Hindawi
Computational Intelligence and Neuroscience
Volume 2019, Article ID 9293617, 21 pages
https://doi.org/10.1155/2019/9293617

Review Article

Dragonfly Algorithm and Its Applications in Applied Science Survey

Chnoor M. Rahman [1,2] **and Tarik A. Rashid** [3]

[1]*Technical College of Informatics, Sulaimany Polytechnic University, Sulaimany, Iraq*
[2]*College of Medicals And Appliend Science, Applied Computer Department, Charmo University, Sulaimany, Iraq*
[3]*School of Science and Engineering, Computer Science and Engineering Department, University of Kurdistan Hewler, Erbil, Iraq*

Correspondence should be addressed to Chnoor M. Rahman; chnoor.rahman@charmouniversity.org

Received 14 August 2019; Revised 24 October 2019; Accepted 13 November 2019; Published 6 December 2019

Academic Editor: Maciej Lawrynczuk

One of the most recently developed heuristic optimization algorithms is dragonfly by Mirjalili. Dragonfly algorithm has shown its ability to optimizing different real-world problems. It has three variants. In this work, an overview of the algorithm and its variants is presented. Moreover, the hybridization versions of the algorithm are discussed. Furthermore, the results of the applications that utilized the dragonfly algorithm in applied science are offered in the following area: machine learning, image processing, wireless, and networking. It is then compared with some other metaheuristic algorithms. In addition, the algorithm is tested on the CEC-C06 2019 benchmark functions. The results prove that the algorithm has great exploration ability and its convergence rate is better than the other algorithms in the literature, such as PSO and GA. In general, in this survey, the strong and weak points of the algorithm are discussed. Furthermore, some future works that will help in improving the algorithm's weak points are recommended. This study is conducted with the hope of offering beneficial information about dragonfly algorithm to the researchers who want to study the algorithm.

1. Introduction

Computational intelligence (CI) is one of the newest areas of research. CI is a set of methodologies inspired by nature. CI based techniques are useful to solve complex real-world problems when the traditional methods are ineffective. Fuzzy logic, artificial neural network, and evolutionary computation are part of CI.

Evolutionary computation mainly concerns optimization problems including combinatorial, mixed, or continuous problems. Evolutionary strategies and genetic algorithms are examples of evolutionary computation. However, this field has extended its scope to cover other areas.

Swarm intelligence (SI), for example, is part of the evolutionary computation. Nevertheless, the efficiency of SI-based algorithms has attracted many researchers in various areas, which makes SI become a separate field [1]. Swarm-based algorithms produce low cost, fast, and robust solutions to complex real-world problems [2]. In SI-based techniques a number of agents form a population. Agents in a population interact with each other and their environment. Nature (particularly, biological systems) is a great inspiration for these algorithms [3]. In SI techniques, agents practice simple rules. General control structures do not exist to show how individuals should behave. Interaction between agents causes the disclosure of global intelligent behaviour, which is not known to the agents [4]. Recently many SI-based algorithms have been proposed. Most of them are mimicking the swarm and animal behaviours in nature. The most popular SI algorithms include particle swarm optimization (PSO) proposed by Kennedy and Eberhart [5]. PSO can be counted as a significant improvement in the field. It mimics the behaviours of the school of birds or fish. A particle represents a single solution that has a position in the search space. Furthermore, at the beginning of the 1990s, Marco Dorigo completed his Ph.D. thesis on optimization

and nature-inspired algorithms. In his thesis, he examined a novel idea known as an ant colony optimization (ACO) algorithm [6]. Chu et al. developed the cat swarm optimization (CSO) algorithm based on the behaviour of cats [7]. Grey wolf optimizer (GWO) was introduced by Mirjalili et al. [8]. GWO mimics the hunting behaviour of wolfs. Later, the dragonfly optimization algorithm (DA) was proposed by the same author [9]. DA was mainly inspired by the hunting and migration behaviours of the dragonfly. Another example is the differential evolution (DE) algorithm [10]. DE is a direct search population-based technique, stimulated by the evolution of living species. In [11], artificial bee colony (ABC) was proposed. ABC mimics the behaviours of honeybees. The results of this algorithm proved that it has well-balanced exploitation and exploration ability. Fitness dependent optimizer (FDO) was proposed in [12]. FDO was inspired by bee swarming reproductive process. However, it does not have any algorithmic connection with the artificial bee colony algorithm or the honey bee colony algorithm. Instead, it is based on the PSO. Donkey and Smuggler Optimization (DSO) algorithm was proposed in [13]. DSO mimics the searching behaviours of donkeys. Searching and selecting routes by donkeys were utilized as an inspiration for the algorithm. Firefly algorithm (FA) [14] is another metaheuristic algorithm. It simulates the flashing behaviour of fireflies and the fact of bioluminescent communication. FA is counted as one of the most powerful techniques for solving constrained optimization problems and NP-hard problems.

The importance of the metaheuristic algorithms and the reason that they have been used in many applications has encouraged the researchers to publish survey papers on the algorithms, for example, a systematic and meta-analysis survey of whale optimization algorithm [15]. In [16], a survey on the new generation of metaheuristic algorithms was presented. Another survey on nature-inspired metaheuristic algorithms with its domain specifications was proposed in [17]. In [18], the recent development on the modifications of the cuckoo search (CS) algorithm was proposed. The CS was originally developed by Yang and Deb [19]. It mimics the brood parasitic behaviour of cuckoo species and utilized the Levy flight action of some fruit flies and birds.

One of the most recently swarm-based algorithms is the dragonfly algorithm. It has been successfully utilized in many different applications. The DA is found to produce competitive and efficient results in almost all the applications that utilized it. After publishing the algorithm in 2016 until the end of working on this survey, it has been utilized to optimize a lot of problems in different areas. Thus, this paper is placed to review the dragonfly algorithm as one of the most recent algorithms in the area.

This work first presents an overview of the DA. The variants of the algorithm are then described. Furthermore, the hybridization versions of the algorithm with other algorithms are addressed. Additionally, applications in the applied science fields are discussed. Moreover, a comparison between the DA and some other metaheuristics is made. The advantages and disadvantages of DA are then discussed. The DA is also tested on the CEC-C06 2019 benchmark functions. Furthermore, the PSO, DE, and FA are tested on the traditional benchmark functions and the results are shown and compared with the results of the DA. In addition, a discussion and some problems of DA are presented along with providing solutions and future works to make the algorithm work better. Finally, a conclusion is given.

2. Overview of DA

DA is mimicking the swarming behaviours of a dragonfly. The reason for their swarming is either migration or hunting (dynamic swarm or static swarm, respectively). In a static swarm, small groups of dragonflies move over a small area to hunt other insects. Behaviours of this type of swarming include local movements and abrupt changes. In dynamic swarming, however, a massive number of dragonflies create a single group and move towards one direction for a long distance [20]. The aforementioned swarming behaviours are counted as the main inspiration of DA. Static and dynamic swarming behaviours are, respectively, in line with the exploration and exploitation phases of the metaheuristic optimization algorithm. Figure 1 shows the behaviours of dragonflies in static and dynamic swarming. To direct artificial dragonflies to various paths, five weights were used, which are separation weight (s), alignment weight (a), cohesion weight (c), food factor (f), enemy factor (e), and the inertia weight (w). To explore the search space high alignment and low-cohesion weights are used, however, to exploit the search space low alignment and high-cohesion weights can be used. Furthermore, to transfer between exploration and exploitation, the radii of neighbourhood enlarged proportionally to the number of iterations were used. Tuning the swarming weights (s, a, c, f, e, and w) adaptively during the optimization process is another way to balance exploration and exploitation. Mathematically, each of the aforementioned weight factors are shown in equations (1)–(5).

The separation can be calculated as mentioned by Reynolds [21]:

$$S_i = -\sum_{j=1}^{N} X - X_j. \tag{1}$$

In equation (1), X indicates the position for the current individual, X_j is the position for the j^{th} neighbouring dragonfly, N is the number of individual neighbours of the dragonfly swarm, and S indicates the separation motion for the i^{th} individual.

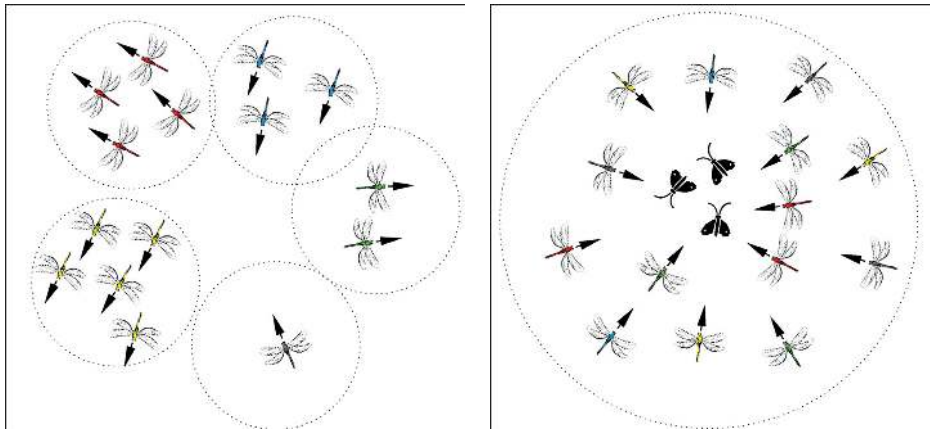
Equation (2) was used for calculating the alignment [9]:

$$A_i = \frac{\sum_{j=1}^{N} V_j}{N}, \tag{2}$$

where A_i is the alignment motion for i^{th} individual and V is for the velocity of the j^{th} neighbouring dragonfly.

Cohesion was expressed as follows:

$$C_i = \frac{\sum_{j=1}^{N} X_j}{N} - X, \tag{3}$$



(a) (b)

Figure 1: Dynamic dragonfly swarming (a) versus static swarming (b) [9].

where C_i is the cohesion for i^{th} individual, N is the neighbourhood size, X_j is the position of the j^{th} neighbouring dragonfly, and X is the current dragonfly individual.

Attraction motion towards food is computed as follows:

$$F_i = X^+ - X, \tag{4}$$

where F_i is the attraction of food for i^{th} dragonfly, X^+ is the position of the source of food, and X is the position of the current dragonfly individual. Here, the food is the dragonfly that has the best objective function so far.

Distraction outwards predators are calculated as follows:

$$E_i = X^- + X, \tag{5}$$

where E_i is the enemy's distraction motion for the i^{th} individual, X^- is the enemy's position, and X is the position of the current dragonfly individual.

For position updating in the search space, artificial dragonflies use two vectors: step vector ΔX and position vector X. The step vector is an analogy to the velocity vector in the PSO algorithm [5]. The position updating is also based mainly on the PSO algorithm framework. The step vector is defined in [9] as follows:

$$\Delta X_{t+1} = \left(sS_i + aA_i + cC_i + fF_i + eE_i\right) + w\Delta X_t, \tag{6}$$

where S_i represents the separation for the i^{th} dragonfly; A_i is the alignment for i^{th} dragonfly; C_i represents the cohesion for i^{th} dragonfly; F_i represents the food source for the i^{th} individual; E_i represents the position of the enemy for i^{th} dragonfly; w is the inertia weight; and t indicates the iteration counter.

When the step vector calculation is finished, the calculation for the position vectors start as follows:

$$X_{t+1} = X_t + \Delta X_{t+1}, \tag{7}$$

where t indicates the current iteration.

In order to raise the probability of exploring the whole decision space by an optimization algorithm, a random move needs to be added to the searching technique. When no neighbouring solutions are there, to increase randomness, stochastic behaviour, and exploration of artificial dragonfly individuals, dragonflies are required to use a random walk (Lévy flight) to fly throughout the search space.

