Received 10 September 2023, accepted 23 September 2023, date of publication 29 September 2023, date of current version 4 October 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3320562

RESEARCH ARTICLE

D-PFA: A Discrete Metaheuristic Method for Solving Traveling Salesman Problem Using Pathfinder Algorithm

PORIA PIROZMAND[1], ALI ASGHAR RAHMANI HOSSEINABADI[2], MAEDEH JABBARI CHARI[3], FAEZEH PAHLAVAN[4], SEYEDSAEID MIRKAMALI[5], GERHARD-WILHELM WEBER[6,7], SUMMERA NOSHEEN[8], AND AJITH ABRAHAM[9,10], (Senior Member, IEEE)
[1]Faculty of Higher Education, Holmes Institute, Sydney, NSW 2000, Australia
[2]Department of Computer Science, University of Regina, Regina, SK S4S 0A2, Canada
[3]Department of Industrial Engineering, Islamic Azad University of Masjed Soleiman, Masjed Soleyman, Khuzestan 6491796581, Iran
[4]Department of Computer Engineering, University of Mazandaran, Babolsar, Mazandaran 4741613534, Iran
[5]Department of Computer Engineering and IT, Payame Noor University, Tehran 19395-4697, Iran
[6]Faculty of Engineering Management, Poznań University of Technology, 60-965 Poznań, Poland
[7]IAM (UME), Middle East Technical University (METU), 06800 Ankara, Turkey
[8]Faculty of Engineering, School of Electrical and Information Engineering, The University of Sydney, Sydney, NSW 2008, Australia
[9]School of Computer Science Engineering and Technology, Bennett University, Greater Noida, Uttar Pradesh 201310, India
[10]Innopolis University, Innopolis, 420500 Republic of Tatarstan, Russia

Corresponding author: Ajith Abraham (ajith.abraham@ieee.org)

This work was supported by the Analytical Center for the Government of the Russian Federation, in November 2021, under Grant 70-2021-00143 and Grant IGK 000000D730321P5Q0002.

ABSTRACT The Traveling Salesman Problem (TSP) which is a theoretical computer science and operations research problem, has several applications even in its purest formulation, such as the manufacture of microchips, planning, and logistics. There are many methods proposed in the literature to solve TSP with gains and losses. We propose a discrete metaheuristic method called D-PFA to solve this problem more efficiently. Initially, the Pathfinder Algorithm (PFA) was presented to handle issues involving continuous optimization, where it worked effectively. In recent years, there have been various published variants of PFA, and it has been frequently employed to address engineering challenges. In this study, the original PFA algorithm is broken into four sub-algorithms and every sub-algorithm is discretized and coupled to form a new algorithm. The proposed algorithm has a high degree of flexibility, a quick response time, strong exploration and exploitation. To validate the significant advantages of the proposed D-PFA, 34 different instances with different sizes are used in simulation results. The proposed method was also compared with 12 State-of-the-Art algorithms. Results indicate that the suggested approach is more competitive and resilient in solving TSP than other algorithms in different aspects. A conclusion and an outlook on future studies and applications are given at the end of the paper.

INDEX TERMS Symmetric TSP, optimization, operational research, discrete pathfinder algorithm, population-based metaheuristic.

I. INTRODUCTION

In several disciplines, combinatorial optimization issues emerge, including Artificial Intelligence (AI), optimization, and many others [1]. Also, many real-world routine problems, such as the Graph Coloring Problem (GCP) [2], Job-Shop Scheduling Problem (JSSP) [3], and Water Pump Switching Problem (WPSP) [4] can be expressed as issues requiring combinatorial optimization, the optimal solution of which can provide us with numerous economic benefits [5], [6]. One of the most common problems that extensively uses discrete hybrid optimization problems is the Traveling Salesman Problem (TSP) [7], in which a salesperson must visit many places and determine the shortest route in which each city is

The associate editor coordinating the review of this manuscript and approving it for publication was Genoveffa Tortora.

© 2023 The Authors. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

visited exactly once. This problem is of the NP-hard [8] kind, and as the number of cities rises, the algorithm′s execution time grows exponentially, making it very difficult to find an exact solution. This subject is thus a prominent topic in Operations Research (OR) studies, Management Science, and Computer Science research [9], [10].

There are two broad types of TSP solutions [7]: The heuristic algorithms and the exact algorithms. The exact algorithms, such as Lagrangian dual [11], branch-and-bound [12], [13], cutting planes [14], and dynamic programming [15], [16], are ideal for solving small to medium-sized issues and may attain the theoretical optimum by mathematical derivatives. However, as the number of cities rises, so does the amount of time required to solve the problem using conventional approaches, and as a result, optimum solutions cannot be obtained [17] therefore, heuristic methods should be used instead [7].

Heuristic algorithms [18], instead of precise approaches, may provide optimum or near-optimal solutions within a fair time and offer the ensuing advantage of straightforward implementation with higher portability, flexibility, and stability [19]. There are several conventional heuristic solutions for TSP, including Genetic Algorithm (GA) [20], [21] Ant Colony Optimization (ACO) [22], [23], Particle Swarm Optimization (PSO) [24], and Simulated Annealing (SA) [25].

Simultaneously, numerous novel bionic algorithms, such as the Bat Algorithm (BA) [26], [27] were introduced to handle optimization problems with great potential for TSP resolution, the Artificial Bee Colony (ABC) [28], the Gray Wolf Optimization (GWO) [29], the Symbiotic Organisms Search (SOS) [30], etc. Although the aforementioned approaches may increase the quality of solutions and have a high global search capacity, they are susceptible to being trapped in local optimums and have a lengthy convergence time. For this reason, hybrid algorithms have been proposed to improve the algorithm′s ability to jump from a local maximum, such as some versions of BA [31], [32], [33], Simulated Annealing Based Symbiotic Organisms Search (SA-OS) [34], Growing Self-Organizing Array (GSOA) [35], Selection Schemes [36], Hybrid Discrete Artificial Bee Colony (HDABC) [37], Hybrid algorithm based on PSO and GA (PSOGA) [38] and so forth [7]. To simulate issues in the actual world, different versions of TSP, such as Asymmetric Traveling Salesman Problem (ATSP), Multiple Traveling Salesman Problem (MTSP), Colored Traveling Salesman Problem (CTSP), etc. have been designed [39], [40], [41], [42].

This paper uses a metaheuristic algorithm called Pathfinder Algorithm (PFA) to solve the TSP problem. This algorithm is inspired by the nature and movement of herds of animals to find food or prey, which was presented in 2019 by [43]. In this algorithm, others in the group follow the leader, all moving randomly. Like other optimization algorithms, PFA was initially used for continuous optimization problems. It has been selected as an optimal global solver as it achieves a globally optimal solution in a shorter time and does not get stuck in the local optimum. Besides, it is capable of addressing problems with single and multiple objectives.

This study aims to show off the effectiveness of the PFA in solving discrete problems. TSP is a complex and NP-hard problem [8], and using the discrete PFA optimization method, it is possible to characterize the global optimum to address challenging discrete optimization issues successfully. This work introduces a unique discrete PFA method called D-PFA for solving the complicated Symmetric Traveling Salesman Problem (STSP). In this study, the original PFA algorithm is broken into four sub-algorithms, coupled with discretized sub-algorithms to form a new technique. To validate the performance of the proposed D-PFA, different experiments are conducted on 34 TSP benchmark instances, and different tests are utilized to compare the proposed D-PFA with other cutting-edge algorithms such as the Improved Bat Algorithm (IBA) [31], Discrete Imperialist Competitive Algorithm (DICA) [31], GA [31], Evolutionary Simulation Annealing (ESA) [31], Discrete Grey Wolf Optimizer (D-GWO) [1], Island-Based Distributed Genetic Algorithm (IDGA) [31], Discrete Firefly Algorithm (DFA) [31], Discrete Sparrow Search Algorithm (DSSA) [7], Discrete Symbiotic Organisms Search (DSOS) [19], Combinatorial Bees Algorithm with Nearest Neighbor Method (CBA-NNM) [44], Combinatorial Artificial Bee Colony (CABC) [45], Differential Evolution Algorithm (DE) and Partial Swarm Optimization (PSO), namely (D-DEPSO) [46] and a more Advanced Version of ACO [47] called Heterogeneous Adaptive Ant Colony Optimization (HAACO).

The paper′s organization is as follows. Some work to address the TSP issue by the researchers is represented in Section II. The problem statement and a brief description are given in Section III. PFA algorithm and its differences from other algorithms are investigated in Section V. The proposed algorithm and simulation results are provided in Sections IV and VI, respectively. Finally, Section VII contains a conclusion and an outlook for future studies.

II. LITERATURE REVIEW

As mentioned in the previous section, the TSP engaged in substantial, in-depth academic and professional study. The first is due to its inherent practical base, which includes issues with vehicle routing, circuit board printing, *X-ray* crystallography, etc.; the second is because of its complexity. A solution to solving NP-hard problems is to make use of metaheuristic strategies. Here are a few of the many metaheuristic strategies for solving the TSP that can be found in the literature:

Zhang et al. [48] solved the *m-Steiner* Traveling Salesman Problem with *Online* Edge Blockages, intending to obtain m closed networks that would meet every client at least once, resulting in the lowest possible m salesmen cost, and then provided an online algorithm to solve the problem.

To solve the TSP problem, in [49] proposed an updated Cuckoo Search (CS) version using cluster analysis and random walks. To maintain the superior solutions obtained by the algorithm and the population diversity, the random preference walk and Levy flights and of the original CS algorithm were swapped out for them utilizing new tools like local discrete random walk and adjustment operator. For large-scale TSP problems, they first divided cities into k categories using the *k-means* method and then combined them using random techniques. They also used the 2-opt operator as a local optimizer to increase the algorithm′s degree of convergence. Finally, they compared the stability and accuracy of their algorithm with other well-known ones.

Kanna et al. [50] combined two metaheuristic algorithms named Deer Hunting Optimization Algorithm (DHOA) and Earthworm Optimization Algorithm (EWA) with a method for hybrid optimization called EW-DHOA to optimally solve the TSP problem to minimize the distance traveled by the salesman considering all the cities. They were able to reduce the computational complexity of the TSP problem and achieve optimal results.

Al-Gaphari et al. [51] used basic operators, dissimilar solutions, and modular arithmetic techniques and provided three algorithms inspired by the discrete crow to solve Discrete Traveling Salesman Problems (DTSPs). The three techniques used ensure switching from continuous to discrete spaces without loss of information. The evaluation criteria of the algorithm are the average accuracy of optimal solutions, average errors, and average computational time.

Bidirectional Graph Neural Network was proposed by [52] to tackle the arbitrary STSP, in which the network learns through imitation to construct the next city it will encounter. The most essential element of proposed method was the bidirectional message carrying layer. It can encrypt graphs with edges and partial solutions. Consequently, the proposed technique may provide near-optimal TSP solutions in any symmetric network. The authors proposed merging their algorithm with informed search in future work to enhance its performance. To improve the performance of their algorithm, the authors suggested combining it with informed search in future work.

Huerta et al. [53] suggested a novel metaheuristic strategy for solving the Euclidean TSP using a collection of five well-known solvers. The authors devised a unique matrix grid for node representation in a spatial format to circumvent the expensive procedure of calculating features. Then, at each level, they suggested a brand-new compact staggered representation for ranking algorithms. The suggested technique greatly decreased the time necessary to forecast the optimal solution and achieved a prediction accuracy of 79.8%.

Gunduz and Aslan [54] proposed a discrete version of the Jaya algorithm called DJAYA to solve discrete optimization problems. They improved the Jaya algorithm, using random permutations to generate initial solutions and nearest neighborhoods. The population generation strategy of the original Jaya algorithm was updated to address discrete optimization issues. In the discrete version, eight transformation operators were used based on the characteristics of the discrete optimization issue. Using the suggested approach, they solved 14 instances of the renowned discrete problem, the STSP. Lastly, to enhance the Best Solution (BS), the 2-opt heuristic method was applied to the BS generated by DJAYA.

Wang and Han [55] combined Symbiotic Organisms Search (SOS) with the ACO to create the SOS-ACO method for solving the TSP. In the proposed technique, after assigning ACO parameters, the remaining parameters are optimized by SOS, and then, using these optimum parameters, ACO finds the best or nearly BS. Consequently, assigning ACO parameters becomes simpler. In addition, the SOS-ACO algorithm employs a basic local optimization method to increase convergence rate and solution quality.

Swarm Intelligence Algorithm (SIA) to solve the TSP issue using the DSSA and the global perturbation approach proposed by [7]. In this technique, the population′s initial solution is produced using roulette-wheel selection, and an order-based decoding mechanism is created to update the sparrow′s location. Exploration and exploitation activities have been balanced using a global perturbation method with Gaussian mutation and a swap operator. The strategies used have increased solution quality and convergence rate.

By including symmetry, transformation, shift, and swap operators into the standard GWO algorithm, [39] developed the Transformation Operator Based Gray Wolf Optimizer (TO-GWO) to address the TSP issue. Each wolf in this algorithm represents a solution to the TSP issue, and the wolves interact with the leader wolves using the swap, shift, and symmetry operators to find the BS to the problem.

Using three algorithms, a) Node clustering based approach in which nodes are categorized in a set of clusters, b) Dynamically regulated information entropy-based adaptive pheromone evaporation, and c) Variety of solutions based termination algorithm, [56] proposed a unique ACO algorithm to solve the TSP, which may enhance overall performance, decrease runtime, and address the drawbacks of ACO-based approaches, such as becoming trapped in local optimum and the difficulties in managing the parameters for various samples.

Dong and Cai [57] defined the Colored Balanced Traveling Salesman Problem (CBTSP), for modeling partial workspace overlapping optimization problems, such as the planning and distribution of resources and supplies, and proposed a new GA, ITO process-based Novel Genetic Algorithm (NGA), which can also be applied to large-scale CBTSPs.

Silva et al. [58] developed a mathematical formulation for the new version of the Quota Traveling Salesman Problem (QTSP), which includes Passengers, Incomplete Ride, and Collection Time (QTSP-PIC) with constraints including travel time, vehicle capacity, passenger limitations, and rides penalties. In this formulation, to cut down on travel expenses, the salesman uses a flexible ridesharing system. To solve this problem, the authors proposed naive heuristics and three ant algorithms called Ant System (AS), Ant Colony System (ACS), and Multi-Strategy Ant Colony System (MS-ACS).

To overcome the disadvantages of the ACO method in solving the TSP [59], proposed a self-adaptive ACO called (DEACO) along with ways to enhance the algorithm′s unpredictable convergence time and arbitrary decision-making that dynamically changes the ACO parameters. This method′s main idea is choosing the first city (starting point) that leads to the shortest route, and DEACO finds the shortest or the cheapest route for each cluster. Although this method improved the quality of solutions in each cycle and uses the self-adaptive mechanism of the ACO method to select the best ant for updating pheromone trails, it requires more experimental basis than theoretical research.

Combining optimization algorithms such as Rider Optimization Algorithm (ROA) and Spotted Hyena Optimizer Algorithm (SHO) [60], developed Spotted Hyena-based Rider Optimization (S-ROA) to solve the TSP problem optimally and effectively, whose objective function is defined as minimizing the distance traveled by the salesman from all cities.

Deckerová et al. [61] created the GSOA, a non-supervised learning-based heuristic solution for solving the non-Euclidean form of the TSP Problem considering neighborhoods on a sphere. This method was able to find an initial solution as good as the sampling-based method. In addition, the authors presented a rapid post-processing optimization technique for enhancing original results.

Pandiri and Singh [62] proposed an ACO with a variable perturbation degree to solve the Generalized Covering Traveling Salesman Problem (GCTSP), in which the perturbation degree of a solution to generate its neighbor solution decreases during iterations. The goal of this strategy was to cover the salesman if they are inside the coverage radius, r_i a facility visited by the salesman and lower him/her overall travelling distance at the same time.

In [1], a novel discrete version of GWO called D-GWO is used to solve TSP. To improve the answers obtained by the GWO algorithm, the authors implemented the 2-opt algorithm on the received responses. To show the efficiency of D-GWO algorithm, they tested the proposed algorithm on 17 TSP instances and compared the results of the proposed algorithm with other algorithms. In their experiments, they showed that the D-GWO algorithm reached a better solution than the compared algorithms in some instances and had a better performance. But the D-DWO algorithm could not work well in large-scale instances, and with the increase in the complexity of the problem, the execution time of the algorithm also increases.

