Konya Mühendislik Bilimleri Dergisi, c. 11, s. 3, 610-634, 2023
Konya Journal of Engineering Sciences, v. 11, n. 3, 610-634, 2023
ISSN: 2667-8055 (Electronic)
DOI: 10.36306/konjes.1199674

A DISCRETE PARTICLE SWARM ALGORITHM WITH SYMMETRY METHODS FOR DISCRETE OPTIMIZATION PROBLEMS

[1,*]Emine BAŞ, [2]Gülnur YILDIZDAN

[1]Konya Technical University, Engineering and Natural Sciences Faculty, Software Engineering Department, Konya, TÜRKİYE
[2]Selcuk University Kulu Vocational School, Konya, TÜRKİYE
[1]ebas@ktun.edu.tr, [2]gavsar@selcuk.edu.tr

Highlights

- DPSO is a discrete optimization method for discrete optimization problems.
- The DPSO has been applied to solve sixteen different TSPs taken from TSPLIB.
- The variations of the DPSO have occurred according to the selected method (DPSO1 (swap method), DPSO2 (shift method), DPSO3 (swap and shift methods), DPSO4 (symmetry method), DPSO5 (swap, shift, and symmetry methods), DPSO6 (swap, shift, symmetry, and 2-opt methods)).
- In this study, the shift and symmetry method is applied for the first time for DPSO.

***Corresponding Author:** Emine BAŞ, ebas@ktun.edu.tr

Konya Mühendislik Bilimleri Dergisi, c. 11, s. 3, 610-634, 2023
Konya Journal of Engineering Sciences, v. 11, n. 3, 610-634, 2023
ISSN: 2667-8055 (Electronic)
DOI: 10.36306/konjes.1199674

A DISCRETE PARTICLE SWARM ALGORITHM WITH SYMMETRY METHODS FOR DISCRETE OPTIMIZATION PROBLEMS

[1,*]Emine BAŞ, [2]Gülnur YILDIZDAN

[1]Konya Technical University, Engineering and Natural Sciences Faculty, Software Engineering Department, Konya, TÜRKİYE
[2]Selcuk University Kulu Vocational School, Konya, TÜRKİYE
[1]ebas@ktun.edu.tr, [2]gavsar@selcuk.edu.tr

(Received: 04.11.2022; Accepted in Revised Form: 28.04.2023)

ABSTRACT: Particle Swarm Optimization (PSO) is a commonly used optimization to solve many problems. The PSO, which is developed for continuous optimization, is updated to solve discrete problems and Discrete PSO (DPSO) is obtained in this study. With DPSO, the Traveling Salesman Problem (TSP), which is well-known in the literature as a discrete problem, is solved. In order to improve the results, the swap method, the shift method, and the symmetry method are added to DPSO. The symmetry method is a new and successful method. The variations of the DPSO occurred according to the selected method type (DPSO1 (swap method), DPSO2 (shift method), DPSO3 (swap and shift methods), DPSO4 (symmetry method), DPSO5 (swap, shift, and symmetry methods), DPSO6 (swap, shift, symmetry, and 2-opt methods)). The effect of each method on the performance of the DPSO has been studied in detail. To demonstrate the success of the variations of the DPSO, the results are additionally compared with many well-known and new discrete algorithms in the literature. The results showed that the performance of DPSO has improved with the symmetry method and it has achieved better results than the discrete heuristic algorithms recently proposed in the literature.

Keywords: *Swap, Shift, Symmetry, 2-OPT, Discrete Optimization, TSP*

1. INTRODUCTION

In the literature, many researchers have either developed old methods or proposed new methods to solve problems that are difficult to solve. In recent years, metaheuristic algorithms for such problems have gained great importance due to their success. Metaheuristic algorithms are created by imitating events that exist in nature. Most metaheuristic algorithms are created by imitating the feeding, hunting, cohabitation, and similar behaviors of living creatures. Many metaheuristic algorithms have been proposed in the literature. The oldest and most frequently used of these are Particle Swarm Optimization (PSO) [1], Ant Colony Optimization (ACO) [2], and Artificial Bee Colony (ABC) [3]. These algorithms have produced solutions for many different problems in the literature. Apart from these, there are different heuristic algorithms that have been newly proposed in the literature in recent years. Some of these are as follows: Snake Optimizer (SO) has created by snakes imitating their special mating behaviors [4], War Strategy Optimization (WSO) has based on the strategic movements of army weapons during the war [5], Red Fox Optimization (RFO) has created by imitating the hunting and feeding behavior of red foxes living in nature on the snowy ground [6], etc. PSO is chosen for discrete optimization problems in this paper. PSO is a heuristic algorithm based on mimicking bird and fish foraging behavior. PSO is simple to code, has few parameters, and has a fast convergence rate [7]. That's why it has been used by many researchers to solve engineering problems. It is used in a wide range of applications, including function optimization, neural network training, fuzzy system control, classification, pattern recognition, signal processing, and robot technology [7], [8], [9], [10], [11]. Gaing has proposed the discrete Binary PSO (BPSO) method in the literature. Gaing has combined BPSO and Lambda iteration methods in the study [12]. For the feature subset selection problem, Unler and Murat have created a modified discrete PSO algorithm [13]. Strasser et al. present a PSO version that can optimize over discrete variables [14]. To solve the no-wait flow shop

***Corresponding Author:** Emine BAŞ, ebas@ktun.edu.tr

scheduling problem with both makespan and total flowtime criteria, a Discrete PSO (DPSO) algorithm is presented [15]. Izakian et al. propose a discrete PSO (DPSO) method for grid job scheduling [16].

The optimization process is the process of obtaining the best value of a problem. In the optimization process, if the search space variables take continuous values, it is called continuous optimization, and if they take discrete values, it is called discrete optimization. Due to the nature of some problems, they take discrete values. The Traveling Salesman Problem (TSP) is one such problem. The TSP is a well-known multidisciplinary problem in operations research and computer science in which the goal is to find the shortest Hamiltonian cycle (circuit) in a network of cities (cost). The problem can be described as follows, given a set of cities (nodes) and the distances between them: A salesman should visit each of the remaining cities exactly once, starting and ending in a single depot city, to minimize the salesman's total journey distance (cost). Various scholars have investigated the TSP thoroughly, and as a result, several viable solutions (shift, swap, use the 2-opt algorithm, use the 3-opt algorithm, etc.) have been proposed [17], [18], [19], [20], [21]. Osaba et al. have proposed an improved discrete version of the WCA (dubbed the DWCA) to solve the Symmetric and Asymmetric TSPs [22]. The inclination feature in DWCA improves exploration and exploitation. Choong et al. have proposed a modified choice function for the TSP as a hyper-heuristic method [23]. Dahan et al. present a TSP adaptation of the Flying Ant Colony Optimization (FACO) algorithm [24]. For the TSP, Zhong et al. have proposed a hybrid discrete artificial bee colony algorithm with a threshold acceptance criterion [25]. Gao has suggested a new ACO algorithm [26]. A strategy of combining pairs of searching ants is used in this algorithm to diversify the solution space. Dong and Cai have introduced a new genetic algorithm for large-scale colored balanced TSPs [27]. Rokbani et al. have used FPA, PSO, and ant colony optimization heuristic algorithms as a hierarchical whole within the local search mechanism for TSP in 2021 [28]. Wu et al. have proposed a novel greedy genetic sparrow search algorithm based on a sine and cosine search strategy (GGSC-SSA) for TSP in 2021 [29]. Zhang and Han have proposed a discrete sparrow search algorithm (DSSA) with a global perturbation strategy to solve the TSP [30]. Huang et al. have proposed a discrete shuffled frog-leaping algorithm for TSPs. They have designed a new individual generation operator and four improved searching strategies to improve the algorithm performance [31]. Panwar and Deep have proposed a novel discrete GWO algorithm (D-GWO) to solve complex discrete TSPs [19]. Zhang and Yang have proposed a discrete cuckoo search algorithm based on the random walk and cluster analysis to solve the TSPs [32]. Zhang et al. have proposed a Discrete Mayfly Algorithm (DMA) for solving spherical asymmetric TSP. The DMA has had inver-over operator, a crossover operator, and a 3-opt operator [33].

In this study, the PSO, which is developed for continuous optimization, is updated to solve discrete problems and Discrete PSO (DPSO) is obtained. With DPSO, TSP, which is well known in the literature as a discrete problem, is solved. In order to improve the results, the swap method, the shift method, the symmetry method, and 2-opt method are added to DPSO. The symmetry method is a newly developed and successful method [21]. The variations of the DPSO have occurred according to the selected method type (DPSO1 (swap method), DPSO2 (shift method), DPSO3 (swap and shift methods), DPSO4 (symmetry method), DPSO5 (swap, shift, and symmetry methods), DPSO6 (swap, shift, symmetry, and the 2-opt methods)). Sixteen low, medium, and large-sized TSP datasets are solved with the variations of the DPSO and the results are compared with each other. In order to demonstrate the success of the variations of the DPSO, the results are compared with many well-known and new discrete algorithms in the literature.

1.1. The Primary Significant Contribution of the Research

The significant distinctions between our study and previous research, as well as the major additions to the literature, are noted below.

- ✓ DPSO is a discrete optimization method for discrete optimization problems.
- ✓ DPSO produces adequate and similar TSP solutions, according to the findings of the experiments.
- ✓ The DPSO has been applied to solve sixteen different TSPs taken from TSPLIB.