3. Convergence and Divergence of DA

For the transition from intensification to diversification, dragonflies should adaptively change their weights. This guarantees the convergence of dragonfly individuals during the optimization process. As the optimization process progresses, to adjust the flying path, the neighbourhood area is expanded; hence, at the final stage of optimization, the swarm becomes one group to converge to a global optimum. The best and the worst solutions found so far become the food source and enemy, respectively. This makes convergence and divergence towards the promising area and outwards nonpromising area of the search space, respectively.

4. Variants of DA

DA has three variants.

4.1. DA for Single-Objective Problems. In DA, at the beginning of the optimization process, randomly a set of solutions is created. Initially, the step and position vectors of artificial dragonflies are assigned to stochastic values between lower and upper bounds. The position and step vectors for each dragonfly should be updated in each iteration using Equations (7) or (8) and (6). To update the step vector and position vector of dragonflies, their neighbourhood is chosen by Euclidean distance calculation. The position updating is continued until meeting the end criterion. Visual 1 shows the pseudocode for DA for single objective problems.

The single DA is the most popular variant among the other variants of DA.

4.2. DA for Binary Problems. In binary search space, the position vector can take 0 or 1. Hence, the position of search agents cannot be updated by adding a step vector to the

Initialize the dragonflies population Xi ($i = 1, 2, \ldots, n$)
Initialize step vectors Xi ($i = 1, 2, \ldots, n$)
while the end condition is not satisfied
  Calculate the objective values of all dragonflies
  Update the food source and enemy
  Update w, s, a, c, f, and e
  Calculate S, A, C, F, and E using equations (1)–(5)
  Update neighbouring radius
  If a dragonfly has at least one neighbouring dragonfly
    Update velocity vector using equation (6)
    Update position vector using equation (7)
  Else
    Update the position vector using Lévy flight
  End if
    Check and correct the new positions based on the boundaries of variables
End while

Visual 1: Pseudocode for DA [9].

position vector. Using the transfer function is the easiest way to convert continuous SI technique to binary algorithm [22]. Transfer function takes velocity (step) values as input and returns a number between 0 and 1 as output, which indicates the probability of changing the individual's position. Similar to continuous optimization, the function simulates sudden changes in particles with large velocity. To use the DA for binary problems (BDA), Equation (8) was used [22]:

$$T(\Delta X) = \left| \frac{\Delta X}{\sqrt{\Delta X^2 + 1}} \right|. \tag{8}$$

Equation (8) was used to calculate the changing probability of the position of all artificial dragonflies. Equation (9) for updating position was then created to update the search agent's position in binary search spaces:

$$X_{t+1} = \begin{cases} X_t, & r < T(\Delta X_{t+1}), \\ X_t, & r \geq T(\Delta X_{t+1}), \end{cases} \tag{9}$$

where r is a number in [0, 1].

In BDA, it was assumed that all of the artificial dragonflies are in one swarm. Therefore, BDA simulates exploration and exploitation by adaptively tuning the swarming factors (s, a, c, f, and e) and the inertia weight (w). Visual 2 presents the pseudocode for BDA.

4.3. The DA for Multiobjective Problems. Multiobjective problems have multiple objectives. The result for multiobjective problems is a set called Pareto optimal set. The set contains the best trade-offs between the objectives [23].

In order to use the DA to deal with multiobjective problems (MODA), an archive was first provided to save and retrieve the best Pareto optimal solutions during the process of optimization. To update the position, the food source is selected from the archive, and the rest of the process of position updating is identical to that of DA.

Similar to the multiobjective particle swarm optimization (MOPSO) algorithm [24], to observe the well-spread Pareto optimal front, the food source is chosen from the least populated region of the produced Pareto optimal front. In MODA, this was carried out through finding the worst and the best objectives of the current Pareto optimal solutions. Furthermore, a hypersphere to cover all the solutions was defined, and then in each iteration, the hyperspheres are divided into equal sub-hyperspheres. When the segments are created a roulette-wheel mechanism with the following probability for every segment was used for the selection process [25]:

$$P_i = \frac{c}{N_i}, \tag{10}$$

where c is a constant number and greater than one and $N_i$is the number of Pareto optimal solutions obtained in the i^{th} segment. Equation (10) gives the MODA a higher probability to select the food source from the less populated segments.

On the other hand, to select predators from the archive, the worst (most populated) hypersphere was chosen, so that the artificial dragonflies are prevented from searching around nonpromising areas. For the selection process the roulette-wheel mechanism with the following probability was used:

$$P_i = \frac{N_i}{c}, \tag{11}$$

where c is a constant number and greater than one and $N_i$is the number of Pareto optimal solutions obtained in the i^{th} segment.

In Equation (11), using the roulette-wheel mechanism, the most crowded hyperspheres have a higher probability of being selected as enemies. To prevent the archive from becoming full, if at least one of the archive residences dominates the solution, then it should not be allowed to enter the archive. However, the Pareto optimal solutions dominated by the solution should be removed from the archive and the solution should be added to the archive. If the archive is full, then one or more solutions may be removed from the most populated segments [25]. In addition

Initialize the dragonflies population Xi ($i = 1, 2, \ldots, n$)
Initialize step vectors Xi ($i = 1, 2, \ldots, n$)
while the end condition is not satisfied
  Calculate the objective values of all
  Dragonflies Update the food source and
  enemy Update w, s, a, c, f, and e
  Calculate S, A, C, F, and E using equations (1)–(5)
  Update step vectors using equation (6)
  Calculate the probabilities using equation (8) Update
  position vectors using equation (9)
End while

Visual 2: Pseudocode for BDA [9].

to the parameters of DA, MODA has two new parameters: one for defining the maximum number of hyperspheres and another parameter to specify the archive size. The pseudocode for MODA is presented in Visual 3.

5. Hybridization Versions of DA

In the metaheuristic context, hybridization refers to merge the powerful characteristics of two or more algorithms in order to provide a new powerful algorithm based on the features of the merged ones [26]. In the following sections, the hybridization versions of DA are discussed.

Ranjini and Murugan [26] combined some features of PSO with DA and produced a new algorithm called memory-based hybrid dragonfly algorithm (MHDA). In MHDA, two more features are added to the DA to refine its performance. (1) Internal memory is added to observe the possible solution. This internal memory has a great role in converging to global optima. When the internal memory is added, each dragonfly individual will be able to keep track of its correlates in the problem space, which are related to the value of fitness. In the PSO algorithm, this is named as *p*best. The best fitness value in each iteration is compared to the search agent's fitness value of the current population. As a result, the DA-*p*best is created from the better-saved solutions. The dragonfly individuals are also able to keep track of the best value founded so far by any dragonfly in the neighbourhood. This is similar to the concept of *g*best in the PSO. Here, DA-*g*best is used to store the best value. The capability of exploitation in DA is enhanced by these two novel concepts: *p*best and *g*best. The internal memory feature gives a greater performance compared with the conventional algorithm and gives the power to escape from local optima [27].

(2) Iterative level hybridization with PSO runs on the saved solution's set. To enhance the performance of optimization in the iteration level hybridization approach, two algorithms are executed iteratively in sequence [28]. Here to extend the search space and converge to a more promising area, DA with internal memory is used, and then the previously limited area is exploited using PSO to find better solutions.

Thus, to reach global optimal solutions in the MHDA, the DA's exploration features in the initial stage and PSO's exploitation features in the final stage were combined. Visual 4 shows the pseudocode of MHDA. MHDA's superior performance on unimodal functions showed speed converge and an accurate diversification of the algorithm. The results of the algorithm showed the competitive performance of the algorithm and that it can be used to optimize hard problems.

Hence, compared with the original DA, the hybrid algorithm-MHDA showed better performance because the MHDA provided good stability between exploration and exploitation capabilities offered by DA and PSO, respectively. Then, the *p*best and *g*best of PSO were initialized using DA-*p*best and DA-*g*best matrixes, respectively.

The equations for position and velocity of PSO were modified as follows:

$$V_{k+1}^{i} = wV_{k}^{i} + C_1 r_1\left(\mathrm{DA} - p\mathrm{best}_{k}^{i} - X_{k}^{i}\right) + C_2 r_2\left(\mathrm{DA} - g\mathrm{best}_{k}^{g} - X_{k}^{i}\right), \tag{12}$$

$$X_{k+1}^{i} = X_{k}^{i} + V_{k+1}^{i}, \tag{13}$$

where $\mathrm{DA} - p\mathrm{best}_{k}^{i}$ and $\mathrm{DA} - g\mathrm{best}_{k}^{g}$ are the *p*best and *g*best in PSO, respectively, and k is the size of the swarm.

In [29], DA was combined with an extreme learning machine (ELM) to overcome the problems in gradient-based algorithms. In this technique to optimally select the biases of the hidden layer, DA was used. Using DA, the overall performance of ELM was improved. The convergence of DA-ELM was expected in a small number of iterations. Moreover, the overfitting problems in traditional ELM were overcome using the DA-ELM model. The results showed that in general DA-ELM could outperform both GA-ELM and PSO-ELM. It was also examined that DA has a good ability in searching the feature space adaptively and showed its capability in avoiding local minima that may cause premature convergence. Furthermore, it was proved that the DA has the minimum root mean square error that showed the ability of DA in finding optimal feature combination in less prediction error. To some extent, the average computational time of the DA was also compatible with PSO and GA. The compared and proposed models trained using a thousand iterations.

For the optimization process, RMSE was examined as a fitness function and the number of iterations was used as a criterion to stop the process. The ranges of Reference

```
Initialize the dragonflies population Xi (i = 1, 2, ..., n)
Initialize step vectors Xi (i = 1, 2, ..., n)
Define the maximum number of hyperspheres (segments). Define the archive size
while the end condition is not satisfied
   Calculate the objective values of all dragonflies
   Find the nondominated solutions
   Update the archive with respect to the obtained nondominated solutions
   If the archive is full
      Run the archive maintenance mechanism to omit
      one of the current archive members Add the new
      the solution to the archive
   End if
   If any of the new added solutions to the archive is
      located outside the hyperspheres
      Update and reposition all of the hyperspheres to cover the new solution(s)
   End if
   Select a food source from the archive: =
   SelectFood (archive)
   Select an enemy from archive: = SelectEnemy (archive)
   Update step vectors using equation (8)
   Update position vectors using equation (9)
   Check and correct the new positions based on the
   boundaries of variables
End while
```

Visual 3: Pseudocode for MODA [9].

[30] proposed a hybrid version of the dragonfly algorithm with support vector regression (SVR) for online voltage stability assessment. Parameter selection in SVR highly affects its performance. The important parameters for SVR include penalty parameters C, nonsensitivity coefficient ε, and the kernel parameters. The DA was used in parameter settings of SVR, which improved the performance of the technique. For training the examined model (DFO-SVR) as an input, the voltage magnitude produced from PMU buses were utilized for various operating conditions and the least values of voltage stability index (VSI) were used as output variables. Three statistical indices were utilized for evaluating the DFO-SVR model. Those statistical indices were correlation coefficient (R), root mean square error (RMSE), and the percentage of mean square error (PRMSE).

Parameters (C, γ, and ε) were [1 1000], [0.0001 0.1], and [0.1 1], respectively. The optimal solution values of the C, ε, and γ parameters for this work are shown in Table 1.

The produced results proved that the proposed model has a good performance for prediction.

Depending on reference [31] determining an adequate cluster radius is required for generating fuzzy rules. In this work, the radius of the cluster was between 0.2 and 0.5. The predicted outputs for the systems IEEE 30-bus and Algerian 59-bus systems were compared with the actual ones using DFO-SVR and ANFIS techniques, respectively. The produced results proved that the performance of prediction for both systems was better in the DFO-SVR technique compared with the ANFIS technique.