Zhang and Han [7] solved TSP using a combination of DSSA with global perturbation. The proposed algorithm initially uses a roulette-wheel mechanism to generate to improve the solutions. Next, a sequence-based encoding and decoding strategy are introduced to complete the sparrow position update. Finally, long and short steps combined with global perturbation mechanism accelerates the algorithm′s convergence and improves the algorithm′s ability to jump from the local optimum. To demonstrate the ability of their algorithm to solve the above problem, the authors implemented the proposed algorithm on 34 TSP instances. They compared the results obtained from the proposed algorithm with various classic, heuristic, and metaheuristic algorithms. They showed that their proposed algorithm has better convergence and robustness than Classic algorithms. Compared to heuristic and metaheuristic algorithms, it has a relative superiority in solving large-scale instances. However, it could not achieve the BKS solution in several cases.

Osaba et al. [31] solved symmetric and asymmetric TSP. Also, they were able to improve the BA. In the Improved Bat Algorithm, which the authors named (IBA), bats have a kind of ''intelligence″ that makes bats follow different movement patterns depending on where they are in space. To demonstrate the effectiveness of the IBA algorithm, the authors compared the proposed algorithm in 37 instances of TSP with the basic BA algorithm and other algorithms. They showed that the IBA algorithm can solve the symmetric and asymmetric TSP problem and can achieve the BKS solution in most instances.

Sahin [44] also used the combinatorial BA to solve the TSP issue. To create the initial population, the nearest neighbor method was used. In the second phase, the Multi-Insert function was introduced to the local search section instead of the Swap function. To show the effectiveness of the suggested method, the author used 24 different TSP instances and compared the results obtained from the proposed algorithm with three other algorithms. The algorithm presented in solving large-scale instances has not been able to solve the problem well. With the increase in the number of cities, the execution time of the algorithm has also increased.

A combinatorial version of standard ABC (CABC) and an improved version of CABC algorithm (ICABC) are used in [45] to solve the TSP. Experimental studies of two ABC algorithms were performed on 15 instances of TSP and then compared with eight different types of heuristic and metaheuristic algorithms. The simulation results of presented algorithms showed that they could perform well in solving the problem and that the CABC algorithm′s convergence performance could be improved by using the ABC idea for TSP.

Ezugwu and Adewumi [19] provided an almost optimal solution for TSP using the DSOS algorithm. To present the answer, the suggested method used three different local search operators based on mutation to reconstruct the population, improve exploration and exploitation capabilities, and faster convergence speed. To show the effectiveness of the presented algorithm, the authors implemented the proposed algorithm on 23 instances of TSP and compared its simulation results with five different algorithms. The DSOS algorithm achieved relative superiority over the compared algorithms and obtained results close to BKS. Still, compared to one of the algorithms named PSO-ACO-3-opt [47], it could not achieve the shortest distance for TSP.

Tuani et al. [63] described an adaptive technique for a heterogeneous ant colony population that develops alpha and beta ACO controller parameters to obtain near-optimal solutions. In their proposed algorithm, they presented a set of rules to adapt the parameters to approach the parameter values to the optimal values, which will improve the search process and the algorithm′s operation. Also, to improve the obtained solutions, they used the 3-opt local search method, which improved the obtained solutions. In order to show the effectiveness of the proposed algorithm and its ability to solve the problem, the authors implemented the algorithm on 10 instances of TSP and compared its results with four other algorithms. The simulation results of the proposed algorithm showed that the algorithm achieved superiority over the compared algorithms in seven instances. In three instances, it reached the BKS solution.

III. THE TRAVELLING SALESMAN PROBLEM

As stated, the TSP is a combinatorial discrete optimization problem with significant applicability in many areas. The goal is to get a Hamilton tour of cities that a salesman should meet. To model TSP, we can use a complete non-directional graph, $G = (N, E)$, where $N = \{1, 2, 3, \ldots, n$, represents the set of cities and E represents the set of edges. Each edge $i, j \in E$ has a non-negative cost d_{ij}. In fact, an edge represents a route between two cities and the distance between the two cities is represented by the length of each edge. The salesman starts from one city and meets n cities and finally returns to the same first city and meets each city exactly once [7].

The Euclidean distance is the standard method for calculating the cost between two cities. The Euclidean distance d between cities c_1 and c_2 is calculated as follows:

$$d = \sqrt{(c_{1x} - c_{2x})^2 + (c_{1y} - c_{2y})^2}. \tag{1}$$

Here, (c_{1x}, c_{1y}) and (c_{2x}, c_{2y}) are the coordinates of the cities c_1 and c_2 respectively.

As mentioned earlier, this study focuses on the symmetric TSP [7] in which travel between two cities from opposite directions costs the same, i.e., $d_{ij} = d_{ji}$ distance as presented in the following Equations (2)-(6):

Minimize :

$$Z = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij} \tag{2}$$

Subject to

$$\sum_{j=1}^{n} x_{ij} = 1, i \in \{1, 2, 3 \ldots n\}, \tag{3}$$

$$\sum_{i=1}^{n} x_{ij} = 1, j \in \{1, 2, 3 \ldots n\}, \tag{4}$$

$$\sum_{i,j \in S} x_{ij} \leq |S| - 1, 2 \leq |S| \leq n - 2, S \subset \{1, 2, 3 \ldots n\}, \tag{5}$$

$$x_{ij} \in \{0, 1\}, \forall i, j \in \{1, 2, 3 \ldots n\} \text{ and} i \neq j. \tag{6}$$

Minimizing the salesman′s round-trip journey time is the objective function defined by Equation (2). If the salesman has gone directly from city i to city j : $x_{ij} = 1$ and otherwise $x_{ij} = 0$. Equations (3) and (4) ensure that each city is met only once. The constraint given in Equation (5) prevents sub-touring and ensures that the final route has only a complete closed-loop route, which is a complement to Equations (3), (4), and (6) that define binary decision variables [7].

IV. THE PATHFINDER ALGORITHM

Metaheuristic methods can be considered in three classes: based on evolutionary methods, based on physics, and based on intelligent particles.

The PFA is inspired by search behavior in the hunting or feeding area led by an individual in animal herds [43]. Unlike other algorithms, this algorithm has a leader, and the other participants follow him. But not all particles move regularly; they all move randomly. This algorithm aims to find the router for the best area for food and hunting, and we can call hunting the optimal global. Each time (t) is repeated, the members consciously move towards the router (the optimal solution chosen is global).

In the following, we will explain the difference between the PFA and some examples of other algorithms:

•*Difference between PSO and PFA:* In PSO, time and speed are important (in movement), and in PFA, speed is not important.

•*The difference between ACO and PFA:* In ACO, there is no hierarchical structure, and it is a collective movement. In PFA, the move is towards the leader and interacting with the neighbors.

•*The difference between ABC and PFA:* ABC has global and local optimization, but all members are equal in PFA and have the same chance.

•*Difference between Firefly Algorithm (FA) and PFA:* FA utilizes the luminosity of fireflies. In the distorted motion of a firefly swarm, every individual moves toward the brightest light. Additionally, the whole population may be subdivided. But in the PFA, the population is not separated into subgroups, and members follow the best-fitting individual. Consequently, the distance between neighbors may be decreased in each cycle, bringing each individual closer together.

We chose the PFA algorithm to solve symmetric TSP based on the differences between it and the other algorithms mentioned above because:

1. The algorithm can solve single/multi-objective problems; the symmetric TSP is a single-objective problem.

2. The algorithm has a high degree of flexibility, and the problem can be easily modeled with it.

3. The response time is faster than other algorithms. This can be very useful in solving our problem because the symmetric TSP space is large and the time to reach the optimal solution of other algorithms is high.

4. It has stronger exploration and exploitation, and we can increase its search space.

5. Unlike other algorithms, it is not stuck in the local optimum.

6. It can provide more accurate solutions and answers.

A. INSPIRATION

All behaviors are performed in a crowd based on a common action of all people. In addition, one person leads the crowd (group) and manages many actions.

The group leader leads his team members to the right conditions. Team members seek help from a leader or member who can make decisions. Of course, the leader is not a permanent leader, and anyone with a higher ability and can provide more comprehensive information will be selected as a leader.

A few experienced individuals can guide the whole herd in the PFA algorithm. In this algorithm, an update is provided in terms of force alignment and their attraction/repulsion in 2D space. The interplay between the members of a swarm and leader is modeled mathematically in the PFA algorithm [43].

B. MATHEMATICAL MODEL OF THE BASIC PFA

In the PFA, every member is assigned a spot in the following 2D, 3D, or d spaces. If a group member is in a promising area at any other time, they will be elected as the leader. Besides, it is considered that all potential solutions to a problem are vectors representing every individual′s position; therefore, individuals in the herd can move in 2D, 3D, and d-dimensional space. It should be noted that in the above algorithm, the group leader is considered a router [43].

The initialization will be done randomly according to the herd′s position in the herd.

To find the goal and follow the leader, we use the following Equation (7).

Fitness Function Calculation: A member with the best position is chosen as the leader.

$$x\left(t+\Delta t\right)=X^{0}\left(t\right)\cdot n+f_i+f_p+\varepsilon, \tag{7}$$

where

t : time period,

x : Position vector,

n : Unit vector without direction,

f_i : Two-way interaction with x_i neighbors and x_j neighbors,

f_p : Calculate the interaction between the leader and the members,

ε : Vibration vector.

$$x_i^{k+1}=x_i^k+R_1\cdot\left(x_j^k-x_i^k\right)+R_2\cdot\left(x_p^k-x_i^k\right)+\varepsilon, i\geq 2. \tag{8}$$

The position of the router is updated and calculated based on the following Equation (8):

$$x_p\left(t+\Delta t\right)=x_p\left(t\right)+\Delta x+A, \tag{9}$$

where

x_p : Router position vector,

Δx : Distance traveled by the router to move from one point to another,

Time traveled by the router to move from one point to another, Δt :

A : Fluctuation rate vector.

The authors further conclude that the aggregate group movement model mentioned in Equations (7) and (8) cannot be used directly to solve optimization problems. Therefore, they modified Equations (7) and (8) in Equations (9) and (10):

In Equation (9), k is the value for the most recent iteration, x_i is the position vector for the i^{th} member, x_j indicates the position vector of the j^{th} member of the set, and R_1 and R_2 are vectors with random variables, where R_1 is equivalent to αr_1 and R_2 equivalent to βr_2, where r_1 and random variables are uniformly created in the domain [0, 1], α is the interaction coefficient that describes the amplitude of each member′s motion relative to its neighbor, and β determine the random distance to keep the herd with the leader, and r_1 and r_2 determine random movements. Two important states will occur for when $\alpha\rightarrow 0, \beta\rightarrow 0, \alpha\rightarrow\infty$ and $\beta\rightarrow\infty$.

In this algorithm, initially, people will move randomly in the search area without any connection to each other. That is, people can take the path that the leader is taking and will not take another path.

In the above algorithm, when $\alpha < 1$ and $\beta < 1$, it is difficult to change the direction of people and get closer to the leader. Also, if $\alpha > 1$ and $\beta > 1$, the distance between people and the group leader will be increased too much, and this will take away from the optimization.

Therefore, in both cases, the algorithm will not find suitable solutions. In this algorithm, α and β are randomly selected in the range $[1, 2]$. A good state is a state where α and β are about one:

$$x_p^{k+1}=x_p^k+2r_3\cdot\left(x_p^k-x_p^{k-1}\right)+A. \tag{10}$$

Here, r_3, a vector with random numbers generated uniformly in $[0, 1]$ domain, each cycle generates A using Equation (12). In Equation (11), ε represents the vibration in each iteration of the algorithm:

$$\varepsilon=\left(1-\frac{K}{K_{max}}\right)\cdot u_1\cdot D_{ij}, D_{ij}=\left\|x_i-x_j\right\|, \tag{11}$$

$$A=u_2\cdot e^{\frac{-2k}{k_{max}}}. \tag{12}$$

Here, u_1 and u_2 are vectors with random numbers ranging between $[-1, 1]$, D_{ij} is the distance between two members while $K_{\max}$ indicates the most iterations possible.

If in Equations (8) and (9) the second condition and also in Equation (10) only the third condition is equal to zero, the two variables A and ε can create random movements for all members.

Therefore, A and ε must be within the appropriate range values to ensure multidirectional and random movement using equations 11 and 12. There must also be a random number generator to provide random movement for these conditions. Besides, they make rapid changes to the initial iterations and then reduce those changes to subsequent iterations, thus facilitating the search in the exploration and productivity stages.

The variables in the range $[-1, 1]$ are set by u_1 and u_2, and members can move to their previous paths. According to the above conditions, if it is $u_1 < -1$ and $u_2 < -1$ or $u_1 > +1$ and $u_2 > +1$, members can change their paths with long steps and can change their positions. Therefore, members can find solutions they may explore in new steps [39], [64].

If ($\alpha < 1, \beta < 1$), then: The leader does not change and the members do not change direction

If ($\alpha > 1, \beta > 1$) then: Members can change.

The router looks for the optimum hunting/eating spot in the PFA algorithm. The best hunting/eating area can be considered optimal overall.

At each stage, the leader is positioned as the best location in the current stage, and the other members approach it. We assume the BS ever seen is the global optimal and is used as a range of food by all members.

The PFA algorithm randomly selects and initializes members, and then the initial population is generated. After that, each member′s fitness is calculated, and the member with the highest fitness is recognized as the leader, and the rest of the members will move towards him. In the next steps, the leader in the problem search space is generated and updated simultaneously by two Equations (10)-(12) and the vibration rate vector A, each time the algorithm is repeated.

According to the discussed Equations and calculating the fluctuation rate and the number of iterations, the algorithm ends whenever the maximum number of iterations is reached [43].

As a result, to understand the PFA and how it will solve the optimization problems, the following can be stated:

1. The vector is directly proportional to the amount of fitness function and is the best router. In other words, the router is chosen based on the vector with the highest fitness. Additionally, the PFA selects the ideal location for the router, guaranteeing that the fitness gained from each repeat is always preserved.
2. The entire population′s position is updated according to close neighbors and routers, and the leader can change.
3. Each person′s position is updated in accordance with the router and the other members. Therefore, they move towards the router and get closer to their neighbor.
4. All members moving at random causes local optimization, but changing the vibration vector ε and approaching the router can prevent this situation.
5. The convergence of A and ε to zero first aids exploration and then exploitation.

Based on the above and the description of the PFA algorithm, it can be concluded that this algorithm can solve optimization problems well and is also able to avoid getting stuck in the local optimization. Therefore, it can be concluded that the PFA algorithm can solve single/multi-objective problems. Fig. 1 depicts a comprehensive flowchart of the PFA algorithm and its phases.

V. THE PROPOSED DISCRETE PATH FINDER ALGORITHM

This method uses the PFA to solve the TSP, which is a metaheuristic method that inspires how a group of animals moves to find food or prey. The purpose of this algorithm is to find the shortest route between cities by the traveling salesman and be able to solve large-scale problems in a short time.

A. INITIAL POPULATION (MEMBERS) GENERATION

In the proposed D-PFA, each member is a one-dimensional array with N cells, each cell is randomly filled by one of the problem cities without repetition, and the first and last cell of each member always contains a value of one because, in the TSP problem, there are several cities or several thousand cities in an environment where the seller has to travel the distance between all these cities so that he should not pass each city twice and get the shortest possible distance between the cities and finally return to the first city.

The following fully explains the process of creating the initial population and the proposed algorithm.

Based on Equation (9), the main equation for moving a member in PFA, Fig. 2 shows how we discretize this equation to solve the TSP problem.

In the following, an example of obtaining the distance between two members is fully described, and we show how d_{ij} and D_{ij} are calculated. Fig. 4 shows Remark 1 of the distance matrix for seven cities, where the distance between all seven cities is displayed.

Remark 1: As shown in Fig. 2, the Equation is divided into four parts. The first one (a) is not moving, which does not need to do anything. But the three other parts should be explained in detail. All the movements *b, c,* and *d* are based on an important notion named *distance.* Distance is specified in Equation (11). This Equation is about continuous problems and has nothing to do with TSP. So, we should first redefine this crucial concept in our problem.

Since there are too many cities in a TSP problem, we have defined distance as an array instead of a single member. It works better sense when we analyze the specifications of TSP. D_{ij} is a distance array between x_i and x_j. Before calculating D_{ij}, d_{ij} should be calculated. Figure 3 shows an example of how we calculate D_{ij}.

