media, 2006.

[4] A.E. Carter, C.T. Ragsdale, Scheduling pre-printed newspaper advertising inserts using genetic algorithms, Omega, 30 (2002) 415-421.

[5] S. Gorenstein, Printing press scheduling for multi-edition periodicals, Management Science, 16 (1970) B-373-B-383.

[6] A. Baykasoğlu, B.K. Özbel, Multiple traveling salesman game for cost allocation: a case problem for school bus services, in: LM-SCM 2016 XIV. International Logistics and Supply Chain Congress, 2016, pp. 64.

[7] F. Wex, G. Schryen, S. Feuerriegel, D. Neumann, Emergency response in natural disaster management: Allocation and scheduling of rescue units, European Journal of Operational Research, 235 (2014) 697-708.

[8] T. Zhang, W. Gruver, M.H. Smith, Team scheduling by genetic search, in: Intelligent Processing and Manufacturing of Materials, 1999. IPMM'99. Proceedings of the Second International Conference on, IEEE, 1999, pp. 839-844.

[9] D.S. Mankowska, F. Meisel, C. Bierwirth, The home health care routing and scheduling problem with interdependent services, Health care management science, 17 (2014) 15-30.

[10] G.A. Croes, A method for solving traveling-salesman problems, Operations research, 6 (1958) 791-812.

[11] J. Kennedy, R. Eberhart, Particle Swarm Optimization, in: IEEE International Conference on Neural Networks, 1995, pp. 1942-1948.

[12] J.H. Holland, Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence, MIT press, 1992.

[13] T.A. Feo, M.G. Resende, A probabilistic heuristic for a computationally difficult set covering problem, Operations research letters, 8 (1989) 67-71.

[14] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, Journal of global optimization, 39 (2007) 459-471.

[15] M. Dorigo, L.M. Gambardella, Ant colony system: a cooperative learning approach to the traveling salesman problem, Evolutionary Computation, IEEE Transactions on, 1 (1997) 53-66.

[16] Ş. Gülcü, H. Kodaz, A novel parallel multi-swarm algorithm based on comprehensive learning particle swarm optimization, Engineering Applications of Artificial Intelligence, 45 (2015) 33-45.

[17] Ş. Gülcü, M. Mahi, Ö.K. Baykan, H. Kodaz, A parallel cooperative hybrid method based on ant colony optimization and 3-Opt algorithm for solving traveling salesman problem, Soft Computing, 22 (2018) 1669-1685.

[18] A.E. Carter, Design and application of genetic algorithms for the multiple traveling salesperson assignment problem, in, Citeseer, 2003.

[19] P. Junjie, W. Dingwei, An ant colony optimization algorithm for multiple travelling salesman problem, in: Innovative Computing, Information and Control, 2006. ICICIC'06. First International Conference on, IEEE, 2006, pp. 210-213.

[20] J.-w. Dang, Y.-p. Wang, S.-x. Zhao, Study on a novel genetic algorithm for the combinatorial optimization problem, in: Control, Automation and Systems, 2007. ICCAS'07. International Conference on, IEEE, 2007, pp. 2538-2541.

[21] T. Bektas, The multiple traveling salesman problem: an overview of formulations and solution procedures, Omega, 34 (2006) 209-219.

[22] T. Bektaş, Formulations and Benders decomposition algorithms for multidepot salesmen problems with load balancing, European Journal of Operational Research, 216 (2012) 83-93.

[23] R. Ponraj, G. Amalanathan, Optimizing multiple travelling salesman problem considering the road capacity, Journal of computer science, 10 (2014) 680.

[24] S. Yuan, B. Skinner, S. Huang, D. Liu, A new crossover approach for solving the multiple travelling salesmen problem using genetic algorithms, European Journal of Operational Research, 228 (2013) 72-82.

[25] M. Hou, D. Liu, A novel method for solving the multiple traveling salesmen problem with multiple depots, Chinese science bulletin, 57 (2012) 1886-1892.

[26] A. Király, J. Abonyi, Redesign of the supply of mobile mechanics based on a novel genetic optimization algorithm using Google Maps API, Engineering Applications of Artificial Intelligence, 38 (2015) 122-130.

[27] R. Necula, M. Breaban, M. Raschip, Tackling the bi-criteria facet of multiple traveling salesman problem with ant colony systems, in: 2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI), IEEE, 2015, pp. 873-880.

[28] H. Zhou, M. Song, W. Pedrycz, A comparative study of improved GA and PSO in solving multiple traveling salesmen problem, Applied Soft Computing, 64 (2018) 564-580.