In reference [32], a hybrid version of the binary dragonfly algorithm with an enhanced particle swarm optimization algorithm for solving the feature selection problem was examined. The proposed approach called Hybrid Binary Dragonfly Enhanced Particle Swarm Optimization Algorithm (HBDESPO). The ability of DA to secure diverse solutions and the enhanced PSO ability to converge to the best global solution produced a hybrid algorithm with better performance. In the examined hybrid technique, dual exploration was used and excessive exploitation was avoided. In the HBDESPO technique, the velocities of participated algorithms updated independently. In this examined system, the K-nearest neighbour (KNN) was used as a classifier for ensuring the robustness of the training data and reach better feature combinations. The proposed algorithm was tested on 20 standard datasets from the UCI repository. The datasets were divided into three sets: testing, validation, and training. The value of K in the KNN was assigned to 5 based on trial and error.

The training set was used for evaluating the KNN on the validation set by using the proposed technique to advise the feature selection process. For the final evaluation of the best nominated feature, the training set was used. The minimization problem for this work is shown in equation (14). The setting of the optimizer and global specific parameters are shown in Table 2.

$$\text{Fitness} = \alpha \text{ER}(D) + \beta \frac{|R|}{|C|}, \tag{14}$$

where ER(D) is the classifier's error rate, R represents the selected feature's length, and C shows the total number of features. β and α are constants for controlling the weights of classification accuracy of the minimization feature.

```
Initializing the set of parameters:
Maximum iteration (Max-iter), maximum number of search agents (N max) number of search agents (N), number of dimensions (d),
upper bound and lower bound of variables. Initialize the dragonflies' populations (X). Initialize the step vectors (ΔX)
while maximum iterations not done
  For each dragonfly calculate fitness value
    If Fitness Value < DA-pbest
      In this iteration move the current value to DA-pbest matrix
    End if
    if fitness value < DA-gbest
      set current value as DA-gbest
    End if
  End
For each dragonfly
  Update the food source and enemy
  Update w, s, a, c, f, and e
  Calculate S, A, C, F, and E using Equtaions (2)–(6)
  Update neighbouring radius
  If a dragonfly has at least one neighbouring dragonfly
    Update velocity vector using equation (8) Update
    position vector using equation (9)
  Else
    Update position vector using equation (10)
  End if
  Check and correct new positions based on boundaries of variables
End
------------------End of DA and Start of PSO------------------
For each particle
  Initialize particle with DA-pbest matrix Set PSO-gbestas DA-gbest
End
while maximum iterations or minimum error criteria is not attained
  For each particle
    Calculate fitness value
    If fitness value < PSO-pbest in history
      set current value as the new PSO-pbest
    End if
  End
  Choose the particle with the best fitness value of all the
  particles as the PSO-gbest
  For each particle
    Calculate particle velocity according to equation (12)
    Update particle position according to equation (13)
  End
End while
```

Visual 4: Pseudocode for MHDA [26].

Table 1: The optimal parameters for the SVR model found by using DA [30].

SVR parameters	Optimal values of SVR parameters	
	IEEE 30-bus	Algerian 59-bus
C	971.9378	985.561
γ	0.1	0.1
ε	0.0001	0.0001

The proposed technique was compared with the results of binary DA from [9] and the enhanced PSO from [22]. From this research work, it was concluded that the examined algorithm could provide high classification accuracy while keeping the ratio of feature selection to the minimum. Moreover, small fitness values were reached and across various runs the algorithm kept its stability. It was also concluded that the values of the standard deviation observed the robustness of the algorithm as it repeatedly could converge to a similar solution.

In DA, having an excessive number of social interactions may reduce the accuracy of the solution, fall easily into local optima, and cause an imbalance between exploitation and exploration. To control these deficiencies, in reference [33], DA was merged with an improved version of the Nelder–Mead algorithm (INMDA). The reason for this hybridization was to make the capability of local explorative stronger and prevent falling into local optima. INMDA consists of two stages. First, to search the solution space, DA was utilized

Table 2: Parameter settings for binary hybrid HBDESPO [32].

Parameter	Value
No. of iterations	70
No. of search agents	5
Dimension	No. of features in the data
Search domain	[0 1]
No. of runs	10
w_{max}	0.9
w_{min}	0.4
Deltax_{max}	6
c_1	2
c_2	2
v_{max}	6
β in fitness function	0.01
α in fitness function	0.99

and gave the required diversity to the individuals to find the global optimum solution. Second, the improved version of the Nelder–Mead (INM) simplex method was utilized to find the best and worst points and calculate the population centroid. One of the main features of INM is that the population's centroid was used to update the position. This improves the possibilities of jumping out of local optima. The efficiency of the proposed technique was tested using 19 unconstraint and 13 large-scale benchmark functions with dimensions of 30, 500, and 1000, respectively. Table 3 shows the parameter settings for INMDA. In Table 3, t represents the current iteration and T is the number of iterations.

For single-objective functions, the proposed technique was compared to other algorithms, such as Memory-based Hybrid Dragonfly Algorithm (MHDA), DA, PSO, and recently SI-based optimization algorithms, such as ALO and WOA. For each optimization algorithm, 30 independent runs were used. The number of agents and the maximum number of iterations were 30 and 1000, respectively, following the literature [26]. Fried man's test functions and Wilcoxon rank sum were used to statistically test the significance of the experimental results for the 19 unconstrained benchmark functions. The results showed the great performance of the proposed work for solving high-dimensional problems compared with the other algorithms, such as DA and MHDA, while they cannot be used to solve high-dimensional problems because they easily encounter "dimensional curse." The work concluded that the INMDA owns a superior performance comparing to the other algorithms. The enhanced exploitation and exploration capabilities from using reverse learning techniques generated this great performance.

Furthermore, for enhancing the performance of optimization by DA, reference [34] examined an improved version of DA. The proposed DA is based on exponential function adaptive steps and elite opposition-based learning strategy. The elite individual was presented to construct the opposite solutions using elite opposition-based learning. The scope of the search area expanded by using the mentioned technique, and it was useful for improving the capability of the global exploration of DA. Furthermore, to replace the original stochastic step, an adaptive step with the exponential step was designed. The improved work was named as

Table 3: Parameter settings for INMDA [33].

$\alpha = 2.\text{rand}.(0.1–0.2t/T)$
$\gamma = 0.1.(1 - 0.2t/T)$
$c = 0.2.\text{rand}.(1 - 2t/T)$
$f = 2.\text{rand}$
$\delta = 0.68$
$\beta = 0.9 - 0.5t/T$
$a = 2.\text{rand}.(0.1 - 0.2t/T)$
$e = 0.1.(1 - 2t/T)$
$w = 0.9 - 0.5t/T$
$\lambda a = 0.9–0.5t/T$

dragonfly based on elite opposition-based learning and exponential function adaptive steps (EOEDA). The reason behind this modification was that DA sometimes has problems to solve complex optimization problems and it easily falls into local optimum, and the speed of convergence was low. The results proved that the EOEDA had better convergence accuracy and the speed of convergence was faster.

In reference [35], different features were selected using a new chaotic dragonfly algorithm (CDA). In CDA, searching iterations of DA were merged into the chaotic maps. To modify the main parameters for movement in DA, ten chaotic maps were utilized to improve the convergence rate and efficiency of the DA. The proposed method was used to select features in the extracted dataset from the drug bank database, which has 6712 drugs. In this work, 553 biotransformed drugs were used. The proposed method was utilized to assess the toxicity of hepatic drugs. The proposed model, in general, consisted of three phases: data preprocessing, feature selection, and classification phase. In phase two, CDA was utilized to pick the features. The k-NN classifier was used to measure the goodness of the selected features. Table 4 presents the initial parameter settings for CDA.

Furthermore, the examined technique was evaluated using three different experiments. The results proved that using chaotic maps with the dragonfly algorithm produced better results. Another experiment was conducted by comparing the performance of CDA with Gauss's chaotic map with seven methods for optimization, namely, PSO, ABC, GWO, CSA, SCA, SSA, and CSO. Table 5 shows the parameters for the participated algorithms. The results proved that CDA provided a better score in most of the cases. However, CDA in most cases provided minimum stability. On the other hand, examining the p value proved that statistically the produced results were important, which shows the superiority of CDA comparing to the aforementioned algorithms. To examine the selected features SVM classifier with various kernel methods was used. The results showed the superiority of the CDA compared with the other techniques. Moreover, in terms of computational time, it was noted that the computational time reduced while using the feature selection algorithm and that the CDA minimized the time remarkably compared with the original DA.

Khadanga et al. [36] proposed a method to control frequency in an islanded AC microgrid (MG). MG is formed

Table 4: Parameter settings for CDA, D stands for dimension [35].

Parameter	Value
β	1.5
D	31
M	50
Lower bound	1
Upper bound	31
Maximum iteration	50

Table 5: Parameter settings for PSO, ABC, CSO, GWO, CSA, and SCA optimization algorithms [35].

Algorithm	Parameters	Value
PSO	An inertia weight	1
	An inertia weight damping ratio	0.9
	Personal learning coefficient	1.5
	Global learning coefficient	2.0
ABC	Number of colony size	10
	Number of food source	5
	Number of limit trials	5
CSO	Number of chicken updated	10
	The percent of roosters population size	0.15
	The percent of hens population size	0.7
	The percent of mother hens population size	0.05
GWO	a	2
SCA	b	2
CSA	Awareness probability	0.1
	Flight length	0.01

by integrating various sources, such as wind power generation, renewable sources of energy, and solar energy generation. In this work, ABC was merged with DA. The idea behind hybrid ABC/DA (HAD) was to merge the exploration and exploitation ability of the DA with the exploration ability of ABC. The proposed technique consists of three components: the dynamic and static swarming behaviour in the DA and the two phases of the global search in ABC. The global search was accomplished by the first component (DA phase), local search was accomplished by the second component (onlooker phase), and the third one accomplished global search (modified scout bee phase).

All the parameters from the original DA and ABC were adapted to the merged technique with one more parameter, which is Prob. Prob parameter was used to balance the dragonfly bee phase and the onlooker bee phase. It was also utilized to make a balance between exploration and exploitation. The prob parameter was set to 0.1 in the carried experiments in the work, which was based on another confirmed experiment. HAD considered D-dimensional solutions and population size of N. Observing the worst case, the time complexity of the iterative process of the hybridized technique was analyzed as follows: in the first phase, the main operation for creating an initial population and the complexity time was O (ND). In the second phase, the stopping criteria were judged and the time complexity was O (1). In the third phase, the value of the rand parameter was judged. If rand is smaller than prob, then perform dragonfly bee phase, else perform onlooker bee phase, then perform a modified scout bee phase, and the time complexity was O (N). In the fourth phase, the solution was updated and the time complexity was O (N). In the fifth phase, continue with the iterations and go back to the second step. Hence, the time complexity of the examined technique was O (ND). 50 iterations were used to calculate the performance of the hybrid technique.

In terms of convergence speed, the results proved that comparing to the original DA and ABC the proposed technique provided better performance and in some cases the results were comparative. In another contribution, the examined technique was used to train the multilayer perceptron (MLP) neural network. The results showed that in terms of convergence speed, achieving the best optimal value, accuracy, and avoiding local minima, the proposed hybrid technique was a better trainer for MLPs in comparison to the original version of the participated algorithms.

6. Applications of DA

Due to the power of DA, enormous research applications in applied sciences have been conducted. For example, machine learning, image processing, wireless, and network applications, and some other areas. In this section, we present applications of the dragonfly algorithm in the aforementioned areas. The purpose of using the DA in the applications in various areas and the results are shown in Table 6.