In Fig. 3, there are two members named $x_i$and x_j, and their distance is called d_{ij}. As shown in this figure, first we get d_{ij} from the distance matrix and then we consider the maximum distance obtained from d_{ij} as max distance. Next, to get D_{ij}, we divide the distance between both cities from d_{ij}

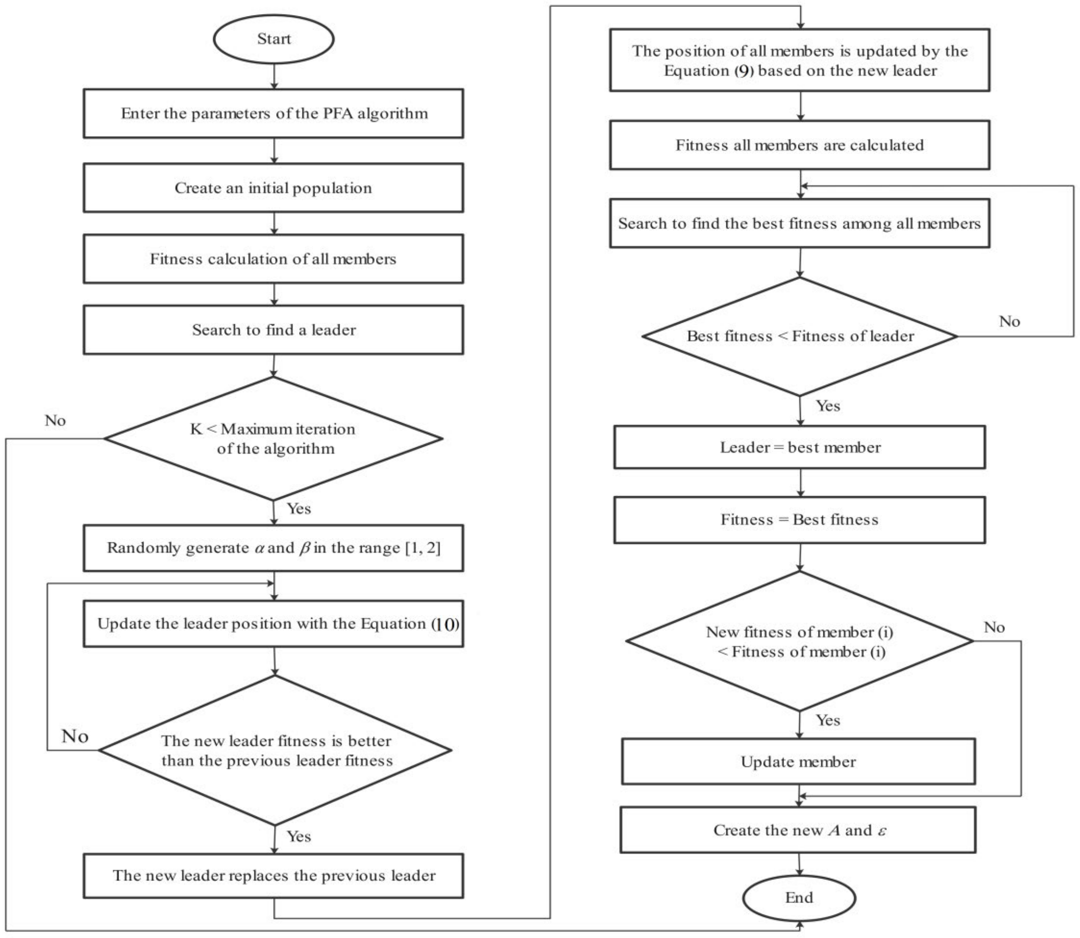

Start
Enter the parameters of the PFA algorithm
Create an initial population
Fitness calculation of all members
Search to find a leader
K < Maximum iteration of the algorithm
No
Yes
Randomly generate α and β in the range [1, 2]
Update the leader position with the Equation (10)
The new leader fitness is better than the previous leader fitness
No
Yes
The new leader replaces the previous leader
The position of all members is updated by the Equation (9) based on the new leader
Fitness all members are calculated
Search to find the best fitness among all members
Best fitness < Fitness of leader
No
Yes
Leader = best member
Fitness = Best fitness
New fitness of member (i) < Fitness of member (i)
No
Yes
Update member
Create the new A and ε
End

FIGURE 1. The flowchart of PFA algorithm.

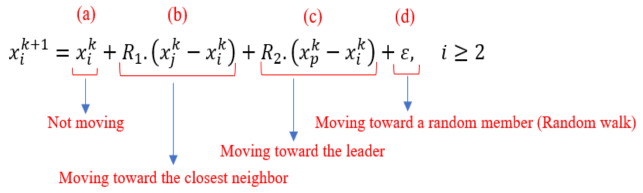

(a)
(b)
(c)
(d)
$x_i^{k+1} = x_i^k + R_1.(x_j^k - x_i^k) + R_2.(x_p^k - x_i^k) + \varepsilon, \quad i \geq 2$
Not moving
Moving toward a random member (Random walk)
Moving toward the leader
Moving toward the closest neighbor

FIGURE 2. Moving a member in PFA.

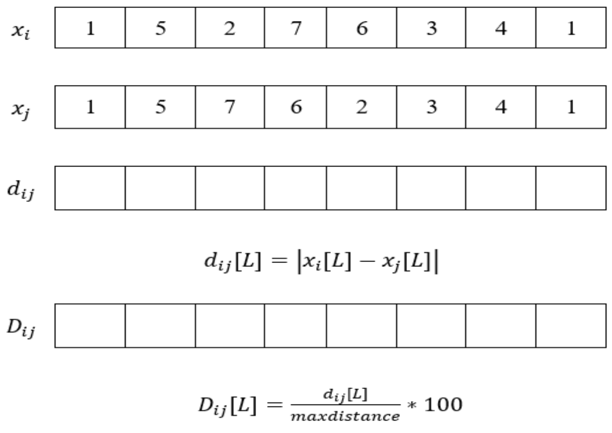

x_i
1
5
2
7
6
3
4
1
x_j
1
5
7
6
2
3
4
1
d_{ij}
$d_{ij}[L] = |x_i[L] - x_j[L]|$
D_{ij}
$D_{ij}[L] = \frac{d_{ij}[L]}{maxdistance} * 100$

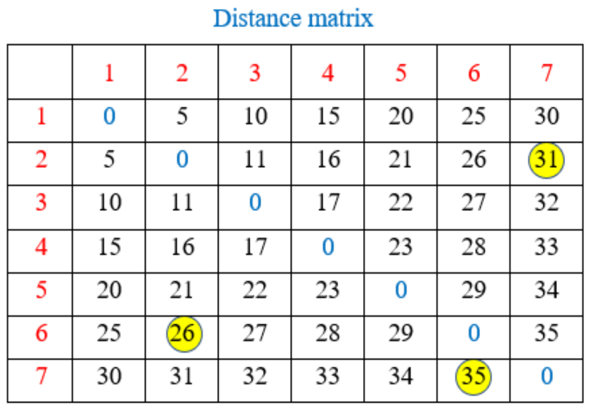
FIGURE 3. Shows how we compute D_{ij}. Here, maxdistance equals the maximum distance between two cities on the map.

by maximum distance and multiply the result by 100. In this way, we will get D_{ij} separately for both cities.

In Fig. 5, two members x_i and x_j were randomly generated for seven cities, the first and last city of which is equal to one. The seller starts his route from city one and returns to the city one after navigating all the cities. Then d_{ij} was calculated between two members x_i and x_j based on the distance matrix, which is **maxdistance=35**.

Now, after computing d_{ij} based on the distance matrix, we need to calculate D_{ij}, the way to calculate it is fully shown in Fig. 6. In this figure, we divide the distance between every pair of cities (d_{ij}) by max distance and then multiply the result by 100.

This array (D_{ij}) shows the probability of changing a member x_i to a new member in another member (x_j). Based on Equation (9), the destination member (x_j) could be j, leader, or random.



Distance matrix

	1	2	3	4	5	6	7
1	0	5	10	15	20	25	30
2	5	0	11	16	21	26	31
3	10	11	0	17	22	27	32
4	15	16	17	0	23	28	33
5	20	21	22	23	0	29	34
6	25	26	27	28	29	0	35
7	30	31	32	33	34	35	0

FIGURE 4. Remark of the distance matrix for seven cities.

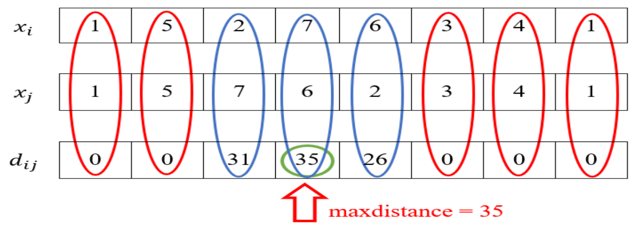
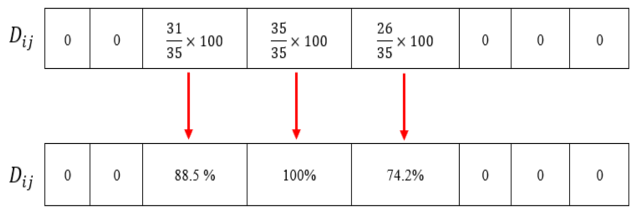


FIGURE 5. Example of calculating d_{ij}.



D_{ij} 0 0 $\frac{31}{35}\times 100$ $\frac{35}{35}\times 100$ $\frac{26}{35}\times 100$ 0 0 0
D_{ij} 0 0 88.5 % 100% 74.2% 0 0 0

FIGURE 6. Example of calculating D_{ij} based on new d_{ij}.

The coefficients of this Equation for the *final percentage array.*

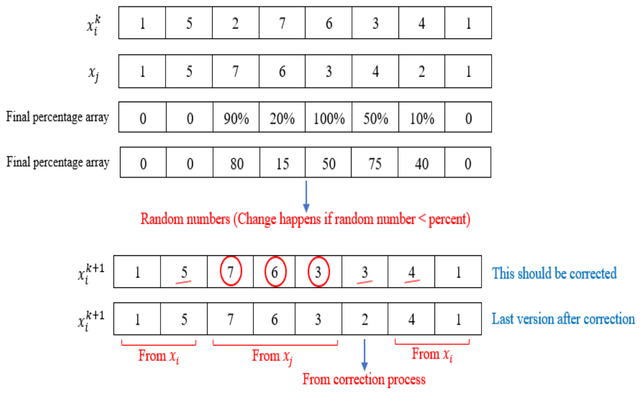
In this Equation:

- j specifies x_j which is the closest member to i based on $\min(\sum d_{ij})$ for all members like x_j,
- Leader is the current best member, also known as the leader.
- Random member (ε) is a member which is selected by chance from the population. This member is used to calculate "ε".
- Here, "ε" implements a seamless move from the exploration to exploitation phase for every member.

B. UPDATE MEMBERS

This section shows how each member is updated and moves towards the leader. Fig. 7 shows a sample of update members acquired.

The new x_i (x_i^{k+1}) is much closer to x_j. Given the fact that x_i is supposed to move toward x_j, the result of this update



x_i^k 1 5 2 7 6 3 4 1
x_j 1 5 7 6 3 4 2 1
Final percentage array 0 0 90% 20% 100% 50% 10% 0
Final percentage array 0 0 80 15 50 75 40 0
Random numbers (Change happens if random number < percent)
x_i^{k+1} 1 5 7 6 3 3 4 1
This should be corrected
x_i^{k+1} 1 5 7 6 3 2 4 1
Last version after correction
From x_i
From x_j
From x_i
From correction process

FIGURE 7. Updating a sample of members.

is sensible. Algorithm 1 shows the pseudocode for updating members.

Algorithm 1 Update Member
```
1:    $R_1$ = random number in range [1, 0]
2:    $R_2$ = random number in range [1, 0]
3:      for i=1 to memberSize – 2 do
4:          calculate movement with Eq. (9)
5:          shift all cells of leader
6:      end for
```

C. LEADER SELECTION

After generating the initial population, the member with the lowest fitness is selected as the leader. If we consider a group of animals, the member that is stronger than the others are selected as the leader, and then all the members go toward him, so, in this algorithm, members move toward the leader's direction. The pseudo-code of Algorithm 2 for the leader selection follows:

Algorithm 2 Leader Selection
```
1:    Ascending sort initial population by fitness
2:        Leader = first member of population
3:        lastLeader = Leader
```

D. UPDATE LEADER

The movement of the leader is based on Equation (10). After the update, the leader's fitness is recalculated. If it is decreased after the update, the leader is appointed as the previous leader, and the changed member is selected as the leader. Otherwise, members will move to the last leader that was selected.

The notation of distance is the same as the previous one. The only difference is A (fluctuation). The distance value A adds something like mutation to the first movement of the leader. At the beginning of the algorithm, it is around one ($A = 100\%$), but gradually it will reach zero ($A = 0$).

It means that we have plenty of random walks in the first rounds, but they are limited in the final rounds. In other words, A facilitates a smooth transition from the exploration to exploitation phase. The Algorithm 3 depicts the pseudocode for leader updating.

Algorithm 3 Update Leader

```
1: for $i = 1$ to $leaderSize$ - $2$ do
2:     calculate movement with Eq. (10)
3:     shift all cells of leader
4: end for
```

All the coefficients of the algorithm, like α, β, u_1, u_2, r_1, r_2, r_3, ..., should be specified based on the problem. In the man PFA, these numbers are set to solve continuous problems. But we need new numbers which can keep the percentage array in a valid range ([0, 100]) .

E. TERMINATION CONDITION

The termination condition of the algorithm is that algorithm is repeated a certain number of times. Each time, we will have a leader with lower fitness. Finally, the algorithm returns the best path and time to reach the solution. Algorithm 4 shows the pseudocode of the proposed method.

Algorithm 4 Pseudocode of Proposed D-PFA

```
1: Initialize population of members $x_i (i = 1, 2, ..., l)$
2: Generate distance matrix
3: Calculate fitness for all members
4: $Leader = best\ member$
5: $LastLeader = Leader$
6: Initialize iteration K=0
7: while $K < maximum\ number\ of\ iterations$ do
8:     $\alpha$ and $\beta$ = random number in [1], [2]
9:     $u_1$ = random number in $[-1, 1]$
10:    $u_2$ = random number in $[-1, 1]$
11:    calculate $\varepsilon$ with Eq. (11)
12:    calculate $A$ with Eq. (12)
13:    update Leader position using Eq. (10)
14:    if $UpdatedLeader$ is better than Leader
15:        $Leader = UpdatedLeader$
16:        $LastLeader = Leader$
17:    end if
18:    for $i = 2$ to number of populations
19:        update members using Eq. (9)
20:    end for
21:    calculate fitness for all members
22:    find $BestMember$
23:    if $BestMember$ is better than Leader
24:        $LastLeader = Leader$
25:        $Leader = BestMember$
26:    end if
27:    for $i = 2$ to number of populations
28:        if new $fitness\ of\ member(i) < fitness\ of\ member(i)$
29:            $member(i) = newMember(i)$
30:        end if
31:    end for
32: end while
33: return the best Leader
```

TABLE 1. Parameter settings of D-PFA for TSP.

Parameter	Value	Parameter	Value
Population size	50	Maximum of iterations	200
α	Random number in [1, 2]	R_1	Random number in [0, 1]
β	Random number in [1, 2]	R_2	Random number in [0, 1]

VI. SIMULATION RESULTS

The proposed D-PFA algorithm′s simulation results are presented in this section. Its comparison with other metaheuristic algorithms is explained in detail. TSPLIB library [65]was used to simulate TSP. To determine the parameters of the algorithm, the previous studies in Section II of this paper, which are explained in detail, have been used as a reference.

In the proposed algorithm, the initial population number is considered as 50, because it seems that a large number of populations will generate more processing load. It provides the cases used by the comparison method in the subsequent section, as well as the number based on their foundation. The experimental parameters utilized for D-PFA are listed in Table 1.

Section A first describes the environment and experimental configuration for the simulation along with the calculation of the Best Solution (**PDB**(%)) percentage deviation of the Average Solution (**PDA** (%)) .