- ✓ The variations of the DPSO have occurred according to the selected method (DPSO1 (swap method), DPSO2 (shift method), DPSO3 (swap and shift methods), DPSO4 (symmetry method), DPSO5 (swap, shift, and symmetry methods), DPSO6 (swap, shift, symmetry, and 2-opt methods)).
- ✓ In this study, the shift and symmetry method is applied for the first time for DPSO in the literature.
- ✓ In this study, a population size (N) and the number of candidate solutions (CS) are examined in detail. Their effects on performance are shown.
- ✓ According to the best (Min), the worst (Max), mean (Avg), standard deviation (Std), relative error, and CPU time, the TSP results show that the proposed algorithm is a viable and competitive optimizer.

In this study, the materials and methods used in the paper (PSO, DPSO, TSP, the swap method, the shift method, the symmetry method, and the 2-opt method) are explained in Section 2, and the experimental results of the variations of the DPSO are given and discussed in Section 3. In Section 4 and Section 5, the results are explained and discussed.

2. MATERIAL AND METHODS

2.1. Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) was modeled in 1995 and is widely used today [1]. PSO is modeled with inspiration from flocks of birds and fish. In the first stage, birds do not know where the food is and look for food in different places. Every bird is considered a particle. All particles work to find the optimum result or solutions that are close to the optimum result. Particles are placed randomly or regularly. There is information sharing between all particles. According to the information sharing, the best position is found and that movement moves in other particles. The optimum result can be reached easily and quickly by sharing information about the particles.

In the PSO, each particle represents the solution-seeking agents of the swarm. It tries to get the best position by changing the positions of search agents in the search space depending on their local and global search capabilities. Each search agent updates its *position (a)* and *velocity* (*vel*) according to Equation 1 and Equation 2. Here, *Best* denotes the best search agent value and *GBest* denotes the global best search agent value.

$$vel_{ij}(t+1) = w * vel_{ij}(t) + k_1 rand_1 \left(Best_{ij}(t) - a_{ij}(t)\right) + k_2 rand_2 (GBest_{ij}(t) - a_{ij}(t)) \quad (1)$$

$$a_{ij}(t+1) = a_{ij}(t) + vel_{ij}(t+1) \quad (2)$$

where, $vel_{ij}(t+1)$ denotes velocities of agent in range [$-Vel_{max}$,Vel_{max}] (i=agent name, j=dimension name, t= iteration number). $a_{ij}(t+1)$ denotes positions of agent. w is the inertia weight. It is used to check the effect of previous speed history. k_1 and k_2 are the cognition learning factor and social learning factor. $rand_1$ and $rand_2$ are random numbers in range [0, 1] [1], [7]. The PSO's pseudo-code has been explained in Algorithm 1.

Algorithm 1: Pseudo-code of the PSO

```
Assign parameter values (c1, c2, w, etc.) of PSO and initialize.
Initialize the positions of the particles randomly. (Particle: i=1, ..., N) (Dimension: j=1, …, D)
while (iter < max_iter) do
  Calculate the objective function for the given particles
  Find the value of the best particle (Best) and find the value of the global best particle (GBest)
  for (i=1 to N) do
    for (j=1 to D) do
      Generate a random values between [0, 1] (rand1 and rand2)
      Calculate velocities of particles with Eq. (1)
      Calculate positions of particles with Eq. (2)
    end for
  end for
  iter=iter+1
end while
The value of the global best particle
```

2.2. Discrete Particle Swarm Optimization (DPSO)

PSO is modified to solve discrete optimization problems as follows: The particles are generated as random permutations during the initialization phase. In the PSO, each dimension of the particles holds the destination city information for the TSP. These values are obtained from the continuous search space obtained in PSO. Figure 1 shows the generation of the discrete search space from the continuous search space in PSO. For example, for a TSP with 10 cities, random variable values are generated with PSO in the range of [1] - [10]. These values are rounded to integer values to make them discrete. Since each city is visited once in TSP, the variable value should not repeat. Recurring variable values are detected with DPSO and the city name that is not included in the particle is added.

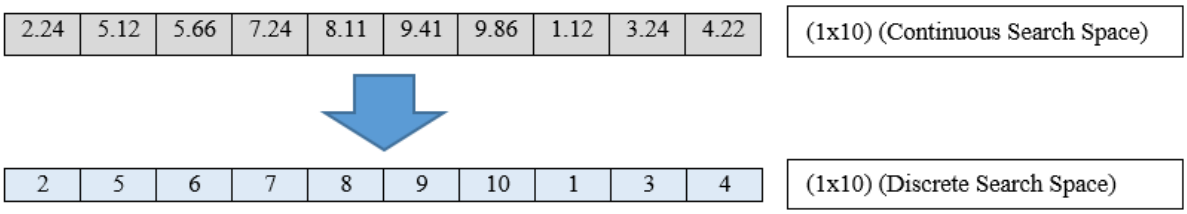

Figure 1. Generation of discrete search space (for DPSO) from continuous search space in PSO

The fitness value of each particle in the swarm is calculated. The individuals in the DPSO population randomly generate the city route to be visited. The resulting random route does not always perform well. Therefore, updates should be made on this route. DPSO generates candidate solutions for each particle by various methods (the swap method, the shift method, the symmetry method, and the 2-opt method). If the fitness value of the produced candidate solution is better, the current individual is replaced with the candidate solution. The swap method, the shift method, the symmetry method, and the 2-opt method are used for new candidate particles in the DPSO. The variations of the DPSO have occurred according to the selected method type (DPSO1 (swap method), DPSO2 (shift method), DPSO3 (swap and shift methods), DPSO4 (symmetry method), DPSO5 (swap, shift, and symmetry methods), DPSO6 (swap, shift, symmetry, and 2-opt methods)). The swap, shift, symmetry, and 2-opt methods are described in Section 2.3. The best solution is obtained after the termination criterion is met. The DPSO's pseudo-code is explained in Algorithm 2. Figure 2 shows the DPSO model.

Algorithm 2: Pseudo-code of the DPSO

```
Assign parameter values (N, D, CS, c1, c2, w, etc.) of DPSO and initialize.
Initialize the positions of the particles randomly. (Particle: i=1, ..., N) (Dimension: j=1, …, D) (D= total number of cities) (Particle: 1, 2, …, N)
while (iter < max_iter) do
  Calculate the objective function for the given particles for TSP (Total route length)
  Find the value of the best particle (Best) and find the value of the global best particle (GBest)
  for (i=1 to N) do
    for (j=1 to D) do
      Generate a random values between [0, 1] (rand1 and rand2)
      Calculate velocities of particles with Eq. (1)
      Calculate positions of particles with Eq. (2)
    end for
  end for
  for (i=1 to N) do
    for (j=1 to CS) do
      Apply swap method on new candidate particle
      Apply shift method on new candidate particle
      Apply symmetry method on new candidate particle
      Apply 2-opt method on new candidate particle
    end for
  end for
  Calculate the objective function for the new candidate particles for TSP (Total route length)
  for (i=1 to N) do
    if the fitness of new candidate particle (i) > the fitness of particle (i)
      Update particle (i) as new candidate particle (i)
    end if
  end for
 iter=iter+1
end while
The value of the global best particle
```

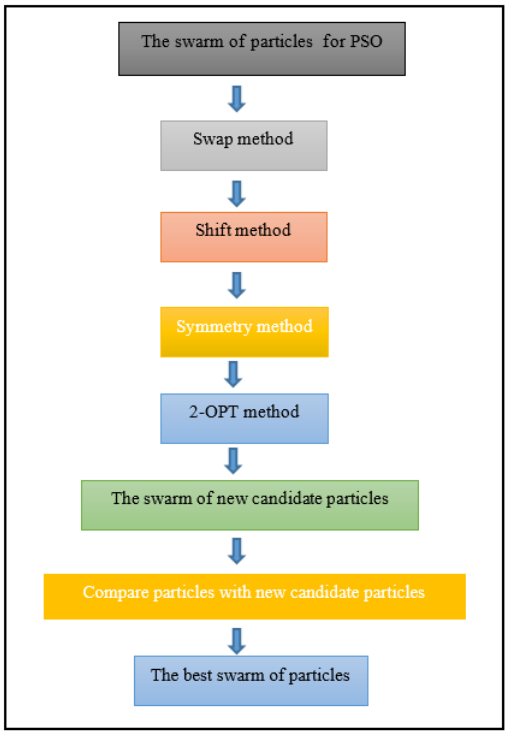

Figure 2. The DPSO model

2.3. Swap, Shift, Symmetry, and 2-opt Methods

2.3.1. Swap method

The swap transformation is based on swapping two randomly selected cities in each particle [21]. Thus, a new candidate particle is created. Although the swap transform is a simple method, it has been used by many researchers in the literature to create new candidate particles. It enables DPSO to approach the optimum result. The swap method is shown in Figure 3.