6.1. Image Processing. In [37], two key crucial factors for traditional ship classification, classifier design and feature selection, were joined together and a novel ship classification model called BDA-KELM classification for high-resolution synthetic aperture radar (SAR) images was proposed. Dragonfly algorithm in a binary search space in this work was used as a searching technique. For the fitness function, the accuracy of the prediction of the subsequent classifier was used. Wrapper-based methods needed too much computational time; in order to overcome this drawback, the kernel extreme learning machine was operated as the elementary classifier and the DA helped in searching for the optimal parameter sets (kernel parameter and penalty factor) for KELM and the optimal feature subset among the feature candidates simultaneously. Integrating both selecting features and classifier design on the base of DA made BDA-KELM simple. The experiment was conducted based on Terra SAR-X SAR imagery. For training, 60% of the samples were used and the rest were used as testing datasets. The same datasets were used in ship classification using the most popular models for classification (KNN, Bayes, Back Propagation neural network (BP neural network), and Support Vector Machine (SVM)). A number of experiments were conducted using different classification models. For each model, the evaluation metrics were computed to prove the superiority of BDA-KELM. Each experiment was run 10 times. To evaluate the proposed model several evaluation metrics were used. They include recall, precision, and F_1-score. The classification results proved the accuracy of the proposed model, which was 97%. Thus, the performance of

Table 6: The purposes of using the DA in various applications and its results.

Reference	Purpose	Result
[37]	BDA helped in searching for the optimal parameter sets (kernel parameter and penalty factor) for KELM and the optimal feature subset among the feature candidates simultaneously	BDA showed its superiority as a searching technique to find the set of optimal parameters and the optimal feature subset
[40]	Multilevel segmentation of colour fundus images	Using the DA as an optimization algorithm produced better results for segmenting colour images
[41]	In a watermarking technique for the medical images, DA was utilized to select the effective pixels	The correlation coefficient values using the DA were greater than the other techniques such as PSO, GA, and random selection
[42]	Exploring the pixels of images and discovering which pixel contains significant information about the object (DA was used as a detection model)	The DA could work as an efficient and fast object extraction from images
[43]	DA was used as a parameter optimizer of SVM; furthermore, the effect of the number of solutions and generations on the accuracy of the produced result and computation time was investigated	It was shown that the classification error rate for the proposed work was lower than that in PSO + SVM and GA + SVM, and the reason for this was that the DA parameters could be altered iteratively; furthermore, it was shown that increasing either the number of solutions or generations decreased the rate of misclassification and rose the computational time
[47]	New updating mechanism and elitism were added to the binary dragonfly algorithm; the improved technique was then used to classify different signal types of infant cry, and it was used to overcome the dimensionality problem and select the most salient features	It was noted that the improved technique reduced the percentage of error rate compared with the original binary dragonfly algorithm
[48]	The DA-based artificial neural network technique was utilized for predicting the primary fuel demand in India	The proposed model using the DA was provided with more accurate results comparing to the existing regression models
[49]	Binary-BDA, multi-BDA, and ensemble learning-based BDA were used for wavelength selection	Using binary-BDA causes instability; however, stability boosted by using the multi-BDA and the ensemble learning-based BDA; in addition, the computational complexity of ensemble learning-based BDA was lower than the multi-BDA
[50]	Instead of gradient-based techniques, DA was used for designing filters of IIR	Using the DA prevented trapping into local optima and coefficients close to the actual value were evaluated, and the minimum mean square value was found; in addition, the superiority of the DA was proved to compare to the PSO, CSO, and BA for the aforementioned problem
[51]	Dragonfly-based clustering algorithm was used to focus on the scalability of the internet of vehicles	The proposed technique was compared to a comprehensive learning PSO and ant colony optimization algorithm; the results proved that in high density and medium density the examined technique showed better and average performance, respectively; however, in a low density, the proposed technique performance was bad while the comprehensive learning PSO performed well
[52]	Dragonfly algorithm was utilized to predict the location of randomly deployed nodes in a designated area; also it was used to localizing different noise percentages of distance measurement (Pn)	For range-based localization with varying Pn, dragonflies could produce fewer errors compared with PSO; furthermore, increasing Pn caused an increase in the distances between real and approximated nodes by DA and PSO
[53]	DA was used to enlarge the lifetime of the RFID network	The cluster breakage was reduced through choosing the cluster heads that had similar mobility but high leftover energy; this reduction reduced energy consuming; hence, compared with the existing techniques the efficiency was improved

Table 6: Continued.

Reference	Purpose	Result
[54]	DA with two selection probabilities were used as new loud balancing technique called (FDLA); the new technique was then used to keep the stability of processing multiple tasks in the cloud environment	The proposed technique provided the minimum load by allocating less number of tasks
[57]	DA was utilized to examine the optimal sizing and location of distributed generation in radial distribution systems to reduce the power loss in the network	Compared with the DA and WOA, MFO performed better and converged earlier
[59]	In the court case assignment problem, the ability of the judicial system highly depends on time and the efficiency of operation in the court case; the DA was used to find the optimal solution of the assignment problem	The DA could show superior results compared with the FA
[60]	DA was used to optimize the optimum sitting of the capacitor in different radial distribution systems (RDSs); the main aim of this study was to minimize power loss and total cost with voltage profile enhancement	The results proved that DA-based optimization provided comparative results with GWO- and MFO-based optimization methods in terms of a small number of iterations and convergence time; however, it provided superior results compared with the PSO-based technique

the examined technique for classification was better than the examined techniques in the literature.

The thresholding of histograms is a technique that is widely used for segmenting grey scale images. However, for colour images, it is not trivial because of its multilevel structure [38, 39]. For this reason, in [40], the authors tried to overcome this problem by using a dragonfly optimization algorithm for doing multilevel segmentation (SADFO) of colour fundus image. The problem of multilevel segmentation was shown as an optimization problem and DA was used to solve it. The threshold values were optimized for the chromatic channels of colour fundus images by exploring the solution space effectively and finding the global best solution. Kapur's entropy was used in the proposed technique. The result proved that the proposed method produces much better results compared with segmentation after changing the image to grey scale.

In the medical images, watermarking is a hot topic that gives security to the capsulated secret code to the images. Hemamalini and Nagarajan [41] examined a powerful watermarking technique that depends on the weight of the pixels. To determine effective pixels for watermarking, discrete wavelet transform (DWT) was utilized for extracting the low- and high-frequency bands. DA used to select the effective pixels which followed the objective function based on edge level, neighbourhood strength, gradient energy, and wavelet energy (ENeGW) of the pixels. Medical retinal images were used for the experiment, and patient data were used as a watermark. In terms of performance metrics, a comparative analysis was carried out in this work. The metrics used were correlation coefficient and peak signal-to-noise ratio (PSNR). In this work, it was observed that the correlation coefficient values for the examined technique compared with the other techniques, such as random selection, PSO, and GA, were greater. The proposed work obtained the correlation at a rate of 0.936719 for the random noise, 0.974479 for salt and pepper noise, and 0.983073 for the rotational noise. Similarly, the PSNR results for the examined dragonfly technique under the existence of the random noise, salt and pepper noise, and rational noise were greater and they were 62.39155 dB, 62.95912 dB, and 63.02815 dB, respectively. However, for the already existing method, such as random selection, the values of PSNR with respect to the noises were 59.59593 dB, 61.48404 dB, and 59.87195 dB, the PSNR values of PSO were 60.20927 dB, 61.63731 dB, and 60.53219 dB, respectively, and the PSNR values of the genetic algorithm were 59.62668 dB, 61.46258 dB, and 59.90074 dB, respectively.

Połap and Woźniak [42] used dragonfly to explore the pixels in images and assess which of these pixels represent significant components of the objects. Therefore, this technique works as a detection model for finding interesting features. In this work, a fitness function was modelled to work as a detection tool to select pixels related to the shapes of objects. In each iteration, the individuals in a given population were assessed for adaptation to the environment. In the case of images, unfortunately, the essential search areas may vary in many places. Hence, a set of tree functions was examined. Results of the examined bioinspired extraction technique showed that utilizing different components of the fitness function resulted in a different selection of key points. This was because different aspects of the image were focused on the different components. From this work, it was concluded that the proposed technique helped on efficient and fast object extraction from images.

6.2. Machine Learning. Parameters in the SVM such as the kernel and penalty parameters have a great impact on the accuracy and complexity of the classification model. In order to decrease classification errors in [43], the dragonfly optimization algorithm was used for parameter optimization of the SVM. The values of kernel and penalty parameters were sent by the DA for training the SVM using the training data. The bounds for searching range of penalty parameter of the SVM was $C_{\min} = 0.01$ and $C_{\max} = 35000$, and the bounds of

the searching range of σ was $\sigma_{\min} = 0.01$ and $\sigma_{\max} = 100$ [44]. In this work, the effects of the number of solutions on the computational time and testing error rate were investigated.

As shown in Figures 2(a) and 2(b), it was concluded that increasing the number of solutions decreases the rate of misclassification. However, computational time was raised. In addition to the number of dragonflies, it was also proved that the number of generations also had effects on the testing error rate and computational time. Increasing the number of generation reduces the error rate to an extent after that enlarging the number of generation did not make any changes in the error rate of misclassification. Furthermore, the computational time increased with increasing number of generations.

For testing the datasets, nonparametric Wilcoxon signed rank test was used in this proposed work. However, to test the estimated error rate, 10-fold cross validation was used.

Furthermore, it was shown that the classification error rates for the DA + SVM algorithm were lower than those in the PSO + SVM algorithm [45]. The reason for this is that the DA parameters could be altered iteratively, whereas PSO parameters were fixed and they had to be set first. Thus, DA automatically made the best trade-off between explorations to exploitation. Furthermore, in comparison to the GA + SVM algorithm [46], in most cases, the DA-SVM produced lower classification errors.

The data for both Figures 2(a) and 2(b) are taken from [43], and here it is shown as figures.

Hariharan et al. [47] proposed a combination technique of wavelet packet-based features and the improved version of binary dragonfly optimization (IBDFO) algorithm-based feature selection that was used for classifying various signal types of infant cry. Cry signals were obtained from two different databases. Each database contained a number of samples for different reasons for crying, as shown in Table 7. In Mel Frequent Cepstral Coefficient (MFCC) (16 features), Linear Predictive Coding (LPC) based cepstral (56 features), Wavelet packet transform energy, and nonlinear entropies (496 features) were extracted. IBDFO algorithm was used to overcome the dimensionality problems and choose the most salient features. A wrapper-based feature technique was proposed and various types of infant cry were classified. Extreme learning machine kernel classifier was used, and all and highly informative features were utilized. New updating mechanism and elitism were added to the basic binary dragonfly optimization algorithm (BDFO) to enhance its performance when optimizing the crying features. It was noted that by using a two-class experiment, the percentage rate of recognition accuracy of IBDFO was improved very well compared with BDFO.

It was discovered that the results achieved for IBDFO using seven class experiments were better than those achieved using other techniques (improved binary dragonfly optimization algorithm (IBDFO), GA, and PSO). The results indicated that the combination of feature selection and extraction technique provided better classification accuracy.

In reference [48], DA-based artificial neural network (ANN) model was utilized to estimate India's primary fuel demand. Two multilayer feedforward networks were used. Each of the networks processed input, output, and hidden layers and each network trained with DA. Along with the networks socioeconomic indicators are involved, for example, population and per capita gross domestic product (GDP). The connection weights of ANN models were optimized via searching problem space effectively to find the global best solution. The model proposed in this work required as input the forecast year, and then the primary fuel demands are predicted. The forecast up to the year 2025 was compared with the regression model, and the forecast's accuracy was calculated using the mean absolute percent error (MAPE). This work showed that the proposed model was more accurate than the existing regression model.