The simulation results of D-PFA are then given for 34 instances of TSP, and the results of D-PFA are compared with the DSSA algorithm. In addition, simulation results for the D-PFA, D-GWO, and DSSA algorithms for 17 instances of TSP datasets are shown.

Then, in Section *B*, the simulation results of the D-PFA algorithm and its comparison with other eight metaheuristic algorithms are completely discussed.

A. EXPERIMENTAL CONFIGURATION

This section uses 34 instances of TSP benchmarks to test the proposed algorithm (D-PFA), which has 30 to 1002 cities. It is important to note that the D-PFA parameters, including the number of initial populations, distance concept, number of iterations, etc., are set experimentally by running the algorithm several times for each dataset. The suggested approach is implemented using Java programming language and tested on a desktop machine with an Intel i5-9600k @ 3.7 GHz and 16 GB RAM running 64-bit Windows 10. In the context of this research paper, it is imperative to note that the project's underlying source code is readily accessible and available for further scrutiny through the following GitHub repository: D_PFA. This open access to the code facilitates transparency, reproducibility, and collaboration within the scientific community.

D-PFA algorithm is compared with 12 metaheuristic algorithms named IBA [31], DFA [31], DICA [31], GA [31], ESA [31], IDGA [31], D-GWO [7], DSSA [7], CBA-NNM [44], CABC [45], DSOS [19] and HAACO, a more advanced version of ACO [63]. It is worth noting that all algorithms are executed 20 times for each instance.

In the comparisons performed, besides showing the BS and *Average Solution*, the percentage deviation of Best Solution (PDB(%)) is also obtained to show how close the solution obtained from algorithms is to the Best-Known Solution (BKS). Equation (13) is used to obtain PDB(%) [66]:

$$PDB(\%) = \frac{(Best - BKS)}{BKS} \times 100. \tag{13}$$

In this Equation, PDB(%) is the BS obtained for each TSP instance after 20 times of each algorithm. Where *Best* indicates the BS of each method over 20 times each TSP instance, and *Avg* represents the average length of each algorithm over 20 times each TSP instance.

To demonstrate the superiority of the D-PFA method over other algorithms, the percentage deviation of the *Average Solution* is computed using Equation (14):

$$PDA(\%) = \frac{(Avg - BKS)}{BKS} \times 100. \tag{14}$$

Here, PDA(%) is the average value length obtained for each TSP instance after 20 times.

Table 2 shows the simulation results of the proposed algorithm (D-PFA) along with the execution time of the algorithm 20 times for every instance. As shown in this Table, D-PFA was able to find the BS in 31 of 34 instances and was also able to find a better solution than BKS in seven instances. In just three instances, *kroA200*, *kroB200*, and *Pr1002*, D-PFA could not obtain the BKS solution, showing that the approach was well-converged and the result was stable.

As seen in Table 2, the D-PFA algorithm obtained the best-known values for instances up to 159 cities. For these instances, PDB is zero in 20 instances. For instances with 200 to 439 cities, results with PDB were between 0 and 0.054% for the best-known values. In addition to these instances, *Pr1002*, which has 1002 cities, was also investigated. The PDB value for this instance was found to be 0.671%.

From the (Run Time) column, we can see that D-PFA solved the instances in a respectable period of time, demonstrating its effectiveness.

Fig. 8 compares the time necessary to finish the D-PFA algorithm, which is presented based on the findings in Table 2. Part (a) of this figure displays findings obtained from the first 15 instances, whereas Part (b) displays runtimes produced from all instances combined.

Table 3 compares the simulation results of the D-PFA technique against the DSSA approach for 34 TSP instances. The D-PFA algorithm has a considerable advantage over the DSSA algorithm and achieved the optimum solution in 31 instances. In comparison, the DSSA algorithm found the BS in 26 instances.

Furthermore, the D-PFA obtained a better solution than BKS in seven instances, while the DSSA got a better solution than BKS in just one instance, *Dantzig42*.

Fig. 9 and Fig. 10 compare two D-PFA and DSSA algorithms for 34 instances based on PDB(%) and PDA(%). As shown, the D-PFA algorithm has less PDB(%) and PDA (%) than the DSSA algorithm, demonstrating its superiority in identifying the BS and stability in most instances.

Table 4 shows the simulation results of the D-PFA and its comparison with two algorithms named D-GWO and DSSA on 17 TSP instances after running 20 times of each algorithm per instance. As shown in Table 4, the proposed algorithm (D-PFA) achieved significant superiority over the two compared algorithms D-GWO and DSSA. Also, the D-PFA algorithm achieved a better solution than BKS in five instances of the benchmarks, and only in one instance, *Pr1002*, could not achieve the optimal solution. D-GWO algorithm has been able to reach the optimal solution in six instances out of 17 instances and in two instances *dantgiz42* and *pr107* reached a better solution than BKS. The DSSA algorithm has reached the optimal solution in 12 of the 17 instances and reached a better solution than BKS in one *dantgiz42* instance. This indicates the algorithm′s optimality and reaching proper convergence of the proposed algorithm compared with other methods.

In simulating the D-PFA and comparing it with other algorithms, the run time of the algorithms is not considered because the researchers used different computers to run their algorithms.

B. DISCUSSION OF NUMERICAL RESULTS

In this subsection, the simulation results of the D-PFA are compared to those of previous metaheuristic algorithms, and it is shown that the suggested method can reach optimum outcomes in most cases and effectively solve the problem.

Table 5 shows the simulation results of the D-PFA algorithm on three instances of TSP compared to other algorithms with similar parameters. As you can see in Table 5, D-PFA found a better solution than BKS in two instances and found a solution equal to BKS, for the instance *pr144*. D-PFA was also able to reduce its percentage deviation from BKS below zero. The run time criterion is not used here to compare the algorithms, as the authors used different computers to run and simulate the above algorithms.

To show the performance of D-PFA, we compared eight other metaheuristic algorithms on 10 instances of TSP with the proposed method. Table 6 shows the simulation results of the D-PFA and its comparison with the other eight metaheuristic algorithms. It should be noted that the simulation parameters of all algorithms are considered to be as similar as possible.

As you can see in Table 6, the D-PFA has solved nine of the 10 instances well and achieve the optimal solution, and only in one of the instances called *Pr1002* could not find the optimal solution. Also, this algorithm achieved a better answer than BKS in two instances. These achievements indicate the superiority of this algorithm and its good convergence as well.

The D-GWO has achieved optimal results in four instances and improved the *Pr107* and *Pr144* instances. The DSSA algorithm has reached the optimal solution in six instances and competed with the D-PFA algorithm. The GA could achieve the optimal solution only for one instance called

TABLE 2. Simulation results of D-PFA algorithm for 34 TSP instances from TSPLIB dataset.

S.N.	Instances	BKS	D-PFA						
			Best	Avg.	SD	Diff	PDB (%)	PDA (%)	Run Time (s)
1	Oliver30	420	**420**	420	0.00	0	0.000	0.000	0.42
2	Dantzig42	699	**671**	673	1.89	-28	-4.005	-3.719	0.50
3	Att48	33522	**33522**	33587	35.87	0	0.000	0.193	0.96
4	Rand50	5553	**5553**	5553	0.00	0	0.000	0.000	1.15
5	Eil51	426	**426**	426	0.00	0	0.000	0.000	1.37
6	Berlin52	7542	**7542**	7542	0.00	0	0.000	0.000	1.41
7	St70	675	**675**	675	0.00	0	0.000	0.000	2.21
8	Eil76	538	**538**	538	0.00	0	0.000	0.000	3.35
9	Pr76	108159	**108159**	108456	3.72	0	0.000	0.274	2.14
10	KroA100	21282	**21282**	21284	1.32	0	0.000	0.009	5.80
11	KroB100	22141	**22138**	22356	125.95	-2	-0.009	0.975	5.63
12	KroC100	20749	**20749**	20886	139.14	0	0.000	0.660	5.57
13	KroD100	21294	**21294**	21329	85.69	0	0.000	0.164	5.48
14	KroE100	22068	**22068**	22410	253.36	0	0.000	1.549	7.72
15	Eil101	629	**626**	626	0.00	-3	-0.476	-0.476	5.43
16	Lin105	14379	**14378**	14412	31.25	-1	-0.006	0.229	4.97
17	Pr107	44303	**44299**	44346	47.39	-4	-0.009	0.097	7.57
18	Pr124	59030	**59030**	59211	189.36	0	0.000	0.306	13.85
19	Ch130	6110	**6110**	6162	52.38	0	0.000	0.851	17.87
20	Pr136	96772	**96772**	98321	865.48	0	0.000	1.60	11.02
21	Pr144	58537	**58537**	58647	95.37	0	0.000	0.187	21.14
22	Ch150	6528	**6528**	6589	73.66	0	0.000	0.934	19.43
23	KroA150	26524	**26524**	26598	73.98	0	0.000	0.278	20.97
24	KroB150	26130	**26129**	26418	312.25	-1	-0.003	1.102	21.19
25	Pr152	73682	**73682**	74033	411.22	0	0.000	0.476	21.32
26	U159	42080	**42080**	42412	365.85	0	0.000	0.788	6.54
27	kroA200	29368	29384	29553	163.13	16	0.054	0.629	53.61
28	kroB200	29437	29444	29691	213.38	7	0.023	0.862	55.89
29	Tsp225	3916	**3916**	3922	6.72	0	0.000	0.153	51.24
30	Pr226	80369	**80369**	81135	768.93	0	0.000	0.953	46.77
31	Pr264	49135	**49132**	49176	43.58	-3	-0.006	0.083	71.09
32	Lin318	42029	**42029**	42621	589.98	0	0.000	1.408	168.90
33	Pr439	107217	**107217**	111367	4146.76	0	0.000	3.870	382.62
34	Pr1002	259047	260786	263982	3199.31	1739	0.671	1.905	4351.04

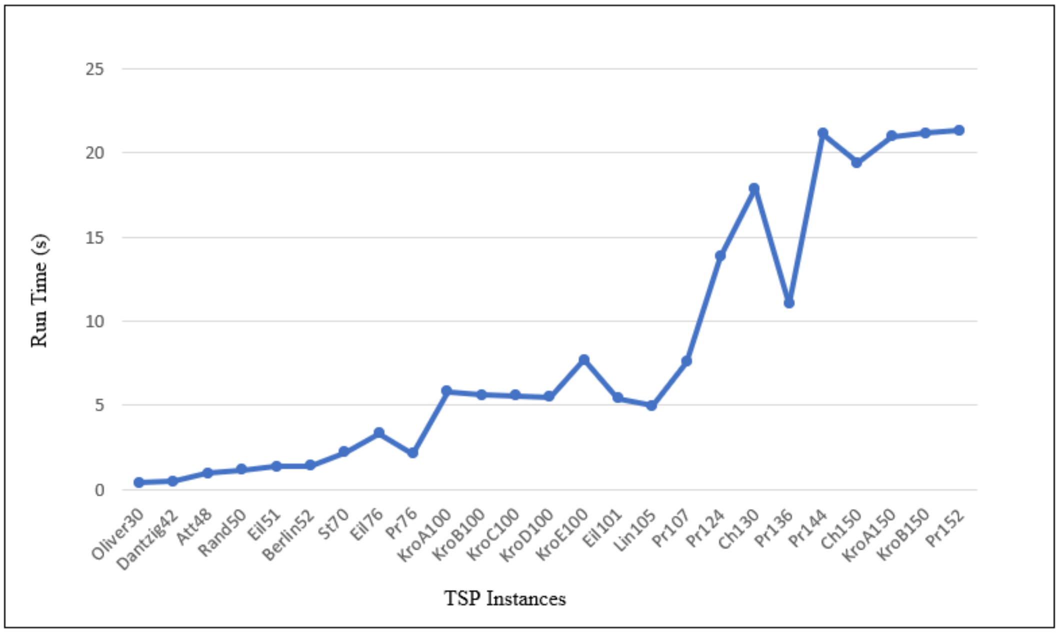
BKS: Represents the best-known solution from TSPLIB to date.
Best: Represents the best outcome of 20 algorithmic runs.
Avg.: Indicates the mean of 20 times.
SD: Indicates the standard deviation of 20 times.
Diff: Differentiates Best from BKS.
PDB (%): percentage deviation from the BKS based on Best.
PDA (%): percentage deviation from the BKS based on Avg.
Bold: The salesman's shortest distance travelled.
-: Indicates a lesser value than BKS.

Pr124 out of 10 and has had a low ability to find the optimal solution. The ESA has been able to achieve optimal results in two instances called *Kroc100* and *Pr124*, the IDGA could not achieve optimal results for any of the instances, which is a sign of the very weak ability of this algorithm. In this Table, 1BA was able to achieve the optimal solution in six instances and DFA and DICA were able to achieve the optimal solution for two instances: *Pr107* and *Pr124*.

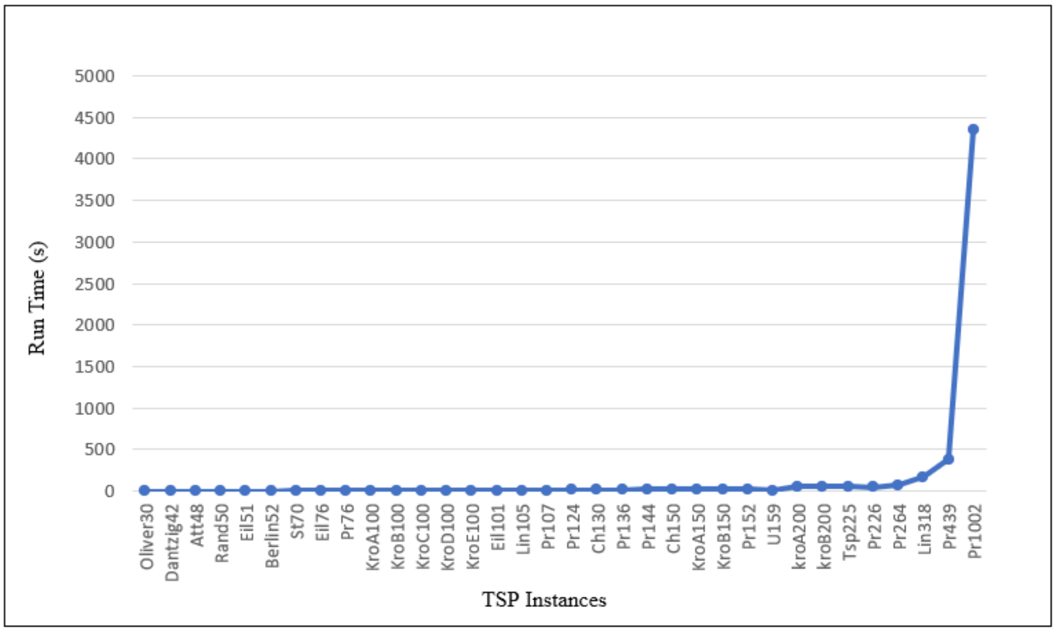
Table 7 compared the simulation results of D-PFA with other algorithms based on Avg Bks in 10 iterations. As you can see in this Table, the DSSA algorithm is superior to other algorithms, but it does not have much ability to find the Best, as seen in Table 6. Also, in this Table, D-PFA algorithm after DSSA for Avg. It has performed better than other algorithms and is highly able to find the optimal solution.

Tables 8 and 9 tabulate the percentage variation of PDB(%) and PDA(%) from the best known for all methods.

Table 8 compares the D-PFA based on PDB with other algorithms. As you can see, the D-PFA algorithm achieved a lower value than the compared algorithms in nine instances, and only in one instance called *Pr144*, the D-GWO algorithm achieved a value lower than D-PFA. After D-PFA, an algorithm that obtained a lower value for more instances, was the DSSA algorithm, which obtained lower values in four instances than other algorithms. D-GWO and IBA algorithms obtained the lower value in three instances. ESA algorithm in two instances and GA, DFA and DICA algorithms can obtain a lower value in one instance. Finally, the algorithm that performed very poorly in this comparison was the IDGA algorithm, which could not obtain a lower value than other algorithms in any instance.



(a)



(b)

FIGURE 8. a) Run time of D-PFA for first 15 instances, and b) Run time of D-PFA for all 34 instances combined.

In Table 9, we compared the proposed algorithm based on PDA with other algorithms. As you can see, the DSSA algorithm achieved a lower value than the compared algorithms in six instances. After DSSA, ESA and D-PFA algorithms obtained lower values in two instances. Other compared algorithms could not get a lower value in any of the instances.