Particle=	1	2	3	4	5	6	7
Swap (2,5)							
The new candidate particle=	1	5	3	4	2	6	7

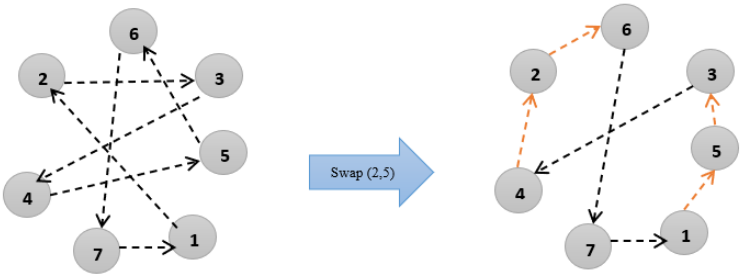

Figure 3. The swap method

2.3.2. Shift method

Each particle holds the city route information that will be visited by the vendor to calculate the fitness function of the problem. In order to generate candidate solutions in the Shift method, two different random cities are selected for each particle [21]. These two cities in the particle are swapped to create a new candidate particle, as in the swap method. During this change, the city location adjacent to the second randomly selected city is also included in the change. Shift transform causes position changes on the particle. In the swap method, two-city locations are changed, while in the shift method, three-city locations are changed. This creates a more efficient solution for finding the optimum route length. The shift method is shown in Figure 4 in detail.

Particle=	1	2	3	4	5	6	7
Shift (3,5)							
The new candidate particle=	1	2	4	5	3	6	7

Figure 4. The shift method

2.3.3. Symmetry method

The symmetry method is a method based on randomly changing city locations, as in the swap and shift methods [21]. In the symmetry method, there is more city location change than in the swap and shift methods. In the symmetry method, first of all, the city location is chosen randomly from 1-D in the form of two different groups. First of all, these two groups are subjected to a change of place among themselves, then the members of each group change their ranks among themselves. Thus, four different city locations are randomly changed. It also enhances the global search capability of the DPSO by placing groups of pairs at random locations in the search space. Relocation within the group also improves the local search capability of the DPSO in the search space. Thanks to this method, both the local and global search capability of DPSO is improved at the same time. While the swap method improves DPSO's global search capability, the shift method improves DPSO's local search capability. The success of a discrete heuristic algorithm depends on its ability to search both locally and globally. The symmetry method is shown in Figure 5 in detail.

Particle=	1	2	3	4	5	6	7
Symmetry ((1,2),(5,6))							
The new candidate particle=	6	5	3	4	2	1	7

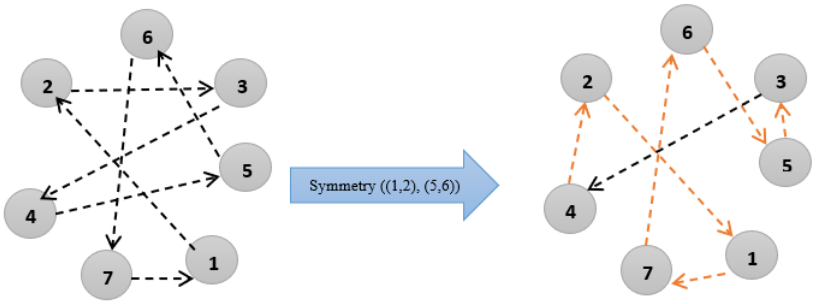

Figure 5. The symmetry method

2.3.4. 2-OPT method

A local search algorithm begins with a candidate solution and progresses iteratively to a neighboring solution. It is often referred to as k-opt. In the literature, 2-opt and 3-opt forms are mostly used in discrete optimization problems. The 2-opt algorithm is most likely the simplest and most widely used algorithm for solving TSP problems. It was first developed by Croes in 1958. The 2-opt local search algorithm seeks two distinct neighbor solutions. To improve the quality of the solution, a 2-opt local search method is implemented on the best individual [38]. Working with local search logic, the algorithm makes improvements by removing two edges from the round and connecting the remaining parts differently. The solution is obtained after all possible changes have been made is called 2-optimal. It can be defined as deleting the two edges in the tour and connecting the tour divided into two parts differently, reducing costs. The 2-opt method is shown in Figure 6 and Figure 7 in detail. In this study, the 2-OPT algorithm is applied on the last route obtained after the methods applied in DPSO.

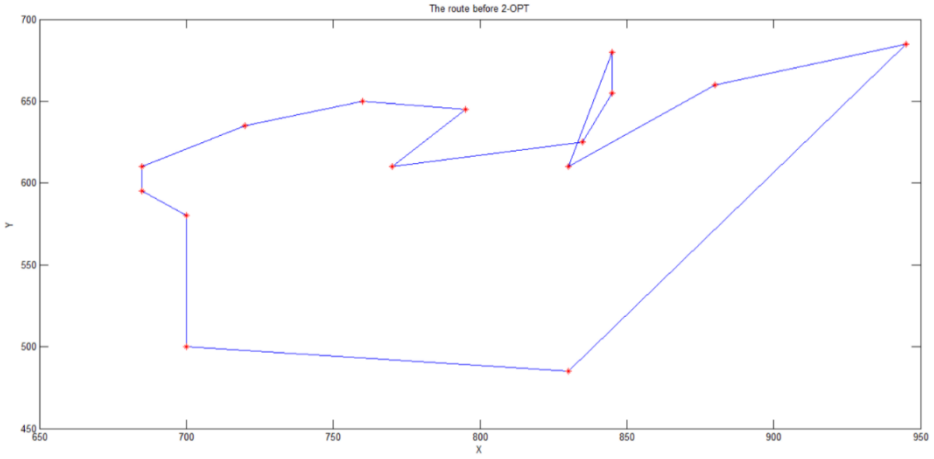

Figure 6. The route before 2-Opt method

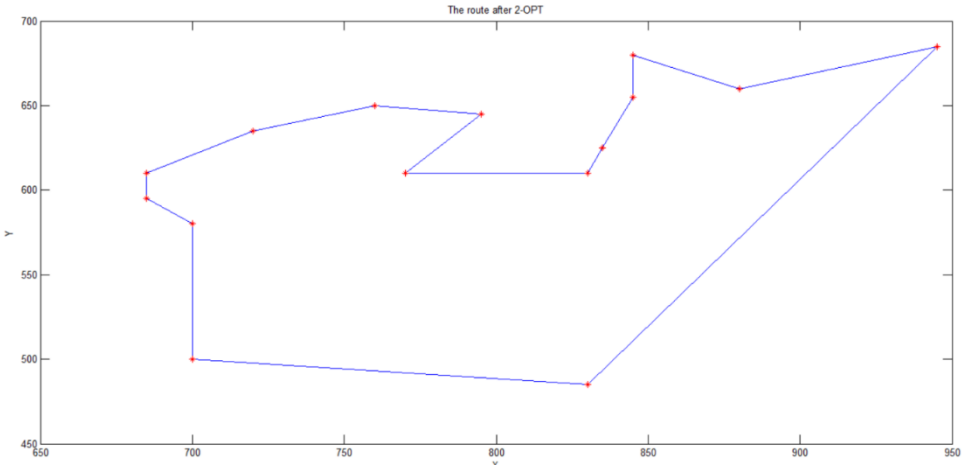

Figure 7. The route after 2-Opt method

2.4. Travelling Salesman Problem (TSP)

TSP has a route consisting of different cities. A random city is selected from the cities and a tour is created by visiting each city once. TSP aims to create the shortest route and the shortest distance. Let there be D cities that a salesperson has to visit. The maximum number of routes to follow in all cities $(D\text{-}1)!/2$ for D cities [21]. For example, there are 25 cities to visit, and the number of routes the algorithm will

review is considered to be ((25-1)!/2=24!/2). In this paper, we calculate the distance between cities m and n using Euclidean distance as follows:

$$distance_{m,n} = \sqrt{(a_m - a_n)^2 + (b_m - b_n)^2} \quad (3)$$

where $distance_{m,n}$ is the distance between city m and city n. a_m and a_n are a coordinates for city m and city n. b_m and b_n are b coordinates for city m and city n. We use the *func* function to calculate the entire tour length as follows [21]:

$$func = distance_{D,i} + \sum_{c=1}^{n-1} distance_{c,c+1} \quad (4)$$

D denotes the total number of cities. The fundamental goal is to find the Hamiltonian path with the lowest cost on a weighted graph.

2.5. TSP datasets used in the paper

TSPLIB contains the problems that are utilized to generate experimental findings [39]. The majority of the problems in TSPLIB have been solved, and the best values have been reported. The city numbers are indicated by the numbers in the problem names. These optimum values are used to make comparisons. Euclidean problems are the coordinated kinds of these problems (EUC). Some studies in the literature calculate other types of problems (for example, geographical-GEO) as EUC, which leads to errors in comparisons. OLIVER30, EIL51, Eil76, EIL101, BERLIN52, ST70, PR76, KROA100, KROB100, KROC100, KROD100, KROE100, KROB150, TSP225, CH150, and A280 datasets selected from the TSPLIB library are used to solve the TSP problem with the DPSO algorithm. There are sixteen low, medium, and high dimensional data sets commonly used in TSP. The optimum values of these TSP datasets are given in Table 1.

Table 1. The optimum values of these TSP datasets [20; 21].