Wavelength selection is a notable issue of preprocessing in near-infrared (NIR) in spectroscopy analysis and modelling. Chen and Wang [49] examined a new technique for wavelength selection based on a binary dragonfly algorithm, which consisted of three typical frameworks: multi-BDA, single-BDA, and ensemble learning-based BDA settings. It was discovered that, for the aforementioned problem, using binary-BDA could cause instability. However, both ensemble learning-based BDA and multi-BDA techniques could boost stability. Here, the key technical skill was to reduce the randomization inherent in the BDA. In addition, the computational complexity of multi-BDA was higher than that of the ensemble learning-based BDA, which mainly has an effect on the computation of the fitness function. For performance validation of the abovementioned techniques, the public gasoline NIR spectroscopy dataset was utilized. The aim was to observe the most representative wavelengths for predicting the content of octane. The values of the parameters of the BDA are listed in Table 8. The results proved that like the traditional swarm optimization techniques, the BDA could be used to deal with the wavelength selection problem.

The difference between multi-BDA and single-BDA is that a voting strategy was used to aggregate the results of the wavelength selection of the multiple-time search. From this experiment, it was found that by adjusting the vote's percentage value (VP), the number of selected wavelengths could be controlled. It was obvious that having a small number of selected wavelengths would cause a reduction in the performance of the quantitative analysis model.

Moreover, in the selection of wavelength using the ensemble learning method and the BDA technique before the BDA, a serious of bootstrap sampling generators were added, which was the main difference between this technique and multi-BDA. In this work, it was shown that although the size of the sample reduced using bootstrap sampling, the performance of the quantitative analysis models with the features selected was adjacent to that of the multi-BDA method. Thus, using this technique helped in reducing the computational complexity.

Designing filters in the field of the infinite impulse response (IIR) depends mainly on the conventional selection of parameters filtered among a huge possible combination. The system identification problem requires exploiting the adaptive IIR filter coefficients by using a new algorithm until it is equivalent to the examined unidentified system and adaptive filter. The design of the filter for a problem depends on discovering the optimal set of parameters for the unrevealed model so that it is a close counterpart with the parameters of the filtered benchmark. In reference [50], DA was used to design the IIR

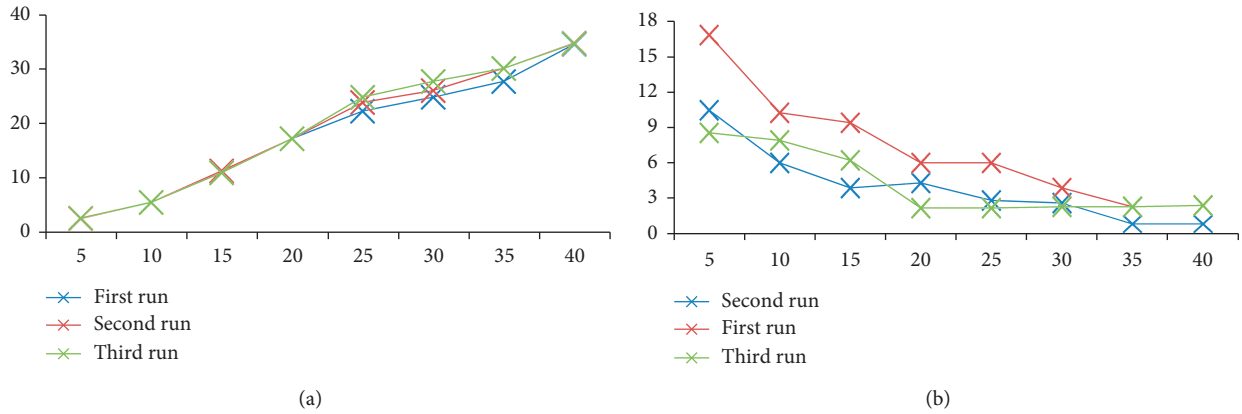


Figure 2: (a) DA-SVM's testing error rate using different number of dragonflies. (b) DA-SVM's computational time with different number of dragonflies.

Table 7: Crying samples in the two utilized databases [47].

Databases	Samples (types of the crying signal)
First database	507 normal crying samples (N) 340 asphyxia crying samples (A) 879 deaf crying samples (D) 350 hungry crying samples (H) 192 pain crying samples (P) 513 jaundice crying samples (J)
Second database	531 premature crying samples (Prem) 45 normal crying samples (N)

Table 8: Parameter values of the BDA [49].

Parameters	Values
Maximum no. of iterations	50
No. of dragonflies	10
No. of wavelengths	401
Separation, alignment, cohesion, food, and enemy factor	Adaptive tuning
No. of principal components	2
No. of folds of cross validation	5

filters instead of using gradient-based methods such as least mean square (LMS). Utilizing the DA prevented locating in the local optima, could evaluate the coefficients close to the actual value, and the minimum mean square value was found. Furthermore, the results proved the superiority of DA against PSO, CSO, and BA to solve the aforementioned problem.

Internet of vehicles (IoV) is utilized to communicate vehicles together. As vehicular nodes are usually considered in moving, hence it makes periodic changes in the topology. Major issues are caused by these changes, such as scalability, routing shortest path, and dynamic changes in topology. Clustering is one of the solutions to such problems. Aadil et al. [51] proposed a new method called the dragonfly-based clustering algorithm (CAVDO) to focus on the scalability of IoV topology. Furthermore, mobility aware dynamic transmission range algorithm (MA-DTR) was used to transmit range adaptation based on traffic density. The proposed work was compared to comprehensive learning PSO (CLPSO) and ant colony optimization. The results proved that in a number of cases CAVDO performed better. The CAVDO performed better in a high density, an average in medium density, and performed worst in low density. However, CLPSO performed well in a very low density only.

6.3. Wireless and Network. Daely and Shin [52] utilized a dragonfly algorithm in two scenarios: (a) to predict the location of randomly deployed nodes in a designated area and (b) to localize different noise percentage of distance measurement (Pn). In both scenarios, the localization was simulated using PSO and DA. In the first scenario, as shown in Figure 3, the simulation results showed that for range-based localization with varying Pn dragonflies could produce fewer errors. In the second scenario, on the other hand, different number of unknown nodes were used for the localization; the simulation result proved that the distances between real and approximated nodes by DA and PSO were increased with the increase of Pn (see, Figure 4).

In radio frequency identification (RFID) network, in order to make an improvement in energy efficiency and maximize it, the network's life should be maximized too by minimizing the use of energy RFID readers and balance the use of energy by every reader in the network. To enlarge the RFID network lifetime, DA was used in reference [53] to develop centralized and protocole-based energy efficient cluster. A high-energy node was used as a cluster head; this devoted less amount of energy while aggregated data were transmitting to the base station. Required residual energy to receive data from the whole readers was defined as a threshold value. The readers with higher leftover energy compared with the threshold value became the cluster head.

The data for both Figures 3 and 4 are taken from [52].

An optimal cluster head among all the cluster heads was chosen using dragonfly clustering. The proposed work decreased the cluster breakage by choosing the cluster head that had similar mobility but high leftover energy. Avoiding the cluster breakage decreased consuming of energy. In addition, redundancy in data was avoided in the cluster head by aggregating the data. Hence, compared with the existing methods, in the RFID network, the efficiency was increased.

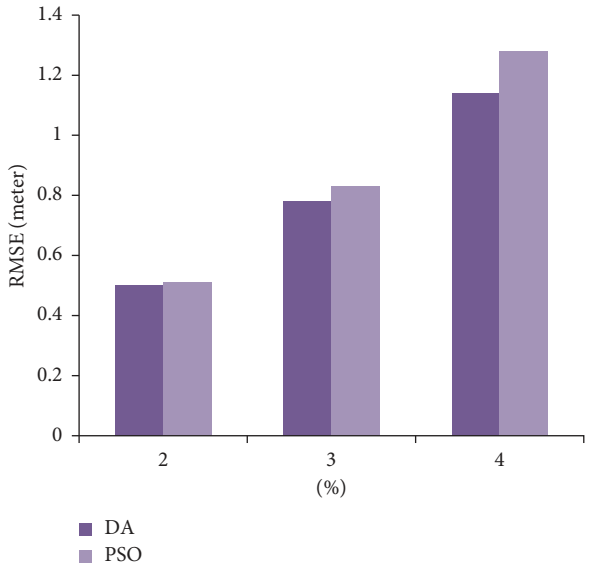


Figure 3: Comparing RMSE between DA and PSO with varying Pn.

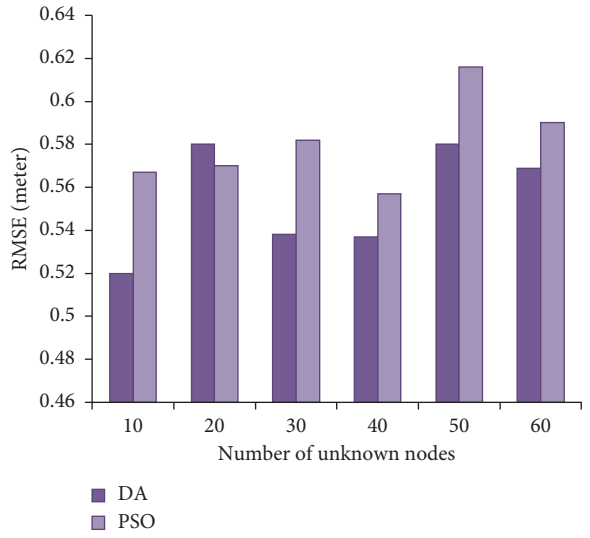


Figure 4: Comparing RMSE between DA and PSO with different number of unknown nodes.

The algorithm steps for selecting a cluster head and formation are described below:

Step 1: initializing the tags and readers in the network R_i

Step 2: assigning potential scores (energy and mobility) to each tag nodes and readers.

Step 3: finding the threshold value by sending the reader's energy level to the base station.

Step 4: if $R_i >$ value of threshold, go to step 5 else go to step 6

Step 5: the readers are updated as eligible cluster head

Step 6: the readers are updated as remaining readers in the network

Step 7: separation, alignment, and cohesion are calculated for the eligible head in the network using equations (8)–(10), respectively

Step 8: add the values found in step 7

Step 9: select the value from step 8 as "Optimal head" if it is high

Step 10: else the value "become ordinary readers" in the network

Step 11: cluster formation ends

In the cloud environment, keeping the stability of processing multiple tasks is a difficult issue. Therefore, a load balancing method is required to allocate the task to the virtual machines (VMs) without influencing the system's performance. Kumar et al. [54] provided a method for load balancing, named as a fractional dragonfly load balancing algorithm (FDLA). In the proposed work two selection probabilities and fractional DA were examined. FDLA was implemented by integrating fractional calculus (FC) into the process of position updating in DA. Equation (15) is the position updating equation for the proposed work. The examined model used certain parameters of physical machines (PMs) and VMs for selecting the function to be reallocated in the VMs for load balancing. Probabilities were used to select the tasks. Task selection probability (TSP) and VM selection probability (VSP) were used. The objective function for the examined technique was based on three objectives, for example, the load of VMs, task migration cost, and the capacity of VMs. The objective was to maximize the solution's fitness. The values of parameters utilized in the proposed work are shown in Table 9.where $Y(t-1)$, $Y(t-2)$, and $Y(t-3)$ represent the individual positions at iterations $(t-1)$, $(t-2)$, and $(t-3)$, respectively and $\Delta Y\,(t+1)$ represents the step vector.

$$(t+1) = \alpha Y(t) + \frac{1}{2}\alpha Y(t-1) + \frac{1}{6}(1-\alpha)Y(t-2) + \frac{1}{24}\alpha(1-\alpha)(2-\alpha)Y(t-3) + \Delta Y(t+1). \tag{15}$$

Three different techniques were compared with the examined technique in this paper. The other techniques are PSO [55], Honey Bee Behaviour-inspired Load Balancing (HBB-LB) [56], and DA (DA was applied instead of the FDLA for balancing the load).