Fig. 11 and Fig. 12 illustrate the percentage variation of PDB(%) and PDA(%) from the best known for all methods. These figures show that the D-PFA exhibits less variation than its counterparts.

Table 10, compared the proposed algorithm with four other metaheuristic algorithms namely, CBA-NNM [44], CABC [45], DSOS [19] and HAACO [63] on 16 instances compared with BKS.

As you can see, the D-PFA algorithm found the BKS solution in 11 instances and solved the problem well. The DSOS algorithm was able to obtain the BKS solution in

TABLE 3. D-PFA simulation results and comparison with DSSA for 34 TSP instances.

S.N.	Instances	BKS	DSSA					D-PFA				
			Best	Avg.	Diff	PDB (%)	PDA (%)	Best	Avg.	Diff	PDB (%)	PDA (%)
1	Oliver30	420	**420**	420	0	0.00	0.00	**420**	420	0	0.000	0.000
2	Dantzig42	699	675	675	-24	-3.43	-3.43	**671**	673	-28	-4.005	-3.719
3	Att48	33522	**33522**	33522	0	0.00	0.00	**33522**	33587	0	0.000	0.193
4	Rand50	5553	**5553**	5553	0	0.00	0.00	**5553**	5553	0	0.000	0.000
5	Eil51	426	**426**	426.60	0	0.00	0.14	**426**	426	0	0.000	0.000
6	Berlin52	7542	**7542**	7542	0	0.00	0.00	**7542**	7542	0	0.000	0.000
7	St70	675	**675**	675.15	0	0.00	0.02	**675**	675	0	0.000	0.000
8	Eil76	538	**538**	543.10	0	0.00	0.95	**538**	538	0	0.000	0.000
9	Pr76	108159	**108159**	108159	0	0.00	0.00	**108159**	108456	0	0.000	0.274
10	KroA100	21282	**21282**	21290.20	0	0.00	0.04	**21282**	21284	0	0.000	0.009
11	KroB100	22141	22141	22173.10	0	0.00	0.14	**22138**	22356	-2	-0.009	0.975
12	KroC100	20749	**20749**	20770.50	0	0.00	0.10	**20749**	20886	0	0.000	0.660
13	KroD100	21294	**21294**	21319.05	0	0.00	0.12	**21294**	21329	0	0.000	0.164
14	KroE100	22068	**22068**	22091.90	0	0.00	0.11	**22068**	22410	0	0.000	1.549
15	Eil101	629	634	641.50	5	0.79	1.99	**626**	626	-3	-0.476	-0.476
16	Lin105	14379	14379	14379	0	0.00	0.00	**14378**	14412	-1	-0.006	0.229
17	Pr107	44303	44303	44322	0	0.00	0.04	**44299**	44346	-4	-0.009	0.097
18	Pr124	59030	**59030**	59030	0	0.00	0.00	**59030**	59211	0	0.000	0.306
19	Ch130	6110	**6110**	6153.65	0	0.00	0.71	**6110**	6162	0	0.000	0.851
20	Pr136	96772	96920	97302.35	148	0.15	0.55	**96772**	98321	0	0.000	1.60
21	Pr144	58537	**58537**	58537	0	0.00	0.00	**58537**	58647	0	0.000	0.187
22	Ch150	6528	**6528**	6590.15	0	0.00	0.95	**6528**	6589	0	0.000	0.934
23	KroA150	26524	26525	26699.85	1	0.00	0.66	**26524**	26598	0	0.000	0.278
24	KroB150	26130	26130	26220.40	0	0.00	0.35	**26129**	26418	-1	-0.003	1.102
25	Pr152	73682	**73682**	73731.35	0	0.00	0.07	**73682**	74033	0	0.000	0.476
26	U159	42080	**42080**	42262.75	0	0.00	0.43	**42080**	42412	0	0.000	0.788
27	kroA200	29368	29459	29682.15	91	0.31	1.07	29384	29553	16	0.054	0.629
28	kroB200	29437	29564	29850.55	127	0.43	1.40	29444	29691	7	0.023	0.862
29	Tsp225	3916	**3916**	3926.05	0	0.00	0.26	**3916**	3922	0	0.000	0.153
30	Pr226	80369	**80369**	80369.20	0	0.00	0.00	**80369**	81135	0	0.000	0.953
31	Pr264	49135	49135	49271.85	0	0.00	0.28	**49132**	49176	-3	-0.006	0.083
32	Lin318	42029	42495	42742.70	466	1.11	1.70	**42029**	42621	0	0.000	1.408
33	Pr439	107217	107494	107844.90	277	0.26	0.59	**107217**	111367	0	0.000	3.870
34	Pr1002	259047	264212	266352.35	5165	1.99	2.82	260786	263982	1739	0.671	1.905

BKS: Represents the best-known solution from TSPLIB to date.
Best: Represents the best outcome of 20 algorithmic runs.
Avg.: Indicates the mean of 20 times.
Diff: Differentiates Best from BKS.
PDB (%): percentage deviation from the BKS based on Best.
PDA (%): percentage deviation from the BKS based on Avg.
Bold: The salesman's shortest distance travelled.
-: Indicates a lesser value than BKS.

TABLE 4. Simulation results of D-PFA and its comparison with D-GWO and DSSA.

S.N.	Instances	BKS	D-GWO					DSSA					D-PFA				
			Best	Avg.	Diff	PDB (%)	PDA (%)	Best	Avg.	Diff	PDA (%)	PDA (%)	Best	Avg.	Diff	PDB (%)	PDA (%)
1	Dantgiz42	699	679	680	-20	-2.86	-2.71	575	675.3	-24	−3.433	−3.398	**573**	673	-28	-4.005	-3.719
2	Att48	33522	35523	33600	1	0.002	0.23	**33522**	33522.0	0	0.000	0.000	**33522**	33587	0	0	0.193
3	Pr76	108159	**108159**	108900	0	0	0.68	**108159**	108163.3	0	0.000	0.004	**108159**	108456	0	0	0.274
4	Krob100	22140	22159	22444.6	19	0.085	1.37	22141	22173.1	0	0.000	0.149	**22138**	22356	-2	-0.009	0.975
5	Kroc100	20749	**20749**	21078	0	0	1.58	**20749**	20757.8	0	0.000	0.042	**20749**	20886	0	0	0.660
6	Kroe100	22068	22131	22410	63	0.28	1.54	**22068**	22094.0	0	0.000	0.118	**22068**	22410	0	0	1.549
7	Lin105	14379	14382	14520	3	0.02	0.98	14379	14383.4	0	0.000	0.031	**14378**	14412	-1	-0.006	0.229
8	Pr107	44303	44301	44685.1	-2	-0.004	0.86	44303	44377.6	0	0.000	0.168	**44299**	44346	-4	-0.009	0.097
9	Pr124	59030	**59030**	59390.9	0	0	0.61	**59030**	59036.9	0	0.000	0.012	**59030**	59211	0	0	0.306
10	Pr136	96772	97826	99310.5	1054	1.08	2.62	96785	97042.8	13	0.013	0.280	**96772**	98321	0	0	1.60
11	Pr144	58537	**58535**	58600.5	-2	-0.003	0.1	58537	58537.0	0	0.000	0.000	58537	58647	0	0	0.187
12	Krob105	26130	26320	26756.2	190	0.72	2.39	26135	26243.1	5	0.019	0.433	**26129**	26418	-1	-0.003	1.102
13	Pr152	73682	73690	74230	8	0.01	0.74	**73682**	73751.5	0	0.000	0.094	**73682**	74033	0	0	0.476
14	U159	42080	42142	42563.3	62	0.14	1.14	**42080**	42137.1	0	0.000	0.136	**42080**	42412	0	0	0.788
15	Pr226	80369	80648	81135.7	279	0.34	0.95	**80369**	80369.4	0	0.000	0.000	**80369**	81135	0	0	0.953
16	Pr439	107217	110415	112850.3	3198	2.98	5.25	107261	107451.9	44	0.041	0.219	**107217**	111367	0	0	3.870
17	Pr1002	259047	264922	267713.2	5875	2.26	3.34	263649	265370.5	4602	1.777	2.441	260786	263982	1739	0.671	1.905

TABLE 5. Simulation results of D-PFA algorithm on three instances of TSP compared to other metaheuristic algorithms.

S.N.	Instances	BKS	D-GWO				IBA				DFA				D-PFA			
			Best	Avg.	Diff	PDB (%)	Best	Avg.	Diff	PDB (%)	Best	Avg.	Diff	PDB (%)	Best	Avg.	Diff	PDB (%)
1	Dantgiz42	699	679	680	-20	-2.86	NA	NA	NA	NA	NA	NA	NA	NA	**671**	673	-28	-4.005
2	Pr107	44303	44301	44685.1	-2	-0.004	44303	44793.8	0	0	44303	44790	0	0	**44299**	44346	-4	-0.009
3	Pr144	58537	**58535**	58600.5	-2	-0.003	58537	58876.2	0	0	58546	58993	9	0.015	58537	58647	0	0

NA: The term not applicable appears often in the Table.

seven instances. HAACO, CBA-NNM and CABC algorithms reached the solution of BKS in four, three and two instances, respectively, and the CBA-NNM algorithm achieved a solution lower than BKS in two instances *KroB100* and *KroA200*.

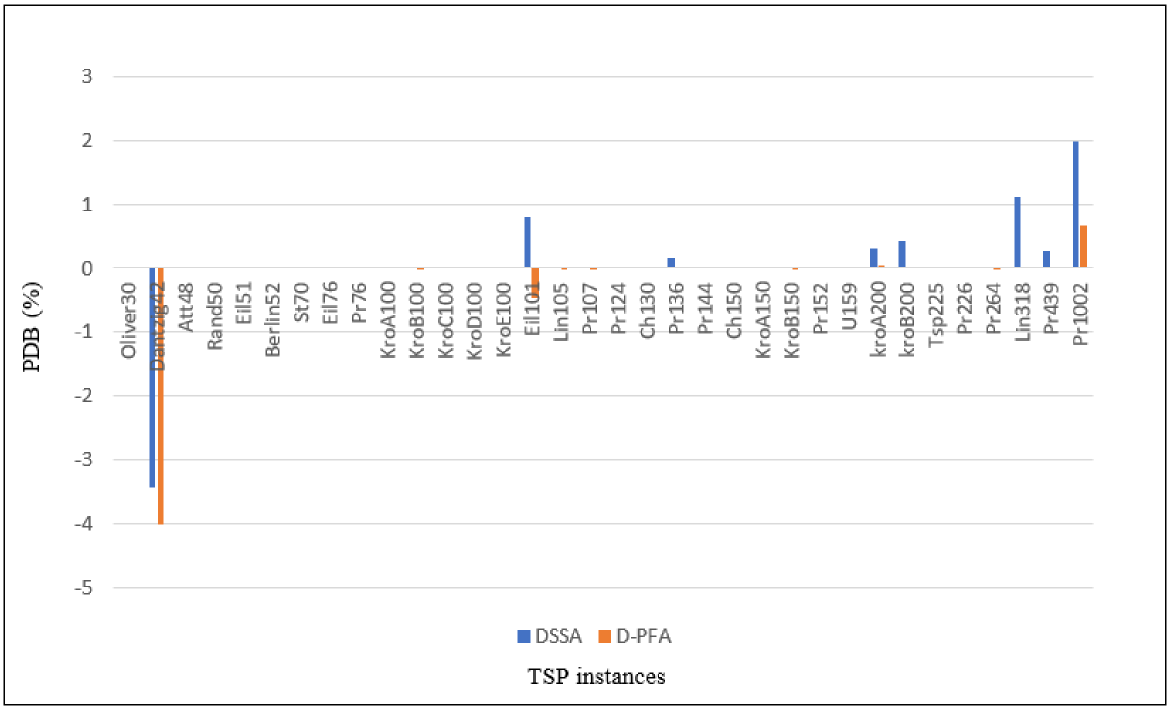


FIGURE 9. Comparison results of D-PFA and DSSA algorithms based on percentage deviation of best solution (PDB(%)) for 34 instances of TSP.

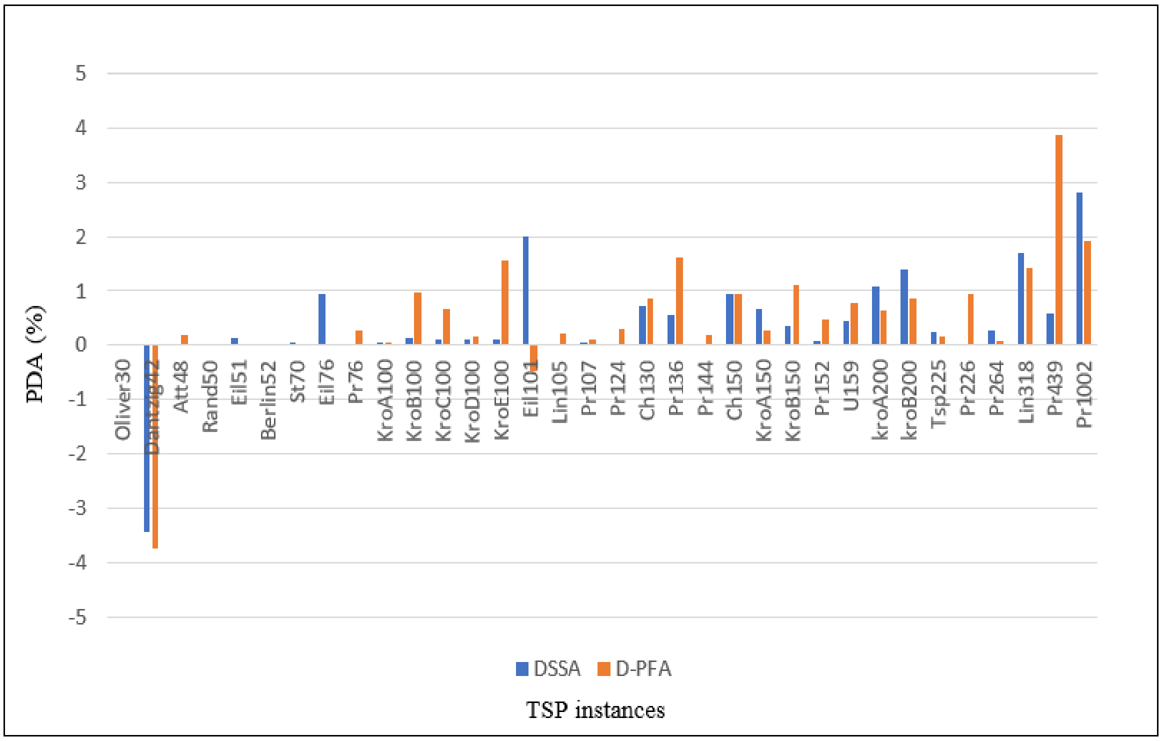


FIGURE 10. Comparison results of D-PFA and DSSA algorithms based on percentage deviation of the average solution (PDA(%)) for 34 instances of TSP.

In the following, Table 11 compares the proposed algorithm for 16 instances with four other metaheuristic algorithms named CBA-NNM [44], CABC [45], DSOS [19] and HAACO [63] for PDB and PDA values it shows. As you can see in this Table, the D-PFA algorithm obtained the PDB value for all 16 instances less than the other four algorithms.

TABLE 6. Simulation results of D-PFA and its comparison with other algorithms based on Best.

S.N.	Instance	BKS	D-GWO	DSSA	GA	ESA	IDGA	IBA	DFA	DICA	D-PFA
			Best	Best	Best	Best	Best	Best	Best	Best	Best
1	Krob100	22140	22159	22141	22176	22202	22208	22140	22183	22180	**22138**
2	Kroc100	20749	**20749**	**20749**	20861	**20794**	20830	**20749**	20756	20756	**20749**
3	Kroe100	22068	22131	**22068**	22150	22099	22110	**22068**	22079	22083	**22068**
4	Pr107	44303	44301	44303	44392	44413	44428	44303	44303	44303	**44299**
5	Pr124	59030	**59030**	**59030**	**59030**	**59030**	59072	**59030**	**59030**	**59030**	**59030**
6	Pr136	96772	97826	96785	98432	98499	98532	97547	97716	97736	**96772**
7	Pr144	58537	58535	58537	58599	58574	58581	58537	58546	58563	58537
8	Pr152	73682	73690	**73682**	74520	74172	74249	73921	74033	74052	**73682**
9	Pr439	107217	110415	107261	113576	113497	113207	111538	111967	111983	**107217**
10	Pr1002	259047	264922	263649	273001	273496	272893	270016	272003	272080	260786

TABLE 7. Simulation results of D-PFA and its comparison with other algorithms based on Avg.