ID	TSP Name	Dimension size (D=city size)	Optimum value
1	OLIVER30	30	423.74
2	EIL51	51	428.87
3	BERLIN52	52	7542 (7544.37)
4	ST70	70	677.11
5	EIL76	76	545.38
6	PR76	76	108159.44
7	KROA100	100	21282 (21285.44)
8	KROB100	100	22141
9	KROC100	100	20749
10	KROD100	100	21294
11	KROE100	100	22068
12	EIL101	101	642.31
13	KROB150	150	26130
14	CH150	150	6532.10
15	TSP225	225	3859
16	A280	280	2586.77

3. RESULTS AND DISCUSSION

The variations of the DPSO (DPSO1, DPSO2, DPSO3, DPSO4, DPSO5, and DPSO6) have occurred according to the selected method type (DPSO1 (swap method), DPSO2 (shift method), DPSO3 (swap and shift methods), DPSO4 (symmetry method), DPSO5 (swap, shift, and symmetry methods), and DPSO6 (swap, shift, symmetry, 2 opt methods)). The success of each method on DPSO is tested separately. In the variations of the DPSO results, the best fitness values (Min), the average of the fitness values (Avg), maximum (Max) fitness values, the standard deviation of the fitness values (Std), and the mean CPU time are stored. Error (%) values are presented to measure the performances of the algorithms. The relative error (Error) is determined as follows using Eq. 5. The resulting answer (mean of 20 different runs) is referred to as Result, while the optimum value of the problem is referred to as Optimum. To make the comparisons easier, Error (%) values are used, and the best results are noted in boldface font type. During performance measurements, the experiment set is run 20 times. All experiments are run on a Windows 7 using Intel(R) Core(TM) i5-2410M, 4 GB of RAM, and the codes are implemented in Matlab R2014a.

$$Error = \frac{Result - Optimum}{Optimum} \times 100 \tag{5}$$

The application of the developed methods (swap, shift, and symmetry methods) in DPSO has been investigated by experimental studies.

3.1. The Parameters of the DPSO

For determining the optimum parameters for DPSO, the number of the particle (N) and the number of candidate solutions (CS) are analyzed. Ten different population size values (10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 200, and 300) are tested on the BERLIN52, EIL76, and KROA100 TSPs to see the effect of population size (N) on performance (for DPSO1). The function evaluations (MaxFEs) and the number of candidate solutions (CS) are used as 800000 and 5, respectively. Obtained results are shown in Table 2. According to the results, the population size does not have a significant effect on the results. In many studies in the literature, population size is taken as equal to the total number of cities [17], [21]. In this study, the population size amount is chosen as 100.

Five different numbers of candidate solutions (CS) values (1, 5, 10, 15, and 20) are tested on the BERLIN52, EIL76, and KROA100 TSPs to see the effect of the number of candidate solutions (CS) on performance (for DPSO1). The MaxFEs and population size (N) parameters are used as 800000, and 100, respectively. Obtained results are shown in Table 3. According to the results, as the number of candidate solutions (CS) increases, the performance increases. The number of candidate solutions is determined as the most appropriate value in this study. Experiments are carried out with the given parameter values. All parameters values use in the study are shown in Table 4.

Table 2. Population size (N) analyses of DPSO1.

BERLIN52						
N	*Min*	*Max*	*Avg*	*Std*	*Mean Error(%)*	*Mean CPU time*
10	8040.00	8133.00	8100.00	38.12	7.84	26.92
20	8035.00	8133.00	8066.50	39.88	7.08	18.83
30	8032.00	8133.00	8075.50	46.40	6.54	18.08
40	8035.00	8133.00	8052.55	31.27	6.54	19.09
50	8032.00	8133.00	8065.90	39.62	6.60	16.96
60	8032.00	8133.00	8049.60	28.67	6.60	19.53
70	8032.00	8059.00	8037.80	5.49	6.85	20.64
80	8032.00	8066.00	8038.65	6.81	6.95	12.88
90	8035.00	8040.00	8037.25	2.49	6.54	14.04
100	8032.00	8040.00	8036.40	2.46	**6.50**	14.31
200	8035.00	8076.00	8039.55	8.71	6.60	16.08
300	7542.00	8063.00	8014.85	108.64	6.60	15.63
EIL76						
N	*Min*	*Max*	*Avg*	*Std*	*Mean Error(%)*	*Mean CPU time*
10	606.79	608.39	607.75	0.78	11.44	24.55
20	606.79	606.79	606.79	0.00	11.26	15.47
30	592.21	608.39	602.69	6.63	10.51	12.13
40	606.79	608.39	607.27	0.73	11.35	10.74
50	586.45	606.79	604.75	6.10	10.89	10.41
60	592.21	606.79	602.57	6.45	10.49	10.52
70	592.21	606.79	605.33	4.37	10.99	10.31
80	592.21	606.79	605.33	4.37	10.99	10.28
90	586.27	606.79	603.70	6.58	10.69	9.69
100	592.21	606.790	603.19	5.66	10.60	10.43
200	594.47	608.39	602.34	4.50	**10.44**	10.72
300	598.01	608.39	605.39	3.63	11	11.25
KROA100						
N	*Min*	*Max*	*Avg*	*Std*	*Mean Error(%)*	*Mean CPU time*
10	22261.64	22643.48	22560.04	100.89	6.01	26.1
20	22586.91	22864.92	22619.22	82.18	6.28	17.05
30	22586.91	22601.93	22589.66	5.52	6.14	13.72
40	22479.11	22652.94	22588.39	43.77	6.14	12.54
50	22453.36	22634.11	22587.11	47.18	6.13	12.2
60	22500.58	22861.06	22640.55	115.69	6.38	11.81
70	22274.08	22746.07	22505.67	127.51	**5.75**	11.37
80	22586.91	22712.84	22622.9	44.78	6.30	11.52
90	22586.91	22923.99	22658.08	93.69	6.47	11.00
100	22586.91	22984.92	22653.87	116.17	6.45	10.09
200	22586.91	23114.58	22839.32	142.92	7.32	10.71
300	22991.37	23349.15	23180.61	141.05	8.92	10.77

Table 3. The number of candidate solutions (CS) analyses of DPSO1.

BERLIN52							
CS	*Min*	*Max*	*Avg*	*Std*	*Mean Error(%)*	*Mean time*	*CPU*
1	8035.00	8063.00	8038.70	6.11	6.54	16.82	
5	8035.00	8130.00	8050.60	26.08	6.54	20.57	
10	8032.00	8076.00	8038.40	8.97	7.08	21.09	
15	8032.00	8040.00	8036.55	2.99	6.54	16.21	
20	8035.00	8076.00	8040.60	12.01	6.54	15.25	
EIL76							
CS	*Min*	*Max*	*Avg*	*Std*	*Mean Error(%)*	*Mean time*	*CPU*
1	593.48	608.39	605.62	4.07	11.04	14.77	
5	592.21	606.79	604.27	5.12	10.80	10.65	
10	588.07	606.81	602.42	6.93	10.46	9.90	
15	593.74	607.60	603.29	5.56	10.62	10.18	
20	592.87	607.60	603.41	5.55	10.64	9.81	
KROA100							
CS	*Min*	*Max*	*Avg*	*Std*	*Mean Error(%)*	*Mean time*	*CPU*
1	22465.06	22992.03	22619.06	139.62	6.28	16.42	
5	22575.24	22918.82	22702.58	124.21	6.68	11.57	
10	22465.06	22794.71	22649.42	96.35	6.43	10.89	
15	22500.58	23391.24	22720.08	239.07	6.76	10.83	
20	22601.33	22850.81	22663.12	70.40	6.49	10.52	

Table 4. The sets of parameters of the variations of the DPSO.

Parameters	**Values**
Number of particles	100
MaxFEs	800000
k_1 (social constant)	0.2
k_2 (cognitive constant)	0.2
w (inertia weight)	0.4
Vel_{max} (maximum velocity)	0.8
The number of candidate solutions	20

3.2. The Comparison of the Variations of the DPSO

The variations of the DPSO (DPSO1, DPSO2, DPSO3, DPSO4, DPSO5, and DPSO6) have occurred according to the selected method type (DPSO1 (swap method), DPSO2 (shift method), DPSO3 (swap and shift methods), DPSO4 (symmetry methods), DPSO5 (swap, shift, and symmetry methods), and DPSO6 (swap, shift, symmetry, 2 opt methods)). The success of each method on DPSO has been tested separately. The performance improvement of each method over the result of DPSO is shown in detail. Parameter settings in Table 4 are used in the tests. Minimum result (Min), maximum result (Max), mean result (Avg), the standard deviation of results (Std), mean CPU time, and error rate results have been calculated for all DPSO variations. The results of the DPSO1, DPSO2, DPSO3, DPSO4, DPSO5, and DPSO6 are shown in Tables 5, 6, 7, 8, 9, and 10, respectively. Table 11 shows the comparison results for all DPSO variations (for

error rate (%)). Table 12 shows the comparison results for all DPSO variations (for mean CPU time). Figure 8 shows the convergence graph of mean error (%) for all methods.

According to Table 11, the most successful DPSO variations are DPSO5 and DPSO6. While DPSO5 achieves success in 9 of 16 TSP benchmarks, DPSO6 achieves successful results in 8 of 16 TSP benchmarks. The symmetry method provides a noticeable performance increase in the result. Swap, shift, and symmetry methods have improved DPSO to achieve optimum results. The 2-opt algorithm is a local search algorithm. DPSO6 is obtained as a result of running the 2-opt algorithm on the optimum route. It is noteworthy that the results of some TSP benchmarks have improved with the 2-opt algorithm (OLIVER30, KROB150, CH150, TSP225, A280, etc.). The results show that the application of many different methods to the result increases the result performance.

Table 5. The performance analysis of DPSO1 on TSPs (swap method).