It was observed that the examined work provided a minimum load with allocating 14 tasks. Thus, the proposed FDLA obtained maximum performance than the other techniques. And the number of tasks reduced from 27 to 14 using FDLA compared with the PSO and HBB-LB.

7. The Comparison between DA and Other Algorithms

In reference [57], whale optimization algorithm (WOA), moth-flame optimization (MFO), and DA were compared. In the mentioned reference, these algorithms were implemented to examine the optimal sizing and location of

Table 9: Values of parameters for FDLA [54].

Parameters	Values
Separation weight	0.5
Alignment weight	0.5
Cohesion weight	0.5
Food factor	0.5
Enemy factor	0.5
Population size	10

distributed generation in radial distribution systems to reduce the power loss in the network. For this, multiple-DG units were allocated simultaneously and analyzed by considering two load power factors, i.e., unity and optimal. Bus systems 69 and 119 were used to test the algorithms. Four different cases were used to perform simulation, as shown in Table 10.

The performance for the aforementioned optimization algorithms for cases 1, 2, 3, and 4 proved that the MFO algorithm showed its superiority compared with WOA and DA. It performed better and converged earlier for the mentioned objective functions.

Parmar and Darji [58] used DA, MFO, and WOA to optimize a nonlinear and stochastic optimization problem. The addressed problem in this work was finding the optimum allocation of capacity pricing and capacity between two individual markets for electricity. The markets were having nonidentical designs, and they were interconnected. One of the markets had an energy-only market, while the second one had a capacity-plus-energy market. The objective function for this problem was optimally allocating capacity by generation companies (GenCos) in a way that this allocation could be able to increase general revenue. Likewise, the independent system operator (ISO) acquires energy and capacity; thus, it could reduce the cost of the purchase. In this paper, it was discovered that the maximum value of GenCos's revenue, the capacity price, and the smallest value of ISO's purchase cost were increased with the cost of recall, probability of recall, and the load forecasting error. Furthermore, it was concluded that various algorithms required various numbers of iterations to converge. It is worth mentioning that none of the algorithms were proved its superiority with respect to the examined problem.

Wongsinlatam and Buchitchon [59] focused on court case assignment that has a great impact on improving the judicial system's efficiency. The ability of the judicial system highly depends on the time and the efficiency of operation of the court case. In this work, a mixed integer linear programming (MILP) was used to examine the case assignment issue in the justice court. The assignment problem objective included N cases that should be assigned to M teams and each team had the ability to do all the cases. Nevertheless, due to case specification, personal capability, and working on other cases at the time, the teams needed to spend different time to maximize or minimize the assignment problem's objective. To find the assignment problem's optimal solution DA and FA were used. Two problems were examined for uniform distribution the problems and were shown as, for example, problem one ($P1$): effectiveness rate $(\mu_i) = (1,\ 90)$, lower bound $(L_l) = (1,\ 30)$, upper bound $(U_i) = (1,\ 90)$, and problem two ($P2$): effectiveness rate $(\mu_i) = (1,\ 90)$, lower bound $(L_l) = (1,\ 60)$, and upper bound $(U_i) = (1, 90)$. The produced results proved that DA required less CPU time to find the optimal solution and an average of percent deviation for maximizing effectiveness compared with FA.

The results showed that, for 50 cases and 3 justice teams for experimental parameters $P1$ $(50 : 3, 4, 5)$ and $P2$ $(50 : 3, 4, 5)$, the results of DA were superior compared to those of FA.

In reference [60], DA, moth flame (MFA), and GWO techniques were examined to optimize the capacitor's optimum sitting in different radial distribution systems (RDSs). The factor of loss sensitivity was considered to determine the candidate buses. To validate the efficiency and effectiveness of the examined optimization techniques 33-, 69-, and 118-bus RDSs were considered. The main aim of this study was to minimize power loss and total cost with voltage profile enhancement. The results of the aforementioned optimization techniques were later compared with PSO to prove the superiority of the techniques. To ensure the equality in comparing the results of the techniques, the same initial population was selected for MFO, DA, GWO, and PSO for the 33-bus distribution system. The results proved that DA-, GWO-, and MFO-based optimization methods were much superior compared with PSO-based technique in terms of a small number of iterations and convergence time for the examined study.

Furthermore, MFA-, DA-, and MFA-based optimization showed a higher convergence rate for the 69-bus distribution system case. However, PSO was able to determine the optimal sizing and sitting of the capacitors for the 33-bus system, but it could not find the optimal solution for the 69-bus system accurately. The metrics used to evaluate the performance of algorithms are shown in Table 11.

Additionally, the three algorithms DA, GWO, and MFA were evaluated using statistical tests. The parameter settings were implemented as the original references. Moreover, 35 iterations are used, the population size was 20, and each algorithm runs 30 times for each case. From the evaluation results for the mentioned algorithms, it was observed that DA, GWO, and MFA had a tolerable root mean square error (RMSE). However, with respect to the other two techniques, GWO proved that it has the best values. Additionally, the stability of DA, GWO, and MFA was proved by the values of standard deviation (STD).

8. Advantages and Disadvantages of DA

DA is one of the most recently developed algorithms in the area. As shown in the literature, it has been used to optimize various problems in different areas. One of the reasons that this algorithm has been able to contribute to different applications is that it is very simple and easy to implement. It can be seen that it suits applications in different areas. Furthermore, selecting the predators from the archive, the worst (most populated) hypersphere prevents the artificial dragonflies from searching around nonpromising areas. Moreover, having few parameters for tuning is another

Table 10: Case studies [57].

Case #	The operation mode of DG	System
Case 1	DG operating at a unity power factor	IEEE 69-bus radial distribution system
Case 2	DG operating at an optimal power factor	
Case 3	DG operating at a unity power factor	IEEE 119-bus radial distribution system
Case 4	DG operating at an optimal power factor	

Table 11: Metrics used to evaluate the performance of algorithms [60].

Metric
Relative error (RE)
Mean absolute error (MAE)
Root mean square error (RMSE)
Standard deviation (Std)
Efficiency

advantage of DA. Furthermore, the convergence time of the algorithm is reasonable. Over other optimization algorithms, it is firmer and it easily can be merged with other algorithms.

On the other hand, it does not have an internal memory that can lead to premature convergence to the local optimum. This disadvantage was overcome in reference [26] by proposing a novel Memory-based Hybrid Dragonfly Algorithm (MHDA). Furthermore, DA is easily stuck into local optima because it has a high exploitation rate. Levy flight mechanism was utilized to model the random flying behaviour of dragonflies in nature. The disadvantages of Levy flight are overflowing of the search area and interruption of random flights due to its big searching steps.

9. Results and Evaluations

In the original research work, three groups of classical benchmark functions were used to test the performance of the algorithm. The groups are unimodal (F_1–F_7), multimodal (F_8–F_{13}), and composite test functions (F_{14}–F_{23}). Furthermore, the Wilcoxon ranksum test functions were used to show the significance of the results statistically, as shown in Table 12. The results of F_1–F_{19} in Tables 12 and 13 are taken from the original research work. However, the authors of this research work have tested both PSO and DA for the results of F_{20}–F_{23} in both tables. Though, the DE and FA were tested on all the benchmark functions (F_1–F_{23}) by the authors.

As shown in Table 13, for unimodal test functions, in general, the results of the FA and DE outperformed DA and PSO. This proved that the FA and DE have superior exploitation and greater convergence speed compared to the DA and PSO. On the other hand, in comparison to the PSO, DA showed superior results. Furthermore, the results of the aforementioned references in this research work proved the high convergence speed of the DA. As shown in reference [59], the DA and FA were used to optimize the same problem. The results of DA were superior to that of FA.

In addition, the results of the multimodal test functions proved the high exploration of the DA, which helps in searching the search space. Nonetheless, the results of the DE, in general, were better in this group of the test functions.

For the composite test functions, the FA showed superior results comparing to the other algorithms. The DA has the third place among the four. It was better than the PSO and worse than the other two. This means that the balance of exploration and exploitation of the FA algorithm is superior. The reason for this is that the exploration of the DA is higher than the exploitation rate.

For statistical test functions, the results of the DA and PSO were used. As shown in Table 12, the p values of the unimodal test functions are less than 0.05, which means that the results were statistically significant. For most of the multimodal test functions, as shown in the table, the results were statistically significant and less than 0.05. Moreover, the statistical results of the PSO and DA for most of the composite test functions were significant and less than 0.05.

In the original research work, the DA was not evaluated for large-scale optimization problems using the CEC-C06 benchmark functions. For most of the real-world problems, time is not as important as much as providing an accurate answer. In addition, in reality, people run an algorithm for more than one trail. This means users try to find the most successful technique in their scenario regardless of time. The 100-digit challenge also known as "CEC-C06 benchmark test functions" examines this feature of the number optimization process [61]. In this survey, the algorithm is tested using the CEC-C06 2019 test functions. The utilized CEC-C06 2019 test functions are shown in Table 14. All the CEC test functions are scalable. Furthermore, the CEC04 to CEC10 is shifted, rotated, and these functions are set as minimization problems that have dimension of 10; however, the CEC01 to CEC03 is not shifted and rotated and these functions have different dimensions.

Since in the original paper the results of the DA were compared with the PSO, hence the results of DA for the CEC-C06 test functions will be compared with PSO. 100 iterations and 30 agents were used for both algorithms. The results are presented in Table 15. As shown, the results of both algorithms are comparative in CEC03 and CEC10. However, the DA provided better results in CEC01, CEC02, and CEC06; in the rest of the CEC functions the results of the PSO were better.

The results of the CEC-C06 2019 benchmark functions proved that the DA algorithm can be used to solve large-scale optimization problems.

10. Discussion and Future Works

Unlike evolutionary algorithms and similar to other swarming techniques, the DA algorithm has few parameters

Table 12: The Wilcoxon ranksum test overall runs for the classical benchmark functions.

F	DA	PSO
F_1	N/A	0.045155
F_2	N/A	0.121225
F_3	N/A	0.003611
F_4	N/A	0.307489
F_5	N/A	0.10411
F_6	0.344704	N/A
F_7	0.021134	N/A
F_8	0.000183	N/A
F_9	0.364166	N/A
F_{10}	N/A	0.472676
F_{11}	0.001008	N/A
F_{12}	0.140465	N/A
F_{13}	N/A	0.79126
F_{14}	N/A	0.909654
F_{15}	0.025748	0.241322
F_{16}	0.01133	N/A
F_{17}	0.088973	N/A
F_{18}	0.273036	0.791337
F_{19}	N/A	0.472676
F_{20}	0.938062	0.938062
F_{21}	N/A	N/A
F_{22}	0.256157	0.256157
F_{23}	0.59754	0.59754

Table 13: Comparison of results of the classical benchmark functions between DA, PSO, DE, and FA.