[29] Q. Yu, D. Wang, D. Lin, Y. Li, C. Wu, A novel two-level hybrid algorithm for multiple traveling salesman problems, in: International Conference in Swarm Intelligence, Springer, 2012, pp. 497-503.

[30] M. Shabanpour, M. Yadollahi, M.M. Hasani, A New Method to Solve the Multi Traveling Salesman Problem with the Combination of Genetic Algorithm and Clustering, IJCSNS, 17 (2017) 221.

[31] T. Kalaycı, Yapay zeka teknikleri kullanan üç boyutlu grafik yazılımları için "extensıble 3d"(x3d) ile bir altyapı oluşturulması ve gerçekleştirimi, Yüksek Lisans Tezi, Ege Üniversitesi, (2006).

[32] D. Donald, Traveling salesman problem, theory and applications, InTech, Rijeka, 2011.

[33] G. Reinelt, TSPLIB—A traveling salesman problem library, ORSA journal on computing, 3 (1991) 376-384.

[34] G. Reinelt, TSP library, in, http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/, 1995, last accessed march 2018.

[35] O. Bozorg-Haddad, M. Solgi, H.A. Loáiciga, Meta-heuristic and evolutionary algorithms for engineering optimization, John Wiley & Sons, 2017.

[36] M. Nouiri, A. Bekrar, A. Jemai, S. Niar, A.C. Ammari, An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem, Journal of Intelligent Manufacturing, 29 (2018) 603-615.

[37] A. AminShokravi, H. Eskandar, A.M. Derakhsh, H.N. Rad, A. Ghanadi, The potential application of particle swarm optimization algorithm for forecasting the air-overpressure induced by mine blasting, Engineering with Computers, 34 (2018) 277-285.

[38] S. Gulcu, H. Kodaz, The estimation of the electricity energy demand using particle swarm optimization algorithm: A case study of Turkey, Procedia computer science, 111 (2017) 64-70.

[39] J.H. Lee, J.-Y. Song, D.-W. Kim, J.-W. Kim, Y.-J. Kim, S.-Y. Jung, Particle swarm optimization algorithm with intelligent particle number control for optimal design of electric machines, IEEE Transactions on Industrial Electronics, 65 (2018) 1791-1798.

[40] M. Hajihassani, D.J. Armaghani, R. Kalatehjari, Applications of particle swarm optimization in geotechnical engineering: a comprehensive review, Geotechnical and Geological Engineering, 36 (2018) 705-722.

[41] G.-d. Qu, Z.-h. Lou, Application of particle swarm algorithm in the optimal allocation of regional water resources based on immune evolutionary algorithm, Journal of Shanghai Jiaotong University (Science), 18 (2013) 634-640.

[42] H.H. Balci, J.F. Valenzuela, Scheduling electric power generators using particle swarm optimization combined with the Lagrangian relaxation method, International Journal of Applied Mathematics and Computer Science, 14 (2004) 411-421.

[43] S. Gaur, S. Ch, D. Graillot, B.R. Chahar, D.N. Kumar, Application of artificial neural networks and particle swarm optimization for the management of groundwater resources, Water resources management, 27 (2013) 927-941.

[44] Y. Zhang, D.-w. Gong, J. Cheng, Multi-objective particle swarm optimization approach for cost-based feature selection in classification, IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB), 14 (2017) 64-75.

[45] J. Izquierdo, I. Montalvo, R. Pérez, V.S. Fuertes, Design optimization of wastewater collection networks by PSO, Computers & Mathematics with Applications, 56 (2008) 777-784.

[46] E.-G. Talbi, Metaheuristics: from design to implementation, John Wiley & Sons, 2009.

[47] M. Gendreau, J.-Y. Potvin, Handbook of metaheuristics, Springer, 2010.

[48] E. Alba, Parallel metaheuristics: a new class of algorithms, John Wiley & Sons, 2005.

[49] D.S. Johnson, L.A. McGeoch, The traveling salesman problem: A case study in local optimization, Local search in combinatorial optimization, 1 (1997) 215-310.

[50] F. Glover, Parametric combinations of local job shop rule, in: ONR Research Memorandum, Carnegie Mellon University, Pittsburgh, PA, 1963.

[51] M.G.C. Resende, C.C. Ribeiro, F. Glover, R. Martí, Scatter Search and Path-Relinking: Fundamentals, Advances, and Applications, in: M. Gendreau, J.-Y. Potvin (Eds.) Handbook of Metaheuristics, Springer US, Boston, MA, 2010, pp. 87-107.

[52] M. Latah, Solving multiple TSP problem by K-means and crossover based modified ACO algorithm, IJERT, 5 (2016) 430-434.