ID	TSP Name	Min	Max	Avg	Std	Mean CPU time
1	OLIVER30	425.27	435.07	433.04	3.89	15.08
2	EIL51	470.58	485.64	482.80	4.47	24.30
3	BERLIN52	8032.00	8077.00	8039.75	12.41	25.20
4	ST70	721.68	728.39	724.04	1.82	21.85
5	EIL76	592.21	606.81	604.46	4.73	18.08
6	PR76	125407.46	127838.70	126536.58	683.54	15.59
7	KROA100	22530.16	22716.90	22615.91	39.16	16.25
8	KROB100	24071.69	25046.60	24554.27	280.58	16.90
9	KROC100	23096.50	23190.43	23152.98	21.41	20.69
10	KROD100	24410.01	24774.30	24659.44	90.02	23.65
11	KROE100	23633.66	24084.27	23841.04	106.15	20.44
12	EIL101	719.19	722.66	720.91	1.11	31.36
13	KROB150	29808.01	30198.10	29952.17	96.08	18.69
14	CH150	6923.70	6969.94	6943.05	11.84	17.43
15	TSP225	4549.02	4598.76	4576.86	13.13	20.71
16	A280	3028.03	3081.84	3063.75	14.21	23.02

Table 6. The performance analysis of DPSO2 on TSPs (shift method).

ID	TSP Name	Min	Max	Avg	Std	Mean CPU time
1	OLIVER30	425.27	465.25	451.21	14.57	11.32
2	EIL51	451.09	470.43	458.86	6.56	9.62
3	BERLIN52	7542.00	7954.00	7857.35	123.74	9.68
4	ST70	715.95	725.83	721.42	2.71	8.31
5	EIL76	575.78	583.07	578.04	2.56	8.45
6	PR76	119750.16	122922.17	120272.00	925.37	8.40
7	KROA100	22664.62	23606.64	22969.30	289.26	8.95
8	KROB100	24003.06	24577.97	24273.99	166.30	8.92
9	KROC100	21866.45	22272.47	22030.39	112.44	9.07
10	KROD100	23619.72	24578.48	24299.20	337.92	9.47
11	KROE100	23493.46	23787.04	23626.89	83.83	9.13
12	EIL101	706.88	711.55	707.98	1.21	9.11
13	KROB150	29680.74	30082.38	29919.49	103.96	10.14
14	CH150	6847.14	6921.75	6875.98	17.79	9.72
15	TSP225	4386.69	4495.04	4452.49	24.30	11.39
16	A280	3027.92	3054.05	3041.06	6.85	12.26

Table 7. The performance analysis of DPSO3 on TSPs (swap and shift methods).

ID	TSP Name	Min	Max	Avg	Std	Mean CPU time
1	OLIVER30	425.27	435.07	432.37	4.10	15.74
2	EIL51	444.13	463.04	453.57	4.44	16.47
3	BERLIN52	7542.00	7938.00	7879.60	81.35	16.63
4	ST70	715.95	721.63	718.39	2.05	17.91
5	EIL76	575.78	582.49	577.95	2.40	19.34
6	PR76	117564.11	119986.96	119070.04	900.90	19.91
7	KROA100	22332.50	22911.56	22542.78	170.54	20.52
8	KROB100	23841.97	24370.47	24016.02	132.10	19.90
9	KROC100	21840.52	22191.96	22012.09	108.50	27.38
10	KROD100	23450.54	24577.59	24067.38	388.39	26.29
11	KROE100	22970.94	23558.86	23181.96	156.66	33.16
12	EIL101	702.77	707.29	705.71	1.30	23.87
13	KROB150	29243.13	29480.60	29406.41	67.71	34.72
14	CH150	6817.49	6873.62	6841.92	17.50	42.44
15	TSP225	4383.50	4461.97	4423.66	20.90	49.68
16	A280	2952.77	3030.44	2997.28	20.00	48.70

Table 8. The performance analysis of DPSO4 on TSPs (symmetry methods).

ID	TSP Name	Min	Max	Avg	Std	Mean CPU time
1	OLIVER30	448.74	449.17	449.11	0.16	9.69
2	EIL51	451.19	467.79	455.11	4.88	10.95
3	BERLIN52	7679.00	7713.00	7694.20	13.83	8.74
4	ST70	733.01	742.31	740.88	3.14	12.18
5	EIL76	573.37	584.62	578.54	3.67	14.02
6	PR76	114508.51	122436.35	116638.10	1959.10	10.50
7	KROA100	23309.00	23629.08	23445.80	69.49	9.27
8	KROB100	23713.21	24608.14	24113.70	202.28	10.51
9	KROC100	22272.29	22736.49	22601.11	138.05	9.26
10	KROD100	22632.36	23514.86	23031.71	258.70	9.06
11	KROE100	23162.77	23366.24	23236.92	42.65	9.04
12	EIL101	684.04	703.93	687.23	4.68	9.11
13	KROB150	28250.12	29012.15	28437.25	215.15	12.88
14	CH150	6865.71	6897.39	6871.80	10.42	16.39
15	TSP225	4169.75	4312.46	4238.54	38.59	15.81
16	A280	2850.60	2946.29	2899.01	26.92	15.59

Table 9. The performance analysis of DPSO5 on TSPs (swap, shift, and symmetry methods).

ID	TSP Name	Min	Max	Avg	Std	Mean CPU time
1	OLIVER30	423.74	434.61	425.71	2.80	26.96
2	EIL51	431.17	440.60	434.68	3.31	22.72
3	BERLIN52	7542.00	7542.00	7542.00	0.00	22.67
4	ST70	677.19	708.38	688.14	7.14	23.69
5	EIL76	562.14	581.02	568.06	4.78	23.30
6	PR76	109686.79	116262.26	112127.67	1734.91	26.84
7	KROA100	21294.40	21910.66	21440.43	173.54	24.22
8	KROB100	22323.35	23331.36	22696.48	213.23	30.72
9	KROC100	21185.42	21904.06	21498.03	178.89	38.58
10	KROD100	21712.39	23286.89	22183.73	325.28	33.13
11	KROE100	22208.15	22614.14	22388.35	112.23	25.43
12	EIL101	660.91	677.49	669.04	4.18	28.18
13	KROB150	26628.34	27513.26	27021.26	223.87	33.99
14	CH150	6565.05	6710.59	6635.86	44.40	41.96
15	TSP225	4100.57	4218.32	4158.55	32.94	43.42
16	A280	2763.18	2895.39	2836.20	34.08	42.83

Table 10. The performance analysis of DPSO6 on TSPs (swap, shift, symmetry, and 2opt methods).

ID	TSP Name	Min	Max	Avg	Std	Mean CPU time
1	OLIVER30	423.74	425.27	424.66	0.75	24.04
2	EIL51	431.17	443.47	436.76	4.13	26.92
3	BERLIN52	7542.00	7542.00	7542.00	0.00	22.82
4	ST70	686.62	699.77	688.62	2.66	25.27
5	EIL76	559.33	573.39	565.85	4.07	26.25
6	PR76	109352.49	116314.51	112372.04	1835.62	25.45
7	KROA100	21294.40	21907.55	21476.41	216.52	31.30
8	KROB100	22581.92	23061.09	22749.21	122.01	31.99
9	KROC100	21234.64	21967.34	21449.68	179.66	31.03
10	KROD100	21801.09	23474.44	22293.46	393.48	35.23
11	KROE100	22161.11	22706.73	22438.09	120.84	32.82
12	EIL101	657.28	679.33	669.39	4.38	34.09
13	KROB150	26527.84	27536.51	26919.52	251.46	33.75
14	CH150	6557.63	6697.65	6622.86	44.58	28.86
15	TSP225	4001.42	4132.14	4057.05	38.00	35.31
16	A280	2653.03	2833.05	2758.15	39.80	39.10

Table 11. The performance analysis of variations of the DPSO on TSPs (according to mean error (%)).

ID	TSP Name	DPSO1	DPSO2	DPSO3	DPSO4	DPSO5	DPSO6
1	OLIVER30	2.19	6.48	2.04	5.99	0.46	**0.22**
2	EIL51	12.57	6.99	5.76	6.12	**1.35**	1.84
3	BERLIN52	6.60	4.18	4.48	2.02	**0.00**	**0.00**
4	ST70	6.93	6.54	6.10	9.42	**1.63**	1.70
5	EIL76	10.83	5.99	5.97	6.08	4.16	**3.75**
6	PR76	16.99	11.20	10.09	7.84	**3.67**	3.89
7	KROA100	6.27	7.93	5.92	10.17	**0.74**	0.91
8	KROB100	10.90	9.63	8.47	8.91	**2.51**	2.75
9	KROC100	11.59	6.18	6.09	8.93	3.61	**3.38**
10	KROD100	15.80	14.11	13.02	8.16	**4.18**	4.69
11	KROE100	8.03	7.06	5.05	5.30	**1.45**	1.68
12	EIL101	12.24	10.22	9.87	6.99	**4.16**	4.22
13	KROB150	14.63	14.50	12.54	8.83	3.41	**3.02**
14	CH150	6.29	5.26	4.74	5.20	1.59	**1.39**
15	TSP225	18.60	15.38	14.63	9.84	7.76	**5.13**
16	A280	18.44	17.56	15.87	12.07	9.64	**6.63**

Table 12. The performance analysis of variations of the DPSO on TSPs (according to mean CPU time).