F	Meas.	DA	PSO	DE	FA
F_1	Mean	$2.85E-18$	$4.2E-18$	**2.23e − 19**	$1.72e-10$
	Std.	$7.16E-18$	$1.31E-18$	**1.75e − 19**	$9.43e-10$
F_2	Mean	$1.49E-05$	0.003154	**6.24e − 12**	$6.01e-07$
	Std.	$3.76E-05$	0.009811	**2.2e − 12**	$3.29e-06$
F_3	Mean	$1.29E-06$	0.001891	27.882386	**1.58e − 10**
	Std.	$2.1E-06$	0.003311	12.845742	**8.66e − 10**
F_4	Mean	**0.000988**	0.001748	0.002883	5.913 − 03
	Std.	**0.002776**	0.002515	0.000577	0.029813
F_5	Mean	7.600558	63.45331	9.50058	**2.383765**
	Std.	6.786473	80.12726	3.204155	**1.350716**
F_6	Mean	$4.17E-16$	$4.36E-17$	**2.22e − 19**	$1.9e-10$
	Std.	$1.32E-15$	$1.38E-16$	**1.47e − 19**	$1.04e-09$
F_7	Mean	0.010293	0.005973	0.005256	**1.57e − 04**
	Std.	0.004691	0.003583	0.001649	**1.01e − 04**
F_8	Mean	**−2857.58**	$-7.1E+11$	−4181.932984	−3566.452419
	Std.	**383.6466**	$1.2E+12$	30.0487686	239.113661
F_9	Mean	16.01883	10.44724	**7.31e − 11**	7.462188
	Std.	9.479113	7.879807	**1.04e − 10**	4.41686
F_{10}	Mean	0.23103	0.280137	**2.1e − 10**	$8.47e-07$
	Std.	0.487053	0.601817	**9.14e − 11**	$4.64e-06$
F_{11}	Mean	0.193354	0.083463	**0.001259**	0.053309
	Std.	0.073495	0.035067	**0.002957**	0.053615
F_{12}	Mean	0.031101	$8.57E-11$	**2.32e − 20**	$1.92e-12$
	Std.	0.098349	$2.71E-10$	**3.1e − 20**	$1.05e-11$
F_{13}	Mean	0.002197	0.002197	**4.25e − 20**	$8.21e-12$
	Std.	0.004633	0.004633	**4.52e − 20**	$4.5e-11$
F_{14}	Mean	103.742	150	0.99800	**0.99800**
	Std.	91.24364	135.4006	0	**1.700065e − 16**
F_{15}	Mean	193.0171	188.1951	0.000698	**3.77e − 04**
	Std.	80.6332	157.2834	$1.546e-04$	**1.853–04**
F_{16}	Mean	458.2962	263.0948	**−1.031628**	**−1.031628**
	Std.	165.3724	187.1352	**6.77e − 16**	**1.06e − 15**

Table 13: Continued.

F	Meas.	DA	PSO	DE	FA
F_{17}	Mean	596.6629	466.5429	**0.3978873**	3.0
	Std.	171.0631	180.9493	**0**	$6.05e-15$
F_{18}	Mean	229.9515	136.1759	2.9999999	**−3.862782**
	Std.	184.6095	160.0187	$1.27e-15$	**2.79e − 15**
F_{19}	Mean	679.588	741.6341	**−3.862782**	−3.259273
	Std.	199.4014	206.7296	**2.71e − 15**	0.059789
F_{20}	Mean	**−3.32199**	−3.27047	−3.321797	−9.316829
	Std.	**−3.38E − 06**	0.059923	0.001054	2.21393
F_{21}	Mean	−10.1532	**−7.3874**	−9.867489	−10.147907
	Std.	**6.60E − 15**	3.11458	0.722834	1.396876
F_{22}	Mean	−10.4029	**−8.5305**	−10.381587	−9.398946
	Std.	**1.51E − 06**	3.038572	0.075194	1.99413
F_{23}	Mean	−10.5364	**−9.1328**	−10.530836	−10.2809
	Std.	**2.97E − 07**	2.640148	0.02909	1.39948

For each test function, the result of the algorithm that has shown its superiority comparing to the other algorithms for solving a specific function is shown in bold.

Table 14: CEC-C06 2019 benchmark functions [61].

Function	Functions	Dimension	Range	$f_{\min}$
CEC01	Storn's Chebyshev polynomial fitting problem	9	[−8192, 8192]	1
CEC02	Inverse Hilbert matrix problem	16	[−16384, 16384]	1
CEC03	Lennard–Jones minimum energy cluster	18	[−4, 4]	1
CEC04	Rastrigin's function	10	[−100, 100]	1
CEC05	Grienwank's function	10	[−100, 100]	1
CEC06	Weierstrass function	10	[−100, 100]	1
CEC07	Modified Schwefel's function	10	[−100, 100]	1
CEC08	Expanded schaffer's F_6 function	10	[−100, 100]	1
CEC09	Happy CAT function	10	[−100, 100]	1
CEC10	Ackley Function	10	[−100, 100]	1

Table 15: IEEE CEC-C06 2019 benchmark test results.

CEC function	Meas.	DA	PSO
CEC01	Mean	**46835.63679**	$1.47127E+12$
	Std.	**8992.755502**	$1.32362E+12$
CEC02	Mean	**18.31681239**	15183.91348
	Std.	**0.041929318**	3729.553229
CEC03	Mean	12.70240422	12.70240422
	Std.	$1.50E-12$	$9.03E-15$
CEC04	Mean	103.3295366	**16.80077558**
	Std.	20.00405422	**8.199076134**
CEC05	Mean	1.177303105	**1.138264955**
	Std.	0.057569859	**0.089389848**
CEC06	Mean	**5.646572343**	9.305312443
	Std.	**4.27E − 08**	$1.69E+00$
CEC07	Mean	898.5188217	**160.6863065**
	Std.	4.023921424	104.2035197
CEC08	Mean	6.210996106	**5.224137165**
	Std.	0.001657324	0.786760649
CEC09	Mean	2.601134198	**2.373279266**
	Std.	0.233292964	**0.018437068**
CEC10	Mean	20.0506995	20.28063455
	Std.	0.070920925	0.128530895

For each test function, the result of the algorithm that has shown its superiority comparing to the other algorithms for solving a specific function is shown in bold.

for adjusting, and this makes it easier to implement the algorithm. It provides a good optimization capability. As proved in the aforementioned references, DA has become a strong metaheuristic algorithm to address complex problems in most of the cases. It also provides a good convergence towards global optima. Furthermore, the superiority

and effectiveness of this algorithm were proved by the applications that utilized this technique.

The DA uses low-cohesion weight and high alignment for exploring the search space. For exploiting the search space, on the other hand, low alignment and high-cohesion weights are used. Another technique to balance exploration and exploitation is tuning the swarming weights (s, a, c, f, e, and w) adaptively during the optimization process. To switch between exploitation and exploration, the radii of the neighbourhood enlarged proportionally to the iteration numbers can be used. For small- and medium-scale problems the DA usually can produce good results. For large-scale problems, however, it needs more affords.

One of the difficulties that the users may face of the DA is that position updating and population centroid of the algorithm are not correlated. This may cause trapping into local optima and difficulty in finding global optima and solutions with low accuracy. As mentioned in [18], the performance of the cuckoo algorithm was improved in a number of references by changing the levy flight mechanism. Hence, testing other strategies instead of the levy flight in the DA is highly recommended.

Moreover, the exploration and exploitation of the DA algorithm are mainly determined by alignment, separation, cohesion, and attraction towards food sources and distraction towards enemy sources. This technique improved the diversity of solutions and caused exploration of the algorithm to become stronger. Nevertheless, the performance of the algorithm decreased with a lot of operators of exploration and exploitation because they cause an increase in the convergence time and trapping into local optima. As discussed in [33], for complex optimization problems, the DA easily falls into local optima and the convergence speed is low. However, for simple problems, the static swarming behaviour of the DA increases the exploration level of the algorithm and helps in avoiding local optima. Furthermore, increasing the number of iterations will result in a high exploitation degree and will increase accuracy in finding approximate global optimum.

Additionally, evaluating the algorithm in the previous section proved that the DA may have problems in balancing exploration and exploitation in some cases; this was because the exploration of the DA is high. In the early steps of the optimization process, the high rate of exploration is good; however, it should be decreased in the final steps of the process and the exploitation rate should be increased. Contrarily, the results of the unimodal test functions and the results of most of the reviewed research works proved the superior convergence of the algorithm for simple to medium problems.

To overcome the bottlenecks of the algorithm, it was hybridized with other algorithms. For instance, MHDA was proposed to overcome the problem of premature convergence to local optima. In spite of the fact that the DA and its hybridized versions have been able to provide good results in solving a number of complex optimization problems, some drawbacks still exist. In dragonfly algorithm, attraction towards food and distraction towards enemies provide a high capacity of exploration and exploitation during optimization technique. Nevertheless, the correlation of position updating rule of DA with the centroid of the population from the previous generation is less. Thus, this may result in a solution with low accuracy, premature convergence to local optima, and difficulties in finding the global optima. Hence, research studies are encouraged to find new techniques to update the positions of dragonflies. Furthermore, another point that would help in improving the algorithm is balancing the exploration and the exploitation phases of the algorithm. This would prevent the algorithm to trap into the local optima. Furthermore, combining new search techniques with the DA and examining new transfer functions with the binary DA are highly recommended. We recommend integrating DA with other methods to dynamically tune the parameters during the optimization process. This technique would be able to provide a better balance between exploration and exploitation.

11. Conclusions

In this paper, one of the most recently developed algorithms was reviewed. The different variants of the algorithms including the hybridization versions with other algorithms were discussed. Furthermore, convergence, exploration, and exploitation of the algorithm were addressed. In addition, a number of optimization problems and applications that used DA were reviewed. During the research, it has been discovered that the DA in most of the cases has a good ability to converge towards the global optimum. Moreover, it has the ability to optimize different and complex problems in various areas. Additionally, the results of the benchmark functions proved that the DA has a great ability in solving simple to medium problems. However, for complex problems, it may face some difficulties during the optimization. These difficulties were overcome by merging the algorithm with other algorithms. Moreover, it was also shown that the balance of exploration and exploitation of the algorithm is not good. However, in some applications, it was shown that the balance of exploration and exploitation of the algorithm is reasonable. For example, for optimizing the parameters in the analyzing stress of perforated orthotropic plates, DA outperformed GA and PSO and it converged earlier since it has higher exploration and exploitation rate. Additionally, DA was successfully used to develop a new method to solve economic dispatch incorporating solar energy. The results proved that the economy could be raised and at the same time system loss could be minimized. In a binary search space, DA was successfully used as a searching technique. Furthermore, compared to GA and PSO, BDA showed a better searching ability and showed the ability to select features with more information. Furthermore, for doing multilevel segmentation for colour fundus image, DA showed its superiority over the other techniques that required changing the image to colour scale before doing the segmentation.

Conflicts of Interest

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

Acknowledgments

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

[1] X.-S. Yang and X. He, "Swarm intelligence and evolutionary computation: overview and analysis," in *Studies in Computational Intelligence*, pp. 1–23, Springer, Cham, Switzerland, 2014.

[2] M. Dorigo, *Optimization, learning and natural algorithms*, Ph.D. thesis, Dipartimento di Elettronica Politecnico di Milano, Milan, Italy, 1992.

[3] V. Kothari, J. Anuradha, S. Shah, and P. Mittal, "A survey on particle swarm optimization in feature selection," in *Communications in Computer and Information Science*, pp. 192–201, Springer, Berlin, Germany, 2012.

[4] Y. Zhang, S. Wang, and G. Ji, "A comprehensive survey on particle swarm optimization algorithm and its applications," *Mathematical Problems in Engineering*, vol. 2015, Article ID 931256, 38 pages, 2015.

[5] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the ICNN'95—International Conference on Neural Networks*, Perth, Australia, December 1995.

[6] M. Dorigo and D. Carro, "Ant colony optimization: a new meta-heuristic," in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99*, August 2002.

[7] S.-C. Chu, P.-w. Tsai, and J.-S. Pan, "Cat swarm optimization," in *Lecture Notes in Computer Science*, pp. 854–858, Springer, Berlin, Germany, 2006.

[8] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014.

[9] S. Mirjalili, "Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems," *Neural Computing and Applications*, vol. 27, no. 4, pp. 1053–1073, 2015.