S.N.	Instance	BKS	D-GWO	DSSA	GA	ESA	IDGA	IBA	DFA	DICA	D-PFA
			Avg.	Avg.	Avg.	Avg.	Avg.	Avg.	Avg.	Avg.	Avg.
1	Krob100	22140	22444.6	**22173.1**	22687.4	22602	22712.6	22506.4	22604	22599.7	22356
2	Kroc100	20749	21078	**20757.8**	21510.4	21170.4	21298.7	21050	21096.3	21103.9	20886
3	Kroe100	22068	22410	**22094.0**	22741.3	22499.7	22721.9	22349.6	22413	22456.3	22410
4	Pr107	44303	44685.1	44377.6	45620	44821.5	44902.5	44793.8	44790.4	44803.3	**44346**
5	Pr124	59030	59390.9	**59036.9**	59910	59593.6	59912.8	59412.1	59404.3	59436.9	59211
6	Pr136	96772	99310.5	**97042.8**	100472.4	99858.3	99932.7	99351.2	99683.7	99583.7	98321
7	Pr144	58537	58600.5	**58537.0**	60591.4	58807.3	58893	58876.2	58993.3	59070.9	58647
8	Pr152	73682	74230	**73751.5**	75658.3	74969.5	75126.7	74676.9	74934.3	74886.7	74033
9	Pr439	107217	112850.3	**107451.9**	116943.4	116706.9	116436.1	115256.4	115558.2	115763.1	111367
10	Pr1002	259047	267713.2	265370.5	279384.7	279419.7	278951.4	274419.7	277344.7	277308.1	**263982**

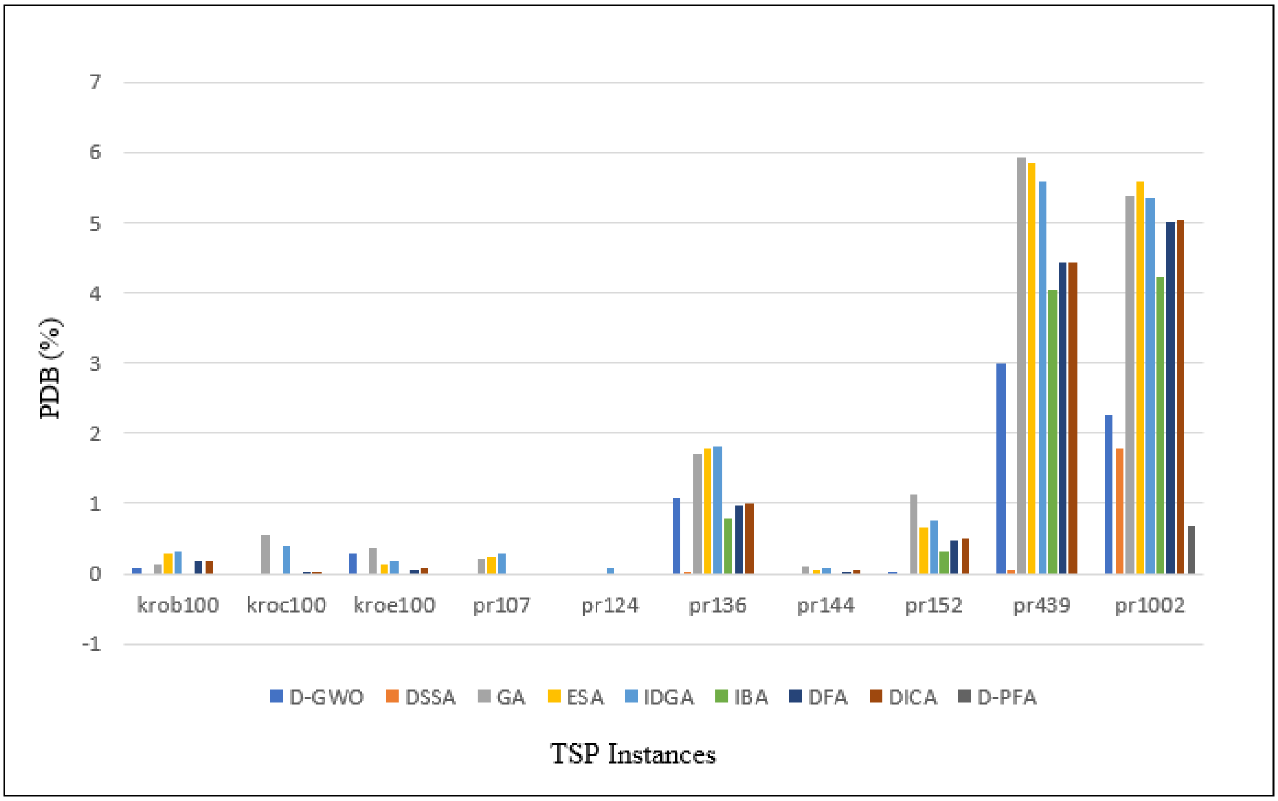


FIGURE 11. Results of percentage deviation of best solution (PDB(%)) for 10 instances of the TSP.

After the D-PFA algorithm, the algorithm that achieved superiority in more instances is the DSOS algorithm, which obtained a lower value for PDB in seven instances than the CBA-NNM and HAACO algorithms. Finally, two algorithms, CBA-NNM and HAACO, obtained the zero value for PDB equally in three instances. Now, according to the

TABLE 8. Simulation results of D-PFA and its comparison with other algorithms based on PDB (%).

S.N.	Instance	BKS	D-GWO	DSSA	GA	ESA	IDGA	IBA	DFA	DICA	D-PFA
			PDB (%)	PDB (%)	PDB (%)	PDB (%)	PDB (%)	PDB (%)	PDB (%)	PDB (%)	PDB (%)
1	Krob100	22140	0.085	0.000	0.132	0.28	0.307	0	0.194	0.18	**-0.009**
2	Kroc100	20749	**0**	**0.000**	0.539	**0**	0.39	**0**	0.033	0.033	**0**
3	Kroe100	22068	0.28	**0.000**	0.371	0.14	0.19	**0**	0.0498	0.067	**0**
4	Pr107	44303	-0.004	0.000	0.2	0.248	0.282	0	0	0	**-0.009**
5	Pr124	59030	**0**	**0.000**	**0**	**0**	0.071	**0**	**0**	**0**	**0**
6	Pr136	96772	1.08	0.013	1.715	1.784	1.818	0.8	0.975	0.996	**0**
7	Pr144	58537	**-0.003**	0.000	0.105	0.063	0.075	0	0.015	0.044	0
8	Pr152	73682	0.01	**0.000**	1.137	0.665	0.7699	0.324	0.476	0.502	**0**
9	Pr439	107217	2.98	0.041	5.93	5.857	5.586	4.03	4.43	4.445	**0**
10	Pr1002	259047	2.26	1.777	5.386	5.577	5.344	4.234	5.001	5.031	**0.671**

TABLE 9. Simulation results of D-PFA and its comparison with other algorithms based on PDA (%).

S.N.	Instance	BKS	D-GWO	DSSA	GA	ESA	IDGA	IBA	DFA	DICA	D-PFA
			PDA (%)	PDA (%)	PDA (%)	PDA (%)	PDA (%)	PDA (%)	PDA (%)	PDA (%)	PDA (%)
1	Krob100	22140	1.37	**0.149**	2.472	0.28	0.307	1.645	2.095	2.034	0.975
2	Kroc100	20749	1.58	0.042	3.669	**0**	0.39	1.45	1.673	1.681	0.660
3	Kroe100	22068	1.54	**0.118**	3.051	0.14	0.19	1.27	1.563	1.729	1.549
4	Pr107	44303	0.86	0.168	2.971	0.248	0.282	1.107	1.1	1.116	**0.097**
5	Pr124	59030	0.61	0.012	1.49	**0**	0.071	0.647	0.634	0.684	0.306
6	Pr136	96772	2.62	**0.280**	3.823	1.784	1.818	2.665	3.008	2.823	1.60
7	Pr144	58537	0.1	**0.000**	3.509	0.063	0.075	0.579	0.779	0.903	0.187
8	Pr152	73682	0.74	**0.094**	2.682	0.665	0.7699	1.35	1.699	1.608	0.476
9	Pr439	107217	5.25	**0.219**	9.071	5.857	5.586	7.498	7.779	7.382	3.870
10	Pr1002	259047	3.34	2.441	7.85	5.577	5.344	5.934	7.036	6.585	**1.905**

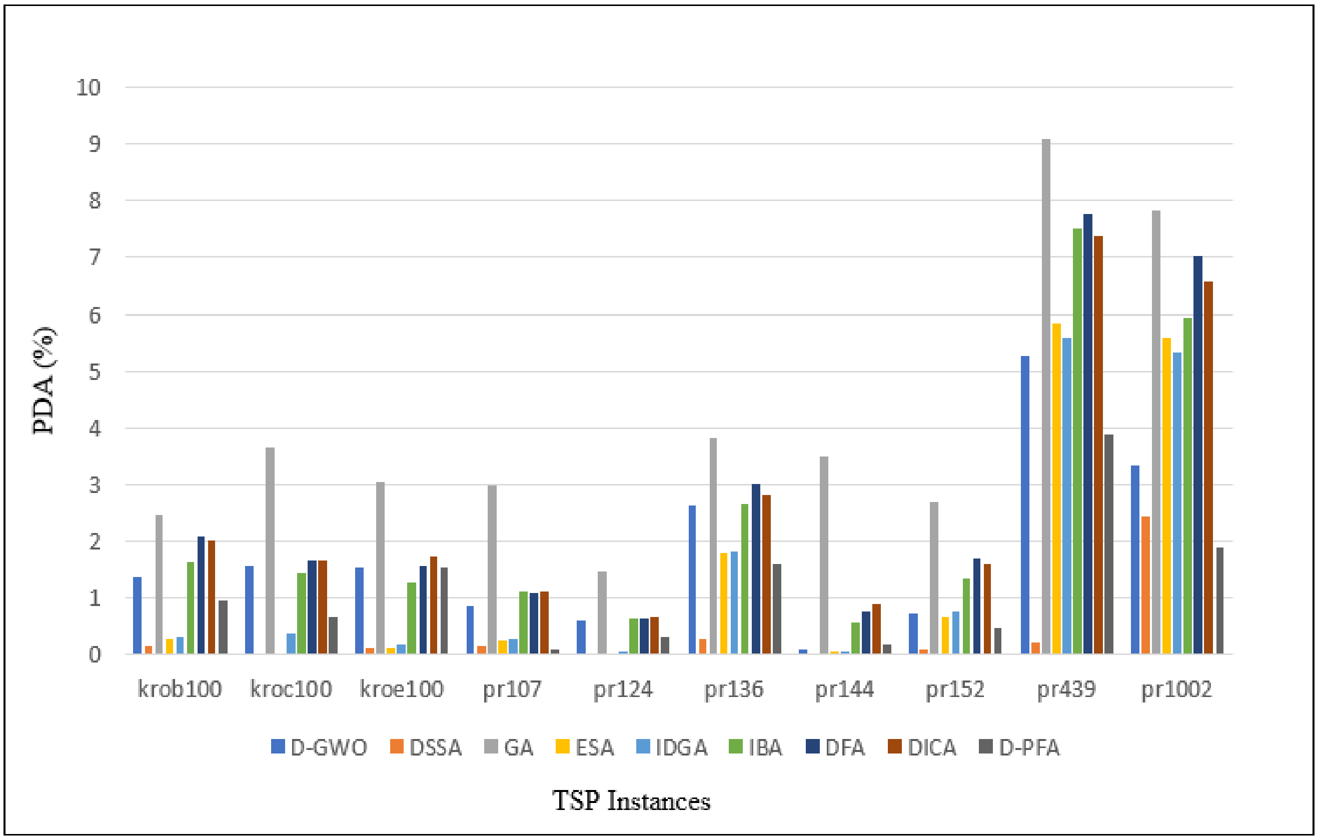


FIGURE 12. Results of Percentage deviation of the average solution (PDA(%)) for 10 instances of TSP.

above Table, it can be seen that the D-PFA and CBA-NNM algorithms were equally able to obtain a PDA value less than the other two algorithms for five instances. Then, CABC and HAACO algorithms obtained a lower value for PDA than other algorithms in three and one instances, respectively.

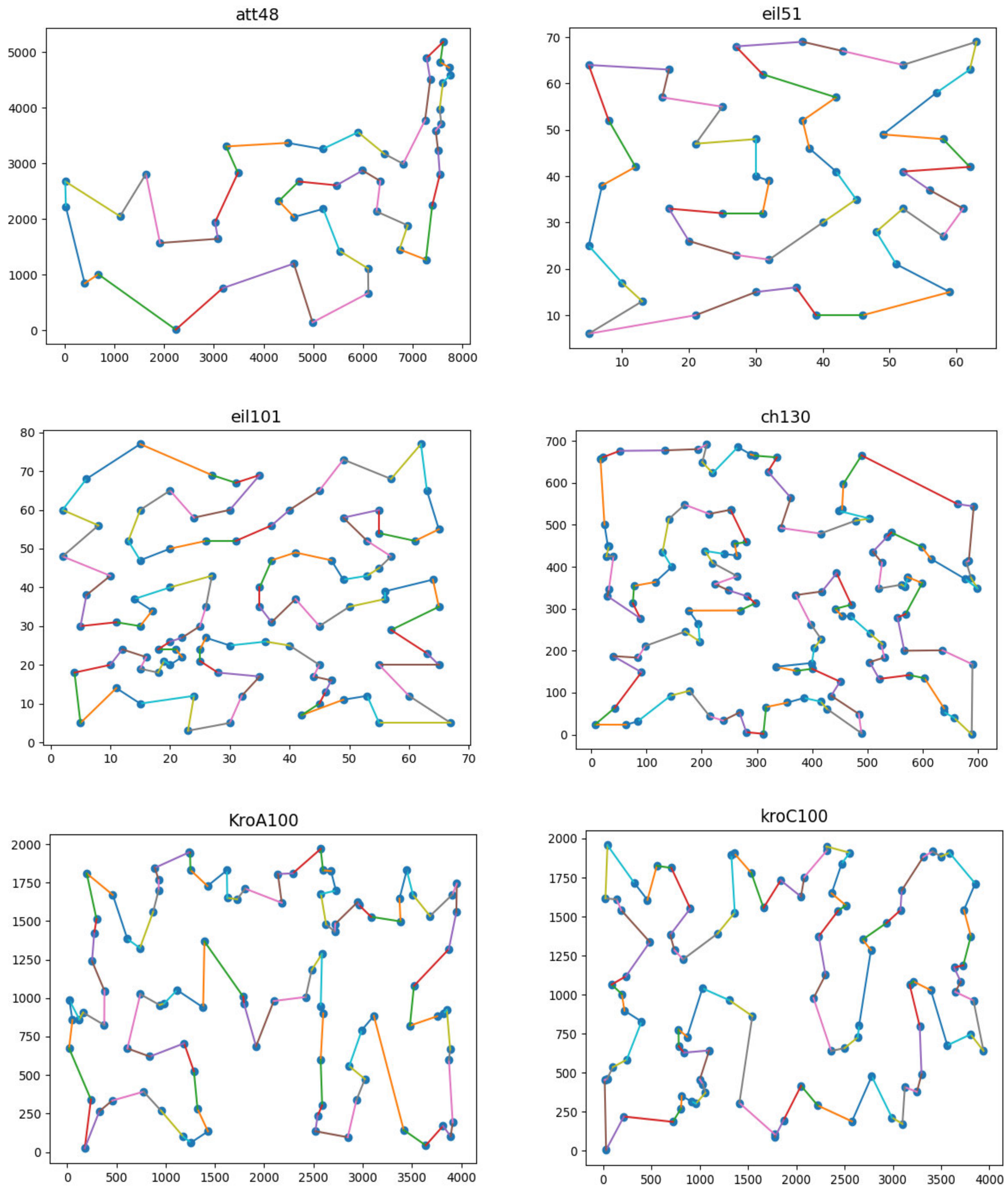


FIGURE 13. The best solution is obtained by D-PFA for att48, eil51, eil101, ch130, kroA100 and kroC100.