ID	TSP Name	DPSO1	DPSO2	DPSO3	DPSO4	DPSO5	DPSO6
1	OLIVER30	15.08	11.32	15.74	9.69	26.96	24.04
2	EIL51	24.30	9.62	16.47	10.95	22.72	26.92
3	BERLIN52	25.20	9.68	16.63	8.74	22.67	22.82
4	ST70	21.85	8.31	17.91	12.18	23.69	25.27
5	EIL76	18.08	8.45	19.34	14.02	23.30	26.25
6	PR76	15.59	8.40	19.91	10.50	26.84	25.45
7	KROA100	16.25	8.95	20.52	9.27	24.22	31.30
8	KROB100	16.90	8.92	19.90	10.51	30.72	31.99
9	KROC100	20.69	9.07	27.38	9.26	38.58	31.03
10	KROD100	23.65	9.47	26.29	9.06	33.13	35.23
11	KROE100	20.44	9.13	33.16	9.04	25.43	32.82
12	EIL101	31.36	9.11	23.87	9.11	28.18	34.09
13	KROB150	18.69	10.14	34.72	12.88	33.99	33.75
14	CH150	17.43	9.72	42.44	16.39	41.96	28.86
15	TSP225	20.71	11.39	49.68	15.81	43.42	35.31
16	A280	23.02	12.26	48.70	15.59	42.83	39.10

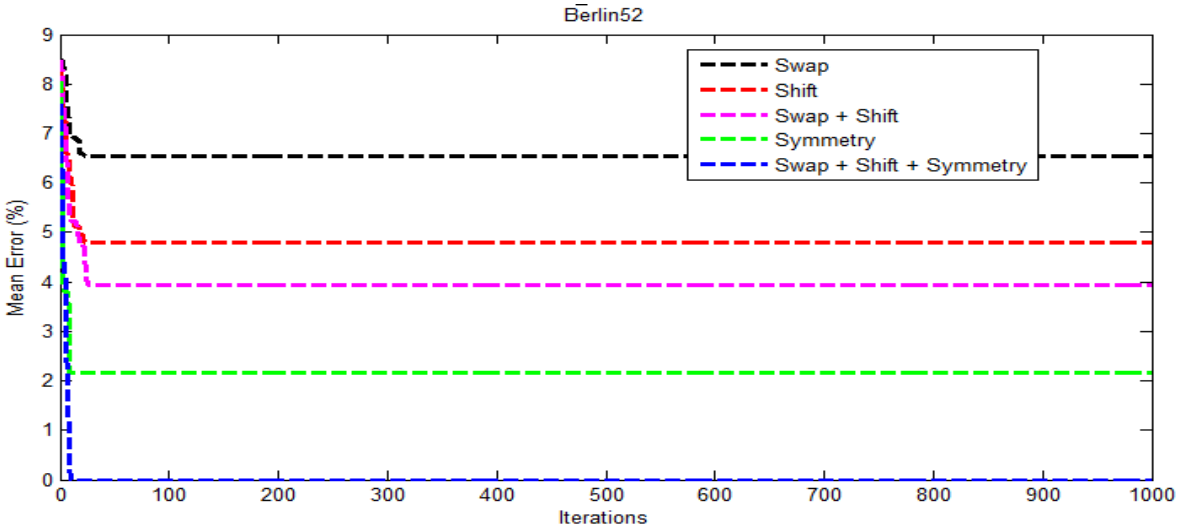

Figure 8. Convergence graph of mean error (%) for all methods.

3.3. The comparison of the variations of the DPSO with DJAYA, SA, ACO, STA, and DTSA

The first comparison is performed with Chunhua et al., Cinar et al., and Gunduz and Aslan [20], [21], [34]. In this study, the DPSO is compared with DJAYA, STA, SA, ACO, and DTSA on the BERLIN52 problem. Comparison results of STA, SA, and ACO are directly taken from Chunhua et al., DTSA is directly taken from Cinar et al., and DJAYA is directly taken from Gunduz and Aslan. For a fair comparison, MaxFEs is set to 4000 for all methods. The parameter settings are shown in Table 13. Comparison results are shown in Table 14. According to the minimum results (Min), DPSO, DJAYA, and DTSA have reached the optimum results. According to the average results (Avg), DPSO has performed better than DTSA, DJAYA, STA, SA, and ACO algorithms. The reason for this success is due to the methods developed by DPSO on a randomly generated route.

Table 13. The sets of parameters of DPSO, STA, SA, DTSA, DJAYA, and ACO.

Parameters	Population size	MaxFEs	Other parameters
STA	20	4000	Temperature=50000; cooling rate=0.97
SA	20	4000	
ACO	20	4000	α= 1, β = 5, ϱ = 0.9
DJAYA	20	4000	
DTSA	20	4000	NS=6; ST=0.5
DPSO	20	4000	CS=20

Table 14. The Comparison of DPSO, SA, ACO, STA, DTSA, and DJAYA on BERLIN52 TSP.

TSP name	Algorithm	Min	Max	Avg	Std
BERLIN52	SA	8186.40	9585.80	8983.80	380.10
	ACO	8240.40	9151.30	8777.60	267.11
	STA	7544.40	8630.50	8247.20	273.45
	DTSA	**7542.00**	7929.00	7689.17	108.40
	DJAYA	**7542.00**	7831.00	7668.35	112.50
	DPSO5	7725.00	7905.00	7823.40	50.70
	DPSO6	**7542.00**	7804.00	**7637.55**	95.89

3.4. The comparison of the variations of the DPSO with DJAYA, SA, DSTA0, DSTAI, DSTAII, and DTSA

The second comparison is performed with Zhou et al., Cinar et al., and Gunduz and Aslan [20], [21], [35]. In this study, the DPSO is compared with DJAYA, SA, DSTA0, DSTAI, DSTAII, and DTSA on KROA100, KROB100, KROC100, KROD100, and KROE100 problems. Comparison results of SA, DSTA0, DSTAI, and DSTAII are directly taken from Zhou et al., DTSA is directly taken from Cinar et al., and DJAYA is directly taken from Gunduz and Aslan. MaxFEs is set to 90000 for all methods to ensure a fair comparison. Initial temperature and cooling rate are the algorithmic parameters of SA, and they are set to 2000 and 0.97, respectively. The maximum iteration number for DSTA variations is set to 900 and the search enforcement is set to 100. Comparison results are shown in Table 15. According to the results, DSTAII and DPSO have performed better than other algorithms.

3.5. The comparison of the variations of the DPSO with DJAYA, ACO, ABC, HA, and DTSA

The other comparison is performed with Gündüz et al., Cinar et al., and Gunduz and Aslan [20], [21], [36]. In this study, the DPSO is compared with DJAYA, ACO, ABC, HA, and DTSA on EIL51, EIL76, EIL101, KROA100, OLIVER30, BERLIN52, ST70, PR76, CH150, and TSP225 problems. Comparison results of ACO, ABC, and HA are directly taken from Gündüz et al., DTSA is directly taken from Cinar et al., and DJAYA is directly taken from Gunduz and Aslan. MaxFEs is set to D×500 for all methods to ensure a fair comparison (D=the number of cities of the TSP). But, the population size and limit parameters of ABC are set to D and D×D×500, respectively. ACO's specific parameters are $\alpha = 1$, $\beta = 5$, and $\varrho = 0.65$. For ACO, the number of ants is set to D, and the maximum number of iterations is set to 500. HA stands for 50% ABC and 50% ACO, which implies the parameter values are the same, but the population sizes are half of D. Comparison results are shown in Table 16.

According to the results, DPSO has achieved more successful results than other algorithms except HA. Out of 10 TSP benchmark datasets, DPSO has performed well in 4 (EIL51, PR76, KROA100, EIL101) of them. DJAYA has outperformed 2 (CH150 and TSP225) out of 10 TSP benchmark datasets. HA has been shown superior success in the 4 TSP benchmark dataset. ACO has been shown superior success in 1 TSP benchmark dataset. This has proven that DPSO has been developed and is an alternative discrete optimization algorithm in the literature.

Table 15. Comparison of DPSO with DJAYA, SA, DSTA0, DSTAI, DSTAII, and DTSA.

TSP name	Algorithm	Avg	Std	Error(%)	Rank
	SA	22635.00	778.72	6.36	5
	DSTA0	23213.00	906.11	9.07	7
	DSTAI	22835.00	715.85	7.30	6
KROA100	***DSTAII***	21767.00	221.64	2.28	4
	DTSA	21506.78	260.55	1.06	2
	DJAYA	21705.98	290.22	1.99	3
	DPSO6	**21434.36**	**178.50**	**0.72**	**1**
	SA	23657.00	445.78	6.85	5
	DSTA0	23794.00	517.05	7.47	7
	DSTAI	23734.00	507.38	7.19	6
KROB100	***DSTAII***	**22880.00**	302.14	**3.34**	**1**
	DTSA	23139.26	181.74	4.51	4
	DJAYA	22973.73	234.79	3.76	3
	DPSO6	23334.86	**158.64**	5.39	2
	SA	22223.00	522.20	7.10	6
	DSTA0	22877.00	709.87	10.26	7
	DSTAI	21891.00	536.88	5.50	5
KROC100	***DSTAII***	**21378.00**	246.34	**3.03**	**1**
	DTSA	21817.08	217.77	5.15	4
	DJAYA	21702.02	186.32	4.59	2
	DPSO6	21703.96	**40.20**	4.60	3
	SA	22911.00	483.01	7.59	5
	DSTA0	23043.00	565.80	8.21	7
	DSTAI	22665.00	592.53	6.44	3
KROD100	***DSTAII***	**21991.00**	**315.32**	**3.27**	**1**
	DTSA	22972.26	390.50	7.88	6
	DJAYA	22631.25	487.62	6.28	2
	DPSO6	22834.33	375.98	7.23	4
	SA	23125.00	389.42	4.79	5
	DSTA0	23738.00	450.82	7.57	7
	DSTAI	23371.00	678.69	5.90	6
KROE100	***DSTAII***	22637.00	166.82	2.58	4
	DTSA	22547.00	121.96	2.17	2
	DJAYA	22582.47	252.07	2.33	3
	DPSO6	**22545.57**	**87.98**	**2.16**	**1**

Table 16. The comparison of DPSO with DJAYA, ACO, ABC, HA, and DTSA.