[10] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[11] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm," *Journal of Global Optimization*, vol. 39, no. 3, pp. 459–471, 2007.

[12] J. M. Abdullah and T. Ahmed, "Fitness dependent optimizer: inspired by the bee swarming reproductive process," *IEEE Access*, vol. 7, pp. 43473–43486, 2019.

[13] A. S. Shamsaldin, T. A. Rashid, R. A. Al-Rashid Agha, N. K. Al-Salihi, and M. Mohammadi, "Donkey and smuggler optimization algorithm: a collaborative working approach to path finding," *Journal of Computational Design and Engineering*, vol. 6, no. 4, pp. 562–583, 2019.

[14] X. Yang, "Firefly algorithms for multimodal optimization," in *Stochastic Algorithms: Foundations and Applications*, pp. 169–178, Springer, Berlin, Germany, 2009.

[15] H. M. Mohammed, S. U. Umar, and T. A. Rashid, "A Systematic and Meta-Analysis Survey of Whale Optimization Algorithm," *Computational Intelligence and Neuroscience*, vol. 2019, Article ID 8718571, 25 pages, 2019.

[16] T. Dokeroglu, E. Sevinc, T. Kucukyilmaz, and A. Cosar, "A survey on new generation metaheuristic algorithms," *Computers & Industrial Engineering*, vol. 137, p. 106040, 2019.

[17] R. Rajakumar, P. Dhavachelvan, and T. Vengattaraman, "A survey on nature inspired meta-heuristic algorithms with its domain specifications," in *Proceedings of the 2016 International Conference on Communication and Electronics Systems (ICCES)*, Coimbatore, India, October 2016.

[18] H. Chiroma, T. Herawan, I. Fister et al., "Bio-inspired computation: recent development on the modifications of the cuckoo search algorithm," *Applied Soft Computing*, vol. 61, pp. 149–173, 2017.

[19] X. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Proceedings of the 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*, Coimbatore, India, December 2009.

[20] R. W. Russell, M. L. May, K. L. Soltesz, and J. W. Fitzpatrick, "Massive swarm migrations of dragonflies (Odonata) in eastern North America," *The American Midland Naturalist*, vol. 140, no. 2, pp. 325–342, 1998.

[21] C. W. Reynolds, "Flocks, herds and schools: a distributed behavioral model," *ACM SIGGRAPH Computer Graphics*, vol. 21, no. 4, pp. 25–34, 1987.

[22] S. Mirjalili and A. Lewis, "S-shaped versus V-shaped transfer functions for binary particle swarm optimization," *Swarm and Evolutionary Computation*, vol. 9, pp. 1–14, 2013.

[23] S. Mirjalili and A. Lewis, "Novel performance metrics for robust multi-objective optimization algorithms," *Swarm and Evolutionary Computation*, vol. 21, pp. 1–23, 2015.

[24] C. A. Coello, "Evolutionary multi-objective optimization: some current research trends and topics that remain to be explored," *Frontiers of Computer Science in China*, vol. 3, no. 1, pp. 18–30, 2009.

[25] C. A. C. Coello, G. T. Pulido, and M. S. Lechuga, "Handling multiple objectives with particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 256–279, 2004.

[26] K. S. S Ranjini and S. Murugan, "Memory based hybrid dragonfly algorithm for numerical optimization problems," *Expert Systems with Applications*, vol. 83, pp. 63–78, 2017.

[27] R. P. Parouha and K. N. Das, "A memory based differential evolution algorithm for unconstrained optimization," *Applied Soft Computing*, vol. 38, pp. 501–517, 2016.

[28] H. Ma, D. Simon, M. Fei, X. Shu, and Z. Chen, "Hybrid biogeography-based evolutionary algorithms," *Engineering Applications of Artificial Intelligence*, vol. 30, pp. 213–224, 2014.

[29] M. Salam, H. Zawbaa, E. Emary, K. Ghany, and B. Parv, "A hybrid dragonfly algorithm with extreme learning machine for prediction," in *Proceedings of the 2016 International Symposium on Innovations in Intelligent SysTems and Applications (INISTA)*, Sinaia, Romania, August 2016.

[30] M. Amroune, T. Bouktir, and I. Musirin, "Power system voltage stability assessment using a hybrid approach combining dragonfly optimization algorithm and support vector regression," *Arabian Journal for Science and Engineering*, vol. 43, no. 6, pp. 3023–3036, 2018.

[31] K. Gopalakrishnan, N. Attoh-Okine, and H. Ceylan, *Intelligent and Soft Computing in Infrastructure Systems Engineering*, Springer, Berlin, Germany, 2009.

[32] M. A. Tawhid and K. B. Dsouza, "Hybrid binary dragonfly enhanced particle swarm optimization algorithm for solving feature selection problems," *Mathematical Foundations of Computing*, vol. 1, no. 2, pp. 181–200, 2018.

[33] J. Xu and F. Yan, "Hybrid Nelder–Mead algorithm and dragonfly algorithm for function optimization and the

training of a multilayer perceptron," *Arabian Journal for Science and Engineering*, vol. 44, no. 4, pp. 3473–3487, 2018.

[34] J. Song and S. Li, "Elite opposition learning and exponential function steps-based dragonfly algorithm for global optimization," in *Proceedings of the 2017 IEEE International Conference on Information and Automation (ICIA)*, Macau, China, July 2017.

[35] G. Sayed, A. Tharwat, and A. Hassanien, "Chaotic dragonfly algorithm: an improved metaheuristic algorithm for feature selection," *Applied Intelligence*, vol. 49, no. 1, pp. 188–205, 2018.

[36] R. K. Khadanga, S. Padhy, S. Panda, and A. Kumar, "Design and analysis of tilt integral derivative controller for frequency control in an islanded microgrid: a novel hybrid dragonfly and pattern search algorithm approach," *Arabian Journal for Science and Engineering*, vol. 43, no. 6, pp. 3103–3114, 2018.

[37] J. Wu, Y. Zhu, Z. Wang et al., "A novel ship classification approach for high resolution SAR images based on the BDA-KELM classification model," *International Journal of Remote Sensing*, vol. 38, no. 23, pp. 6457–6476, 2017.

[38] G. Dong and M. Xie, "Color clustering and learning for image segmentation based on neural networks," *IEEE Transactions On Neural Networks*, vol. 16, no. 4, pp. 925–936, 2005.

[39] T. Sağ and M. Çunkaş, "Color image segmentation based on multiobjective artificial bee colony optimization," *Applied Soft Computing*, vol. 34, pp. 389–401, 2015.

[40] S. Rakoth and J. Sasikala, "Multilevel segmentation of fundus images using dragonfly optimization," *International Journal of Computer Applications*, vol. 164, no. 4, pp. 28–32, 2017.

[41] B. Hemamalini and V. Nagarajan, "Wavelet transform and pixel strength-based robust watermarking using dragonflyoptimization," *Multimedia Tools and Applications*, 2018.

[42] D. Połap and M. Woźniak, "Detection of important features from images using heuristic approach," in *Communications in Computer and Information Science*, pp. 432–441, Springer, Berlin, Germany, 2017.

[43] A. Tharwat, T. Gabel, and A. E. Hassanien, "Parameter optimization of support vector machine using dragonfly algorithm," in *Proceedings of the International Conference on Advanced Intelligent Systems and Informatics 2017*, Springer, Cham, Switzerland, 2017.

[44] A. Tharwat, A. E. Hassanien, and B. E. Elnaghi, "A BA-based algorithm for parameter optimization of Support Vector Machine," *Pattern Recognition Letters*, vol. 93, pp. 13–22, 2017.

[45] S.-W. Lin, K.-C. Ying, S.-C. Chen, and Z.-J. Lee, "Particle swarm optimization for parameter determination and feature selection of support vector machines," *Expert Systems with Applications*, vol. 35, no. 4, pp. 1817–1824, 2008.

[46] C.-L. Huang and C.-J. Wang, "A GA-based feature selection and parameters optimizationfor support vector machines," *Expert Systems with Applications*, vol. 31, no. 2, pp. 231–240, 2006.

[47] M. Hariharan, R. Sindhu, V. Vijean et al., "Improved binary dragonfly optimization algorithm and wavelet packet based non-linear features for infant cry classification," *Computer Methods and Programs in Biomedicine*, vol. 155, pp. 39–51, 2018.

[48] J. Kumaran and J. Sasikala, "Dragonfly optimization based ANN model for forecasting India's primary fuels' demand," *International Journal of Computer Applications*, vol. 164, no. 7, pp. 18–22, 2017.

[49] Y. Chen and Z. Wang, "Wavelength selection for NIR spectroscopy based on the binary dragonfly algorithm," *Molecules*, vol. 24, no. 3, p. 421, 2019.

[50] S. Singh, A. Ashok, M. Kumar, Garima, and T. K. Rawat, "Optimal design of IIR filter using dragonfly algorithm," in *Advances in Intelligent Systems and Computing*, pp. 211–223, Springer, Singapore, 2018.

[51] F. Aadil, W. Ahsan, Z. U. Rehman, P. A. Shah, S. Rho, and I. Mehmood, "Clustering algorithm for internet of vehicles (IoV) based on dragonfly optimizer (CAVDO)," *The Journal of Supercomputing*, vol. 74, no. 9, pp. 4542–4567, 2018.

[52] P. T. Daely and S. Y. Shin, "Range based wireless node localization using dragonfly algorithm," in *Proceedings of the 2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*, Vienna, Austria, August 2016.

[53] C. Hema, S. Sankar, and Sandhya, "Energy efficient cluster based protocol to extend the RFID network lifetime using dragonfly algorithm," in *Proceedings of the 2016 International Conference on Communication and Signal Processing (ICCSP)*, Melmaruvathur, India, April 2016.

[54] C. A. Kumar, R. Vimala, K. A. Britto, and S. S. Devi, "FDLA: fractional dragonfly based load balancing algorithm in cluster cloud model," *Cluster Computing*, vol. 22, no. S1, pp. 1401–1414, 2018.

[55] F. Ramezani, J. Lu, and F. K. Hussain, "Task-based system load balancing in cloud computing using particle swarm optimization," *International Journal of Parallel Programming*, vol. 42, no. 5, pp. 739–754, 2013.

[56] L. D. D. Babu and P. V. Krishna, "Honey bee behavior inspired load balancing of tasks in cloud computing environments," *Applied Soft Computing*, vol. 13, no. 5, pp. 2292–2303, 2013.

[57] A. Saleh, A. Mohamed, A. Hemeida, and A. Ibrahim, "Comparison of different optimization techniques for optimal allocation of multiple distribution generation," in *Proceedings of the 2018 International Conference on Innovative Trends in Computer Engineering (ITCE)*, Aswan, Egypt, February 2018.

[58] A. Parmar and P. Darji, "Comparative analysis of optimum capacity allocation and pricing in power market by different optimization algorithms," in *Advances in Intelligent Systems and Computing*, pp. 311–326, Springer, Singapore, 2018.

[59] W. Wongsinlatam and S. Buchitchon, "The comparison between dragonflies algorithm and fireflies algorithm for court case administration: a mixed integer linear programming," *Journal of Physics: Conference Series*, vol. 1061, p. 012005, 2018.

[60] A. A. Z. Diab and H. Rezk, "Optimal sizing and placement of capacitors in radial distribution systems based on grey wolf, dragonfly and moth-flame optimization algorithms," *Iranian Journal of Science and Technology, Transactions of Electrical Engineering*, vol. 43, no. 1, pp. 77–96, 2018.

[61] J. G. Digalakis and K. G. Margaritis, "On benchmarking functions for genetic algorithms," *International Journal of Computer Mathematics*, vol. 77, no. 4, pp. 481–506, 2001.