Fig. 13 shows the optimized routes obtained by D-PFA for six instances (*att48, eil51, eil101, ch130, kroA100* and *kroC100*). The number of iterations of the algorithm for each instance was considered equal to 200. As you

TABLE 10. Simulation results of D-PFA and its comparison with other algorithms based on Best.

S.N.	Instance	BKS	CBA-NNM	HAACO	CABC	DSOS	D-PFA
			Best	**Best**	**Best**	**Best**	**Best**
1	Berlin52	7542	7544	**7542**	**7542**	**7542**	**7542**
2	St70	675	677	**675**	NA	**675**	**675**
3	KroA100	21282	21285	**21282**	**21282**	**21282**	**21282**
4	KroB100	22141	**22139**	NA	NA	22140	22141
5	KroC100	20749	20751	NA	NA	**20749**	**20749**
6	KroD100	21294	21375	NA	NA	**21294**	**21294**
7	KroE100	22068	22079	NA	NA	**22068**	**22068**
8	Lin105	14379	14383	**14379**	NA	NA	**14379**
9	Pr124	59030	59075	NA	NA	**59030**	**59030**
10	Pr144	58535	**58535**	NA	58537	58565	58537
11	Ch150	6528	6552	6566	6542	NA	**6528**
12	KroB150	26130	26550	NA	26186	NA	**26130**
13	KroA200	29368	**26140**	29483	29467	NA	29459
14	Pr226	80369	80858	NA	893705	NA	**80369**
15	Lin318	42029	42756	NA	42756	42201	42495
16	Pr1002	259047	273825	NA	NA	272381	264212

NA: The term not applicable appears often in the Table.
-: Indicates a lesser value than BKS.

TABLE 11. Comparison D-PFA with CBANNM, HAACO, CABC, and DSOS.

S.N.	Instance	BKS	CBA-NNM		HAACO		CABC		DSOS		D-PFA	
			PDB (%)	**PDA (%)**	**PDB (%)**	**PDA (%)**	**PDB (%)**	**PDA (%)**	**PDB (%)**	**PDA (%)**	**PDB (%)**	**PDA (%)**
1	Berlin52	7542	0.03	0.03	**0**	**0**	**0**	**0**	**0**	0.01	**0.000**	**0.000**
2	St70	675	0.31	0.32	**0**	0.22	–	–	**0**	0.62	**0.000**	**0.000**
3	KroA100	21282	0.02	0.02	**0**	0.39	**0**	0.04	**0**	0.60	**0.000**	**0.009**
4	KroB100	22141	**0**	**0.32**	–	–	–	–	0	0.90	**-0.009**	0.975
5	KroC100	20749	0.01	0.51	–	–	–	–	**0**	0.64	**0.000**	0.660
6	KroD100	21294	**0**	0.12	–	–	–	–	**0**	0.94	**0.000**	0.164
7	KroE100	22068	0.05	**0.29**	–	–	–	–	**0**	0.74	**0.000**	1.549
8	Lin105	14379	0.03	**0.12**	0	0.23	–	–	–	–	**-0.006**	0.229
9	Pr124	59030	0.08	**0.30**	–	–	–	–	**0**	0.68	**0.000**	0.306
10	Pr144	58535	**0**	0.26	–	–	0.09	**0.16**	0.05	0.48	**0.000**	0.187
11	Ch150	6528	0.37	**0.39**	0.58	1.13	0.21	0.44	–	–	**0.000**	0.934
12	KroB150	26130	0.04	**0.66**	–	–	0.31	0.69	–	–	**-0.003**	1.102
13	KroA200	29368	0.29	0.56	0.39	0.9	0.34	**0.51**	-	-	**0.054**	0.629
14	Pr226	80369	0.61	0.91	–	–	0.61	**0.78**	-	-	**0.000**	0.953
15	Lin318	42029	1.73	2.23	-	-	1.86	2.35	0.41	2.24	**0.000**	**1.408**
16	Pr1002	259047	5.70	7.13	-	-	-	-	5.15	7.46	**0.671**	**1.905**

-: Indicates a lesser value than BKS.

can see, D-PFA obtained the optimal path and reached convergence.

VII. CONCLUSION AND OUTLOOK

TSP is an established combinatorial optimization problem. The primary challenge is to find a Hamiltonian route with the lowest cost on a weighted graph. TSP is a significant benchmark problem because it models important problems such as scheduling, computer wiring, routing, threading of scan cells in a testable Very-Large-Scale-Integrated (VLSI), people movement, automatic drilling of printed circuit boards and circuits, and *X-ray* crystallography [67].

This paper suggests a novel discrete PFA approach (D-PFA) for solving the symmetric TSP. The PFA is inspired by search behavior in the hunting or feeding area led by an individual in animal herds. Unlike other algorithms, this algorithm has a leader, and the other participants follow him. But not all particles move regularly; they all move randomly. The algorithm has a high degree of flexibility, a quick response time, strong exploration and exploitation. It can provide more accurate solutions and answers without getting stuck in a local optimum that inspired us to propose a discrete version. In the suggested discrete approach, the original PFA was broken into four sub-algorithms, coupled with discretized sub-algorithms to form a new technique.

The proposed algorithm is run on a desktop machine with an Intel i5-9600k @ 3.7 GHz and 16 GB RAM running 64-bit Windows 10. To test the performance of the proposed D-PFA, experiments are conducted on multiple TSP benchmarks to validate the differences between the proposed D-PFA and other cutting-edge algorithms such as the IBA, DFA, DICA, GA, ESA, IDGA, D-GWO, DSSA, CBA-NNM, CABC, DSOS and HAACO. The analyses provide the following notable results:

In our first set of experiments, 34 TSP datasets are used to test the proposed algorithm (D-PFA), which has 30 to 1002 cities. D-PFA algorithm is compared with 12 meta-heuristic algorithms named IBA, DFA, DICA, GA, ESA, IDGA, D-GWO, DSSA, CBA-NNM, CABC, DSOS and HAACO. As our study shown, D-PFA was able to find the BS in 31 of 34 instances and was also able to find a better solution than BKS in seven instances. In just three instances, D-PFA could not obtain the BKS solution, showing that the approach was well-converged and the result was stable.

(1) From the presented Run Time test, we can understand that D-PFA solved the instances in a respectable period, demonstrating its effectiveness in response time compared to other methods.
(2) In another experiment, we compare D-PFA with DSSA on 34 instances based on PDB(%) and PDA(%). As shown in the paper, the D-PFA algorithm has less PDB (%) and PDA(%) than the DSSA algorithm, demonstrating its superiority in identifying the BS and stability in most instances.
(3) We also compare the results of the D-PFA with two algorithms named D-GWO and DSSA on 17 TSP instances after running 20 times of each algorithm per instance. As shown in the paper, the proposed algorithm (D-PFA) achieved significant superiority over the two compared algorithms D-GWO and DSSA. Also, the D-PFA algorithm achieved a better solution than BKS in five instances of the benchmarks, and only in one instance, it could not achieve the optimal solution. D-GWO algorithm has reached the optimal solution in six out of 17 instances and in two instances reached a better solution than BKS. The DSSA algorithm has reached the optimal solution in 12 out of the 17 instances and also reached a better solution than BKS in one instance. This indicates the algorithm′s optimality and reaching proper convergence of the proposed algorithm compared with other methods.
(4) To show the performance of D-PFA, we compared all 12 metaheuristic algorithms on another 10 and 17 instances of TSP with the proposed method. It should be noted that the simulation parameters of all algorithms are considered to be as similar as possible. As shown in the paper, the D-PFA has solved nine out of the 10 instances well and achieved the optimal solution based on calculated parameters of PDA, PDB, Best and Avg for every instance. Also, this algorithm achieved a better answer than BKS in two instances. These achievements indicate the superiority of this algorithm and its good convergence as well.

Based on the primary limitations of this paper, the following suggestions for further research are offered. The proposed resilient optimization strategy may be contrasted with other uncertainty methods as fuzzy theory [68], [69], grey systems [70], [71], and stochastic optimum control [72]. Since D-PFA was developed to solve symmetric TSP, more study is necessary to establish D-PFA′s in tackling constraints like greenness [73] and time frames [74] for efficient large-scale problem solving. Future research can also focus on adapting the proposed D-PFA to solve different discrete optimization problems and comparing it to other methodologies that solve real-world problems, such as the VRP [75], [76], [77] allocation and scheduling in cloud computing [78], [79], internet of things [80], open shop [66], [81], and other scheduling and routing problems, to demonstrate its efficiency and effectiveness in other engineering problems.

REFERENCES

[1] K. Panwar and K. Deep, “Discrete grey wolf optimizer for symmetric travelling salesman problem,” *Appl. Soft Comput.*, vol. 105, Jul. 2021, Art. no. 107298.
[2] T. Mostafaie, F. M. Khiyabani, and N. J. Navimipour, “A systematic study on meta-heuristic approaches for solving the graph coloring problem,” *Comput. Oper. Res.*, vol. 120, Aug. 2020, Art. no. 104850.
[3] M. A. Şahman, “A discrete spotted hyena optimizer for solving distributed job shop scheduling problems,” *Appl. Soft Comput.*, vol. 106, Jul. 2021, Art. no. 107349.
[4] V. K. Patel and B. D. Raja, “Comparative performance of recent advanced optimization algorithms for minimum energy requirement solutions in water pump switching network,” *Arch. Comput. Methods Eng.*, vol. 28, no. 3, pp. 1545–1559, May 2021.
[5] C. Ammari, D. Belatrache, B. Touhami, and S. Makhloufi, “Sizing, optimization, control and energy management of hybrid renewable energy system—A review,” *Energy Built Environ.*, vol. 3, no. 4, pp. 399–411, Oct. 2022.
[6] T. Adamo, G. Ghiani, P. Greco, and E. Guerriero, “Learned upper bounds for the time-dependent travelling salesman problem,” *IEEE Access*, vol. 11, pp. 2001–2011, 2023.
[7] Z. Zhang and Y. Han, “Discrete sparrow search algorithm for symmetric traveling salesman problem,” *Appl. Soft Comput.*, vol. 118, Mar. 2022, Art. no. 108469.
[8] S. Arora, “Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems,” *J. ACM (JACM)*, vol. 45, no. 5, pp. 753–782, 1998.
[9] Y. Saji and M. Barkatou, “A discrete bat algorithm based on Lévy flights for Euclidean traveling salesman problem,” *Expert Syst. Appl.*, vol. 172, Jun. 2021, Art. no. 114639.
[10] P. Pirozmand, “An improved particle swarm optimization algorithm for task scheduling in cloud computing,” *J. Ambient Intell. Humanized Comput.*, vol. 14, no. 4, pp. 4313–4327, 2023.
[11] G. Laporte, “The traveling salesman problem: An overview of exact and approximate algorithms,” *Eur. J. Oper. Res.*, vol. 59, no. 2, pp. 231–247, Jun. 1992.
[12] M. Padberg and G. Rinaldi, “Optimization of a 532-city symmetric traveling salesman problem by branch and cut,” *Operations Res. Lett.*, vol. 6, no. 1, pp. 1–7, Mar. 1987.
[13] G. Reinelt, *The Traveling Salesman: Computational Solutions for TSP Applications*, vol. 840. Cham, Switzerland: Springer, 2003.
[14] G. Laporte and Y. Nobert, “A cutting planes algorithm for the m-Salesmen problem,” *J. Oper. Res. Soc.*, vol. 31, no. 11, pp. 1017–1023, Nov. 1980.
[15] R. E. Bellman and S. E. Dreyfus, *Applied Dynamic Programming*, vol. 2050. Princeton, NJ, USA: Princeton Univ. Press, 2015.
[16] Ö. Ergun and J. B. Orlin, “A dynamic programming methodology in very large scale neighborhood search applied to the traveling salesman problem,” *Discrete Optim.*, vol. 3, no. 1, pp. 78–85, Mar. 2006.
[17] X. Zhou, D. Y. Gao, C. Yang, and W. Gui, “Discrete state transition algorithm for unconstrained integer optimization problems,” *Neurocomputing*, vol. 173, pp. 864–874, Jan. 2016.
[18] K. Sörensen, “Metaheuristics—The metaphor exposed,” *Int. Trans. Oper. Res.*, vol. 22, no. 1, pp. 3–18, Jan. 2015.
[19] A. E.-S. Ezugwu and A. O. Adewumi, “Discrete symbiotic organisms search algorithm for travelling salesman problem,” *Expert Syst. Appl.*, vol. 87, pp. 70–78, Nov. 2017.
[20] Y. Wang, “The hybrid genetic algorithm with two local optimization strategies for traveling salesman problem,” *Comput. Ind. Eng.*, vol. 70, pp. 124–133, Apr. 2014.
[21] A. A. R. Hosseinabadi, J. Vahidi, B. Saemi, A. K. Sangaiah, and M. Elhoseny, “Extended genetic algorithm for solving open-shop scheduling problem,” *Soft Comput.*, vol. 23, no. 13, pp. 5099–5116, Jul. 2019.
[22] E. Hindi and AlSalman, “Dynamic flying ant colony optimization (DFACO) for solving the traveling salesman problem,” *Sensors*, vol. 19, no. 8, p. 1837, Apr. 2019.
[23] H. Eldem and E. Ülker, “The application of ant colony optimization in the solution of 3D traveling salesman problem on a sphere,” *Eng. Sci. Technol., Int. J.*, vol. 20, no. 4, pp. 1242–1248, Aug. 2017.

[24] K. Tang, Z. Li, L. Luo, and B. Liu, "Multi-strategy adaptive particle swarm optimization for numerical optimization," *Eng. Appl. Artif. Intell.*, vol. 37, pp. 9–19, Jan. 2015.

[25] L. Wang, R. Cai, M. Lin, and Y. Zhong, "Enhanced list-based simulated annealing algorithm for large-scale traveling salesman problem," *IEEE Access*, vol. 7, pp. 144366–144380, 2019.

[26] X.-S. Yang, "A new metaheuristic bat-inspired algorithm," in *Nature Inspired Cooperative Strategies for Optimization (NICSO)*. Cham, Switzerland: Springer, 2010, pp. 65–74.

[27] A. K. Sangaiah, M. Sadeghilalimi, A. A. R. Hosseinabadi, and W. Zhang, "Energy consumption in point-coverage wireless sensor networks via bat algorithm," *IEEE Access*, vol. 7, pp. 180258–180269, 2019.

[28] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Erciyes Univ., Eng. Fac., Comput., Kayseri, Türkiye, Tech. Rep. tr06, 2005,

[29] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, Mar. 2014.

[30] M.-Y. Cheng and D. Prayogo, "Symbiotic organisms search: A new metaheuristic optimization algorithm," *Comput. Struct.*, vol. 139, pp. 98–112, Jul. 2014.

[31] E. Osaba, X.-S. Yang, F. Diaz, P. Lopez-Garcia, and R. Carballedo, "An improved discrete bat algorithm for symmetric and asymmetric traveling salesman problems," *Eng. Appl. Artif. Intell.*, vol. 48, pp. 59–71, Feb. 2016.

[32] T.-T. Nguyen, "A hybridized parallel bats algorithm for combinatorial problem of traveling salesman," *J. Intell. Fuzzy Syst.*, vol. 38, no. 5, pp. 5811–5820, May 2020.

[33] Y. Saji and M. E. Riffi, "A novel discrete bat algorithm for solving the travelling salesman problem," *Neural Comput. Appl.*, vol. 27, no. 7, pp. 1853–1866, Oct. 2016.

[34] A. E.-S. Ezugwu, A. O. Adewumi, and M. E. Frîncu, "Simulated annealing based symbiotic organisms search optimization algorithm for traveling salesman problem," *Expert Syst. Appl.*, vol. 77, pp. 189–210, Jul. 2017.

[35] J. Faigl, "GSOA: Growing self-organizing array–unsupervised learning for the close-enough traveling salesman problem and other routing problems," *Neurocomputing*, vol. 312, pp. 120–134, Oct. 2018.

[36] B. H. Abed-Alguni and F. Alkhateeb, "Novel selection schemes for cuckoo search," *Arabian J. Sci. Eng.*, vol. 42, no. 8, pp. 3635–3654, Aug. 2017.

[37] Y. Zhong, "Hybrid discrete artificial bee colony algorithm with threshold acceptance criterion for traveling salesman problem," *Inf. Sci.*, vol. 421, pp. 70–84, Dec. 2017.