TSP name	Algorithm	Avg	Std	Error(%)	Rank
OLIVER30	*ACO*	424.68	1.41	0.22	2
	ABC	462.55	12.47	9.16	6
	HA	**423.74**	**0.00**	**0.00**	**1**
	DTSA	428.50	4.21	1.12	5
	DJAYA	426.88	2.74	0.74	4
	DPSO6	425.28	0.05	0.36	3
EIL51	*ACO*	457.86	4.07	6.76	5
	ABC	457.86	4.07	6.76	5
	HA	443.39	5.25	3.39	3
	DTSA	443.93	4.04	3.51	4
	DJAYA	440.18	4.95	2.64	2
	DPSO6	**436.11**	**3.18**	**1.69**	**1**
BERLIN52	*ACO*	7659.31	38.70	1.52	5
	ABC	10390.26	439.69	37.72	6
	HA	7544.37	**0.00**	**0.00**	**1**
	DTSA	7545.83	21.00	0.02	3
	DJAYA	7580.30	80.60	0.48	4
	DPSO6	7542.50	1.50	0.01	2
ST70	*ACO*	709.16	8.27	4.73	5
	ABC	1230.49	41.79	81.73	6
	HA	**7544.37**	**0.00**	**0.00**	**1**
	DTSA	7545.83	21.00	0.02	2
	DJAYA	702.30	9.56	3.72	4
	DPSO6	700.53	9.14	3.46	3
EIL76	*ACO*	561.98	3.50	3.04	2
	ABC	931.44	24.86	70.78	6
	HA	**557.98**	4.10	**2.31**	**1**
	DTSA	578.58	3.93	6.09	5
	DJAYA	573.17	6.33	5.10	4
	DPSO6	572.98	**2.60**	5.06	3
PR76	*ACO*	116321.22	885.79	7.55	5
	ABC	205119.61	7379.16	89.65	6
	HA	115072.29	742.90	6.39	4
	DTSA	114930.03	1545.64	6.26	3
	DJAYA	113258.29	1711.93	4.71	2
	DPSO6	**112937.44**	**681.34**	**4.42**	**1**
KROA100	*ACO*	22880.12	235.18	7.49	5
	ABC	53840.03	2198.36	152.94	6
	HA	22435.31	231.34	5.40	4
	DTSA	21728.40	358.13	2.08	2
	DJAYA	21735.31	331.33	2.13	3
	DPSO6	**21599.07**	**190.63**	**1.49**	**1**
EIL101	*ACO*	693.42	6.80	7.96	5
	ABC	1315.95	35.28	104.88	6
	HA	683.39	6.56	6.40	3
	DTSA	689.91	4.47	7.41	4
	DJAYA	677.37	4.87	5.46	2
	DPSO6	**677.34**	**0.58**	**5.45**	**1**
CH150	*ACO*	6702.87	20.73	2.61	3
	ABC	21617.48	453.71	230.93	6
	HA	6677.12	**19.30**	2.22	2
	DTSA	6748.99	32.63	3.32	5
	DJAYA	**6638.63**	52.79	**1.63**	**1**
	DPSO6	6723.35	33.01	2.93	4
TSP225	*ACO*	4176.08	**28.34**	8.22	4
	ABC	17955.12	387.35	365.28	6
	HA	4157.85	26.27	7.74	3
	DTSA	4230.45	58.76	9.93	5
	DJAYA	**4095.02**	42.54	**6.12**	**1**
	DPSO6	4152.81	43.42	7.61	2

3.6. The comparison of the variations of the DPSO with DJAYA, ACO, GA, BH, and DTSA

The last comparison is performed with Hatamlou, Cinar et al., and Gunduz and Aslan [20], [21], [37]. In this study, the DPSO is compared with DJAYA, ACO, GA, BH, and DTSA on EIL51, EIL76, EIL101, BERLIN52, and ST70 problems. Comparison results of, ACO, GA, and BH are directly taken from Hatamlou, DTSA is directly taken from Cinar et al., and DJAYA is directly taken from Gunduz and Aslan. MaxFEs and population size (N) are set to 20000 and 100 for all methods to ensure a fair comparison. ST is set to 0.5 for DTSA. The ACO parameters are $\alpha = 1.5$, $\beta = 2$, and $\varrho = 0.7$. For ACO, the number of ants is set to 100, while the maximum number of iterations is set to 200. Comparison results are shown in Table 17. According to the results, DPSO has been achieved more successful results than other heuristic algorithms in 4 of 5 different TSP datasets. DJAYA, on the other hand, has been showed superior success in 2 of 5 different TSP datasets. According to the results, DJAYA and DPSO outperform other heuristic algorithms. DPSO owes this outstanding success to symmetry and 2-opt methods developed for local search.

Table 17. The comparison of DPSO with DJAYA, ACO, GA, BH, and DTSA.

TSP name	Algorithm	Avg	Std	Rank
EIL51	***ACO***	461.0175	6.2974	6
	GA	453.4773	9.4157	3
	BH	458.9252	38.6365	5
	DTSA	456.5184	8.9247	4
	DJAYA	440.4394	**3.1055**	2
	DPSO6	**439.11**	3.23	**1**
EIL76	***ACO***	594.1442	40.2152	4
	GA	652.0593	122.0972	5
	BH	659.1021	152.1754	6
	DTSA	588.0623	5.7296	3
	DJAYA	**574.4803**	5.6710	**1**
	DPSO6	575.07	**3.89**	2
EIL101	***ACO***	763.9207	59.9684	4
	GA	838.8307	9.9642	5
	BH	897.3813	210.1446	6
	DTSA	689.8384	7.2994	3
	DJAYA	686.8843	**6.0664**	2
	DPSO6	**685.92**	6.23	**1**
BERLIN52	***ACO***	8522.9017	1152.2000	4
	GA	9288.4483	1301.2108	5
	BH	8455.8304	508.9871	3
	DTSA	7761.6000	**62.8594**	2
	DJAYA	**7627.0000**	120.3869	**1**
	DPSO6	**7627.0000**	103.89	**1**
ST70	***ACO***	757.7540	59.6079	4
	GA	1158.8458	52.1734	6
	BH	797.5745	125.2272	5
	DTSA	710.4037	2.7956	3
	DJAYA	707.2151	15.3049	2
	DPSO6	**707.08**	**1.84**	**1**

3.7. Discussion

The PSO has been updated for discrete optimization problems, and the results of DPSO have been examined in this study on TSPs of 16 different sizes. Four different methods have been proposed to improve the performance of DPSO (swap, shift, symmetry, and 2-opt methods). Various DPSO variations are obtained according to the addition of each method to DPSO. Thus, the success of each method on DPSO is examined in detail. The symmetry method provides a noticeable performance increase in the result. Swap, shift, and symmetry methods have improved DPSO to achieve optimum results. DPSO6 is obtained as a result of running the 2-opt algorithm on the optimum route. When DPSO is compared with the literature, it is a remarkable success. Thus, the success of the proposed methods for DPSO has been proven.

4. CONCLUSIONS

PSO is a heuristic algorithm based on swarm intelligence developed to solve continuous optimization problems. Due to its success in solving continuous optimization problems, it has often been preferred by many researchers in the literature for solving real-world problems. But real-world problems do not always consist of continuous problems. Sometimes real-world problems are problems involving independent variables. Such problems are called discrete optimization problems. TSP is a discrete optimization problem that is frequently used in the literature to measure the success of discrete optimization algorithms. In this study, a Discrete PSO (DPSO) is proposed. In order to increase the success of DPSO, new methods have been added in the new candidate particle generation stage. These methods are swap, shift, symmetry, and 2-OPT methods. Although the swap, shift, and symmetry methods are frequently used in the literature, they are used for the first time in DPSO. The symmetry method is a new and successful method. The variations of the DPSO have occurred according to the selected method type (DPSO1 (swap method), DPSO2 (shift method), DPSO3 (swap and shift methods), DPSO4 (symmetry method), DPSO5 (swap, shift, and symmetry methods), DPSO6 (swap, shift, symmetry, and 2-opt methods)). The performance increase of each method on DPSO is examined in detail. Thus, the contribution of each method to the performance of the DPSO is shown. The performance of DPSO is studied on sixteen different TSP datasets with low and high scales. The performance of DPSO is compared with the performances of DJAYA, DTSA, SA, DSTA0, DSTAI, DSTAII, ACO, ABC, HA, ACO, GA, and BH which are recently proposed new discrete optimization algorithms in the literature. Since DPSO is a well-established heuristic algorithm in the literature, it has competed with the newly proposed discrete algorithms. DPSO has excelled in many TSPs. DPSO's success is thanks to the swap, shift, and symmetry methods that it has developed the ability to search locally and globally. In future studies, the success of DPSO is thought to be demonstrated in different discrete optimization problems such as the knapsack problem and discrete real-world problems.