[38] Z. Yang, "A double-loop hybrid algorithm for the traveling salesman problem with arbitrary neighbourhoods," *Eur. J. Oper. Res.*, vol. 265, no. 1, pp. 65–80, Feb. 2018.

[39] K. Panwar and K. Deep, "Transformation operators based grey wolf optimizer for travelling salesman problem," *J. Comput. Sci.*, vol. 55, Oct. 2021, Art. no. 101454.

[40] A. A. R. Hosseinabadi, M. Kardgar, M. Shojafar, S. Shamshirband, and A. Abraham, "GELS-GA: Hybrid metaheuristic algorithm for solving multiple travelling salesman problem," in *Proc. 14th Int. Conf. Intell. Syst. Design Appl.*, Nov. 2014, pp. 76–81.

[41] A. S. Rostami, "Solving multiple traveling salesman problem using the gravitational emulation local search algorithm," *Appl. Math. Inf. Sci.*, vol. 9, no. 2, pp. 1–11, 2015.

[42] X. Chen, P. Zhang, G. Du, and F. Li, "Ant colony optimization based memetic algorithm to solve bi-objective multiple traveling salesmen problem for multi-robot systems," *IEEE Access*, vol. 6, pp. 21745–21757, 2018.

[43] H. Yapici and N. Cetinkaya, "A new meta-heuristic optimizer: Pathfinder algorithm," *Appl. Soft Comput.*, vol. 78, pp. 545–568, May 2019.

[44] M. Sahin, "Solving TSP by using combinatorial bees algorithm with nearest neighbor method," *Neural Comput. Appl.*, vol. 35, no. 2, pp. 1863–1879, Jan. 2023.

[45] D. Karaboga and B. Gorkemli, "Solving traveling salesman problem by using combinatorial artificial bee colony algorithms," *Int. J. Artif. Intell. Tools*, vol. 28, no. 1, 2019, Art. no. 1950004.

[46] X. Ma, C. Zhang, F. Yao, and Z. Li, "Multi-agent task allocation based on discrete DEPSO in epidemic scenarios," *IEEE Access*, vol. 10, pp. 131181–131191, 2022.

[47] M. Mahi, Ö. K. Baykan, and H. Kodaz, "A new hybrid method based on particle swarm optimization, ant colony optimization and 3-Opt algorithms for traveling salesman problem," *Appl. Soft Comput.*, vol. 30, pp. 484–490, May 2015.

[48] Y. Zhang, Z. Zhang, Z. Liu, and Q. Chen, "An asymptotically tight online algorithm for m-Steiner traveling salesman problem," *Inf. Process. Lett.*, vol. 174, Mar. 2022, Art. no. 106177.

[49] Z. Zhang and J. Yang, "A discrete cuckoo search algorithm for traveling salesman problem and its application in cutting path optimization," *Comput. Ind. Eng.*, vol. 169, Jul. 2022, Art. no. 108157.

[50] S. R. Kanna, K. Sivakumar, and N. Lingaraj, "Development of deer hunting linked earthworm optimization algorithm for solving large scale traveling salesman problem," *Knowl.-Based Syst.*, vol. 227, Sep. 2021, Art. no. 107199.

[51] G. H. Al-Gaphari, R. Al-Amry, and A. S. Al-Nuzaili, "Discrete crow-inspired algorithms for traveling salesman problem," *Eng. Appl. Artif. Intell.*, vol. 97, Jan. 2021, Art. no. 104006.

[52] Y. Hu, Z. Zhang, Y. Yao, X. Huyan, X. Zhou, and W. S. Lee, "A bidirectional graph neural network for traveling salesman problems on arbitrary symmetric graphs," *Eng. Appl. Artif. Intell.*, vol. 97, Jan. 2021, Art. no. 104061.

[53] I. I. Huerta, D. A. Neira, D. A. Ortega, V. Varas, J. Godoy, and R. Asín-Achá, "Improving the state-of-the-art in the traveling salesman problem: An anytime automatic algorithm selection," *Expert Syst. Appl.*, vol. 187, Jan. 2022, Art. no. 115948.

[54] M. Gunduz and M. Aslan, "DJAYA: A discrete Jaya algorithm for solving traveling salesman problem," *Appl. Soft Comput.*, vol. 105, Jul. 2021, Art. no. 107275.

[55] Y. Wang and Z. Han, "Ant colony optimization for traveling salesman problem based on parameters optimization," *Appl. Soft Comput.*, vol. 107, Aug. 2021, Art. no. 107439.

[56] P. Stodola, P. Otrísal, and K. Hasilová, "Adaptive ant colony optimization with node clustering applied to the travelling salesman problem," *Swarm Evol. Comput.*, vol. 70, Apr. 2022, Art. no. 101056.

[57] X. Dong and Y. Cai, "A novel genetic algorithm for large scale colored balanced traveling salesman problem," *Future Gener. Comput. Syst.*, vol. 95, pp. 727–742, Jun. 2019.

[58] B. C. H. Silva, I. F. C. Fernandes, M. C. Goldbarg, and E. F. G. Goldbarg, "Quota travelling salesman problem with passengers, incomplete ride and collection time optimization by ant-based algorithms," *Comput. Oper. Res.*, vol. 120, Aug. 2020, Art. no. 104950.

[59] S. Ebadinezhad, "DEACO: Adopting dynamic evaporation strategy to enhance ACO algorithm for the traveling salesman problem," *Eng. Appl. Artif. Intell.*, vol. 92, Jun. 2020, Art. no. 103649.

[60] M. M. Krishna, N. Panda, and S. K. Majhi, "Solving traveling salesman problem using hybridization of rider optimization and spotted hyena optimization algorithm," *Expert Syst. Appl.*, vol. 183, Nov. 2021, Art. no. 115353.

[61] J. Deckerová, J. Faigl, and V. Krátký, "Traveling salesman problem with neighborhoods on a sphere in reflectance transformation imaging scenarios," *Expert Syst. Appl.*, vol. 198, Jul. 2022, Art. no. 116814.

[62] V. Pandiri and A. Singh, "An artificial bee colony algorithm with variable degree of perturbation for the generalized covering traveling salesman problem," *Appl. Soft Comput.*, vol. 78, pp. 481–495, May 2019.

[63] A. F. Tuani, E. Keedwell, and M. Collett, "Heterogenous adaptive ant colony optimization with 3-opt local search for the travelling salesman problem," *Appl. Soft Comput.*, vol. 97, Dec. 2020, Art. no. 106720.

[64] C. Tang, "An enhanced pathfinder algorithm for engineering optimization problems," *Eng. With Comput.*, vol. 38, pp. 1481–1503, Jan. 2021.

[65] X. Chen. (Mar. 5, 2021). *The TSPLIB Symmetric Traveling Salesman Problem Instances*. [Online]. Available: http://elib.zib.de/pub/mp-test data/tsp/tsplib/tsp/index.html

[66] A. H. Ismail, "Using the bees algorithm to solve combinatorial optimisation problems for TSPLIB," *Proc. IOP Conf. Ser., Mater. Sci. Eng.*, vol. 847, no. 1, 2020, Art. no. 012027.

[67] C. Ravikumar, "Parallel techniques for solving large scale travelling salesperson problems," *Microprocessors Microsyst.*, vol. 16, no. 3, pp. 149–158, Jan. 1992.

[68] E. Babaee Tirkolaee, P. Abbasian, M. Soltani, and S. A. Ghaffarian, "Developing an applied algorithm for multi-trip vehicle routing problem with time windows in urban waste collection: A case study," *Waste Manage. Res., J. Sustain. Circular Economy*, vol. 37, no. 1, pp. 4–13, Jan. 2019.

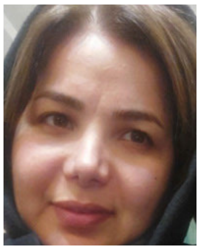
[69] A. Calik, "A novel interactive fuzzy programming approach for optimization of allied closed-loop supply chains," Tech. Rep., 2018.

[70] S. K. Roy, G. Maity, and G.-W. Weber, "Multi-objective two-stage grey transportation problem using utility function with goals," *Central Eur. J. Oper. Res.*, vol. 25, no. 2, pp. 417–439, Jun. 2017.

[71] O. Palanci, "Cooperative grey games: Grey solutions and an optimization algorithm," *Int. J. Supply Oper. Manag.*, vol. 4, no. 3, pp. 202–215, 2017.
[72] E. Savku and G.-W. Weber, "A stochastic maximum principle for a Markov regime-switching jump-diffusion model with delay and an application to finance," *J. Optim. Theory Appl.*, vol. 179, no. 2, pp. 696–721, Nov. 2018.
[73] E. Tirkolaee, A. Hosseinabadi, M. Soltani, A. Sangaiah, and J. Wang, "A hybrid genetic algorithm for multi-trip green capacitated arc routing problem in the scope of urban services," *Sustainability*, vol. 10, no. 5, p. 1366, Apr. 2018.
[74] H. Fan, Y. Zhang, P. Tian, Y. Lv, and H. Fan, "Time-dependent multi-depot green vehicle routing problem with time windows considering temporal-spatial distance," *Comput. Oper. Res.*, vol. 129, May 2021, Art. no. 105211.
[75] A. A. R. Hosseinabadi, J. Vahidi, V. E. Balas, and S. S. Mirkamali, "OVRP_GELS: Solving open vehicle routing problem using the gravitational emulation local search algorithm," *Neural Comput. Appl.*, vol. 29, no. 10, pp. 955–968, May 2018.
[76] A. A. R. Hosseinabadi, N. S. H. Rostami, M. Kardgar, S. Mirkamali, and A. Abraham, "A new efficient approach for solving the capacitated vehicle routing problem using the gravitational emulation local search algorithm," *Appl. Math. Model.*, vol. 49, pp. 663–679, Sep. 2017.
[77] A. A. R. Hosseinabadi, A. Slowik, M. Sadeghilalimi, M. Farokhzad, M. B. Shareh, and A. K. Sangaiah, "An ameliorative hybrid algorithm for solving the capacitated vehicle routing problem," *IEEE Access*, vol. 7, pp. 175454–175465, 2019.
[78] P. Pirozmand, A. A. R. Hosseinabadi, M. Farrokhzad, M. Sadeghilalimi, S. Mirkamali, and A. Slowik, "Multi-objective hybrid genetic algorithm for task scheduling problem in cloud computing," *Neural Comput. Appl.*, vol. 33, no. 19, pp. 13075–13088, Oct. 2021.
[79] P. Pirozmand, A. Javadpour, H. Nazarian, P. Pinto, S. Mirkamali, and F. Ja'fari, "GSAGA: A hybrid algorithm for task scheduling in cloud infrastructure," *J. Supercomput.*, vol. 78, no. 15, pp. 17423–17449, Oct. 2022.
[80] B. Rana, Y. Singh, and H. Singh, "Metaheuristic routing: A taxonomy and energy-efficient framework for Internet of Things," *IEEE Access*, vol. 9, pp. 155673–155698, 2021.
[81] S. H. Shams Lahroudi, F. Mahalleh, and S. Mirkamali, "Multiobjective parallel algorithms for solving biobjective open shop scheduling problem," *Complexity*, vol. 2022, pp. 1–16, Aug. 2022.

PORIA PIROZMAND received the Ph.D. degree from the School of Computer Science, Dalian University of Technology, China, in 2017, and the master's degree in computer science from the University of Mysore, India. He is currently a Faculty Member with the IT Department, Holmes Institute, Sydney, NSW, Australia. His research interests include human mobility in opportunistic networks, social networks, cloud computing, image processing, and real-time systems in body area sensor networks and wireless sensor networks, among other areas of research.

ALI ASGHAR RAHMANI HOSSEINABADI received the M.Sc. degree in software engineering from the Islamic Azad University of Ayatollah Amoli, Iran, in 2016. He is currently pursuing the Ph.D. degree in computer science with the University of Regina, Canada, in 2020. He has been a Research Assistant with the Faculty of Engineering, Tamishan University, since 2013, and a Faculty Member, since 2018. He is the author of more than 200 research papers in several journals and international conferences. His research interests include optimization, computational intelligence, metaheuristic algorithms, vehicle routing problems, wireless networks, cloud computing, and the Internet of Things. He has registered six Iranian patents in the area of computational intelligence and nanotechnology. Besides, he is a member of several editorial boards and a member of several national and international journals and conferences.



MAEDEH JABBARI CHARI received the M.Sc. degree in software engineering from the Islamic Azad University of Masjed Soleiman, Khuzestan, Iran, in 2016. Her research interests include optimization, metaheuristic algorithms, WSN, and cloud computing.

FAEZEH PAHLAVAN received the M.Sc. degree in software engineering from the University of Mazandaran, Iran, in 2023. Her research interests include optimization, computational intelligence, metaheuristic algorithms, wireless sensor networks, cloud and fog computing, and the Internet of Things.

SEYEDSAEID MIRKAMALI received the B.E. degree in software engineering from the Sadjad University of Technology, and the M.Tech. degree in computer cognition and technology and the Ph.D. degree in computer science from the University of Mysore, in 2008 and 2014, respectively. He is currently an Assistant Professor with the Department of Computer Engineering and IT, Payame Noor University. He has authored/coauthored multiple research papers and books in different computer science domains, such as sensor networks, cloud computing, optimization algorithms, image processing, and machine learning. Besides, he served as a member of several editorial boards and a member of several national and international journals and conferences.

GERHARD-WILHELM WEBER received the Diploma degree in mathematics and the Ph.D. degree in economics/business administration from RWTH Aachen University, and the Habilitation degree from TU Darmstadt, Germany. He was a Professor with IAM, Middle East Technical University (METU), Ankara, Turkey, for 15 years. He replaced the Professorships with the University of Cologne, Germany, and TU Chemnitz, Germany. He is currently a Professor with the Faculty of Engineering Management, Poznań University of Technology, Poznań, Poland. His research interests include mathematics, statistics, operational research, data science, machine learning, artificial intelligence, inverse problems, remote sensing, finance, economics, optimization, optimal control, management science, neuroscience, biology, medicine, psychology, development, physics, chemistry, literature and arts, cosmology and spirituality, Christianity, generalized space-time design, research, shift, and travel. He is an IFORS Fellow. He serves in IFORS, specifically EURO where he is an Advisor to EURO Conferences, and as a Section Editor for *IFORS Newsletter*. He is internationally involved in the organization of scientific activities.

SUMMERA NOSHEEN received the Ph.D. degree from the School of Electrical Engineering and Computing, The University of Newcastle, Australia, in 2021. She is currently with the Faculty of Engineering, The University of Sydney, Australia. She received the Commonwealth Department of Education, Science and Training and The University of Newcastle Research Training Program (RTP) tuition fee and stipend scholarships. Her research interests include wireless networks, quality of service, quality of experience, and MAC layer resource allocation.

AJITH ABRAHAM (Senior Member, IEEE) received the B.Tech. degree in electrical and electronic engineering from the University of Calicut in 1990, the M.S. degree from Nanyang Technological University, Singapore, in 1998, and the Ph.D. degree in computer science from Monash University, Melbourne, Australia, in 2001. He is currently the Pro-Vice Chancellor of Bennett University, India. Prior to this, he was the Dean of the Faculty of Computing and Mathematical Sciences, FLAME University, Pune, and the Founding Director of the Machine Intelligence Research Laboratories (MIR Labs), USA, a Not-for-Profit Scientific Network for Innovation and Research Excellence, connecting industry and academia. During the last three years, he also held two University Professorial appointments, including a Professor of Artificial Intelligence with Innopolis University, Russia, and the Yayasan Tun Ismail Mohamed Ali Professorial Chair of Artificial Intelligence, UCSI, Malaysia. He works in a multi-disciplinary environment and has authored/coauthored more than 1,400+ research publications. He has more than 54,000 academic citations (H-index is more than 110 as per Google Scholar). He has given more than 200 plenary lectures and conference tutorials (in more than 20 countries). He was the Chair of the IEEE Systems, Man and Cybernetics Society Technical Committee on Soft Computing (which has over more than 200 members), from 2008 to 2021. He was the Editor-in-Chief of *Engineering Applications of Artificial Intelligence* (EAAI), from 2016 to 2021. He serves/served on the editorial board for over 15 international journals indexed by Thomson ISI. He served as a Distinguished Lecturer for the IEEE Computer Society Representing Europe, from 2011 to 2013.