Declaration of Ethical Standards

Not applicable.

Credit Authorship Contribution Statement

Emine BAŞ: Conceptualization, Investigation, Methodology, Software, Writing – review, Original draft & editing.

Gülnur YILDIZDAN: Conceptualization, Investigation, Methodology, Writing - review, Software, Original draft & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding / Acknowledgements

This study was not funded by any institution.

Data Availability

The dataset used during the current study was obtained from the TSPLIB library (http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/).

5. REFERENCES

[1] R. Eberhard, J. Kennedy, "A New Optimizer Using Particle Swarm Theory," Proceedings of 1995 IEEE 6th International Symposium, pp. 39 – 43.

[2] M. Mahi, Ö.K. Baykan, H. Kodaz, "A new hybrid method based on Particle Swarm Optimization, Ant Colony Optimization and 3-Opt algorithms for Traveling Salesman Problem," *Applied Soft Computing*, vol. 30, pp. 484-490, 2015.

[3] Ş. Öztürk, R. Ahmad, N. Akhtar, "Variants of Artificial Bee Colony algorithm and its applications in medical image processing," *Applied Soft Computing*, vol. 97, Part A, pp. 106799, 2020.

[4] F. A. Hashim, A. G. Hussien, "Snake Optimizer: A novel meta-heuristic optimization algorithm," *Knowledge-Based Systems*, vol. 242, pp. 108320, 2022.

[5] T. S. L. V. Ayyarao, N. S. S. Ramakrishna, R. M. Elavarasan, N. Polumahanthi, M. Rambabu, G. Saini, B.Khan, B.Alatas, "War Strategy Optimization Algorithm: A New Effective Metaheuristic Algorithm for Global Optimization," *in IEEE Access*, vol. 10, pp. 25073-25105, 2022.

[6] D. Połap, M. Woźniak, "Red fox optimization algorithm," *Expert Systems with Applications*, vol. 166, pp. 114107, 2021.

[7] L. Yi, "Study on an Improved PSO Algorithm and its Application for Solving Function Problem," *International Journal of Smart Home*, vol. 10, no. 3, pp. 51 – 62, 2016.

[8] W. Deng, R. Chen, B. He, Y.Q. Liu, L. F. Yin, J. H. Guo, "A novel two-stage hybrid swarm intelligence optimization algorithm and application," *Soft Computing*, vol. 16, no. 10, pp. 1707-1722, 2012.

[9] W. Deng, H. M. Zhao, J. J. Liu, X. L. Yan, Y. Y. Li, L. F. Yin, C. H. Ding, "An improved CACO algorithm based on adaptive method and multi-variant strategies," *Soft Computing*, vol. 19 no. 3, pp. 701- 713, 2015.

[10] X. H. Shi, Y. Zhou, L. M. Wang, Q. X. Wang, Y. C. Liang, "A Discrete Particle Swarm Optimization Algorithm for Travelling Salesman Problem," Computational Methods, 2006, pp. 1063–1068.

[11] O. E. Turgut, M. S. Turgut, M. T. Coban, "Chaotic quantum behaved particle swarm optimization algorithm for solving nonlinear system of equations," *Computers and Mathematics with Applications*, vol. 68, no. 4, pp. 508-530, 2014.

[12] Z. L. Gaing, "Discrete particle swarm optimization algorithm for unit commitment," 2003 IEEE Power Engineering Society General Meeting (IEEE Cat. No.03CH37491), 2003, pp. 418-424.

[13] A. Unler, A. Murat, "A discrete particle swarm optimization method for feature selection in binary classification problems," vol. 206, no. 3, pp. 528-539, 2010.

[14] S. Strasser, R. Goodman, J. Sheppard, S. Butcher, "A New Discrete Particle Swarm Optimization Algorithm," GECCO '16: Proceedings of the Genetic and Evolutionary Computation Conference 2016, 2016, pp. 53–60.

[15] Q. K. Pan, M. F. Tasgetiren, Y. C. Liang, "A discrete particle swarm optimization algorithm for the

no-wait flowshop scheduling problem," vol. 35, no. 9, pp. 2807-2839, 2008.
[16] H. Izakian, B. T. Ladani, A. Abraham, V. Snasel, "A Discrete Particle Swarm Optimization Approach For Grid Job Scheduling," *International Journal of Innovative Computing, Information and Control,* vol. 6, no. 9, pp. 1-09, 2010.
[17] E. Baş, E. Ülker, "Dıscrete socıal spıder algorıthm for the travelıng salesman Problem," *Artificial Intelligence Review*, vol. 54, pp. 1063–1085, 2021.
[18] M. A. Al-Furhud, Z. H. Ahmed, "Genetic Algorithms for the Multiple Travelling Salesman Problem," *(IJACSA) International Journal of Advanced Computer Science and Applications*, vol. 11, no. 7, 2020.
[19] K. Panwar, K. Deep, "Discrete Grey Wolf Optimizer for symmetric travelling salesman problem," *Applied Soft Computing*, vol. 105, pp. 107298, 2021.
[20] M. Gunduz, M. Aslan, "DJAYA: A discrete Jaya algorithm for solving traveling salesman problem," *Applied Soft Computing*, vol. 105 pp. 107275, 2021.
[21] A. C. Cinar, S. Korkmaz, M. S. Kiran, "A discrete tree-seed algorithm for solving symmetric traveling salesman Problem," *Engineering Science and Technology*, vol. 23, pp. 879–890, 2020.
[22] E. Osaba, J. D. Ser, A. Sadollah, M. N. Bilbao, D. Camacho, "A discrete water cycle algorithm for solving the symmetric and asymmetric traveling salesman problem," vol. 71, pp. 277-290, 2018.
[23] S. S. Choong, L. P. Wong, C. P. Lim, "An artificial bee colony algorithm with a Modified Choice Function for the traveling salesman problem," vol. 44, pp. 622-635, 2019.
[24] F. Dahan, K. El Hindi, H. Mathkour, H. AlSalman, "Dynamic Flying Ant Colony Optimization (DFACO) for Solving the Traveling Salesman Problem," *Sensors,* vol. *19*, no. 8, pp. 1837, 2019.
[25] Y. Zhong, J. Lin, L. Wang, H. Zhang, "Hybrid discrete artificial bee colony algorithm with threshold acceptance criterion for traveling salesman problem," *Information Sciences,* vol. 421, pp. 70-84, 2017.
[26] W. Gao, "New Ant Colony Optimization Algorithm for the Traveling Salesman Problem," vol. 13, no. 1, pp. 44 – 55, 2020.
[27] X. Dong, Y. Cai, "A novel genetic algorithm for large scale colored balanced traveling salesman problem," vol. 95, pp. 727-742, 2019.
[28] N. Rokbani, R. Kumar, A. Abraham, A. M. Alimi, H. V. Long, S. Priyadarshini, L. H. Son, "Bi-heuristic ant colony optimization-based approaches for traveling salesman problem," *Soft Computing,* vol. 25, pp. 3775–3794, 2021.
[29] C. Wu, X. Fu, J. Pei, Z. Dong, "A Novel Sparrow Search Algorithm for the Traveling Salesman Problem," in *IEEE Access*, vol. 9, pp. 153456-153471, 2021.
[30] Z. Zhang, Y. Han, "Discrete sparrow search algorithm for symmetric traveling salesman problem," *Applied Soft Computing*, vol. 118, pp. 108469, 2022.
[31] Y. Huang, X. N. Shen, X. You, "A discrete shuffled frog-leaping algorithm based on heuristic information for traveling salesman problem," *Applied Soft Computing*, vol. 102, pp. 107085, 2021.
[32] Z. Zhang, J. Yang, "A discrete cuckoo search algorithm for traveling salesman problem and its application in cutting path optimization," *Computers & Industrial Engineering*, vol. 169, pp. 108157, 2022.
[33] T. Zhang, Y. Zhou, G. Zhou, W. Deng, Q. Luo, "Discrete Mayfly Algorithm for spherical asymmetric traveling salesman problem," *Expert Systems with Applications*, vol. 221, pp. 119765, 2023.
[34] Y. Chunhua, T. Xiaolin, Z. Xiaojun, G. Weihua, "State transition algorithm for traveling salesman problem," in: Proceedings of the 31st Chinese Control Conference IEEE, 2012, pp. 2481–2485.
[35] X. Zhou, D. Y. Gao, C. Yang, W. Gui, "Discrete state transition algorithm for unconstrained integer optimization problems," *Neurocomputing,* vol. 173, pp. 864–874, 2016.
[36] M. Gündüz, M. S. Kiran, E. Özceylan, "A hierarchic approach based on swarm intelligence to solve the traveling salesman problem," *Turkish Journal of Electrical Engineering and Computer Sciences,* vol. 23, no. 1, pp. 103- 117, 2015.

[37] A. Hatamlou, "Solving travelling salesman problem using black hole algorithm," *Soft Computing*, vol. 22, pp. 8167–8175, 2018.

[38] G. A. Croes, "A method for solving traveling-salesman problems", *Operations research*, vol. 6, no. 6, pp. 791–812, 1958.

[39] G. Reinelt, "TSPLIB—A traveling salesman problem library", *ORSA Journal on Computing*, vol. 3, no. 4, pp. 267–384, 1991.