Journal of Computational Design and Engineering, 2020, DOI: 10.1093/jcde/qwaa037

A Survey on Dragonfly Algorithm and its Applications in Engineering

Chnoor M. Rahman[1], and Tarik A. Rashid[2]
[1]Applied Computer Department, College of Health and Applied Sciences, Charmo University, Sulaimany, Iraq.
[2] School of Science and Engineering, Science and Engineering Department, University of Kurdistan Hewler, Erbil, Iraq.

ABSTRACT Dragonfly algorithm (DA) is one of the most recently developed heuristic optimization algorithms by Mirjalili in 2016. It is now one of the most widely used algorithms. In some cases, it outperforms the most popular algorithms. However, this algorithm is not far from obstacles when it comes to complex optimization problems. In this work, along with the strengths of the algorithm in solving real-world optimization problems, the weakness of the algorithm to optimize complex optimization problems is addressed. This survey presents a comprehensive investigation of DA in the engineering area. First, an overview of the algorithm is discussed. Additionally, the different variants of the algorithm are addressed too. The combined versions of the DA with other techniques and the modifications that have been done to make the algorithm work better are shown. Besides, a survey on applications in engineering area that used DA is offered. The algorithm is compared with some other metaheuristic algorithms to demonstrate its ability to optimize problems comparing to the others. The results of the algorithm from the works that utilized the DA in the literature and the results of the benchmark functions showed that in comparison with some other algorithms DA has an excellent performance, especially for small to medium problems. Moreover, the bottlenecks of the algorithm and some future trends are discussed. Authors conduct this research with the hope of offering beneficial information about the DA to the researchers who want to study the algorithm and utilize it to optimize engineering problems.

KEYWORDS Metaheuristic Algorithms, Optimization Algorithms, Swarm Intelligence, Single objective Optimization, Multi-objective Optimization, DA, Dragonfly Algorithm

I. INTRODUCTION

One of the newest areas of research is Computational Intelligence (CI). CI is a set of methodologies inspired by nature. Researchers can use CI to solve complex real-world problems when the traditional techniques are ineffective. Fuzzy logic, artificial neural network, and evolutionary computation are part of CI.

Swarm intelligence (SI) is part of the evolutionary computation. The efficiency of natural swarm systems amazed natural scientists and biologists to study the behaviours of swarms and creatures. Swarm-based algorithms are part of the nature-inspired population-based algorithm's family. This group of algorithms produce low cost, fast, and robust solutions to complex real-world problems [1]. Bonabeau defined SI as "The emergent collective intelligence of groups of simple agents" [2]. SI systems consist of several agents that form a population. The agents interact locally with each other and their environment. Nature, primarily biological system, was a great inspiration for these algorithms [3]. In SI, agents follow simple rules. No centralized control structure exists to say how individuals should behave. In reality, the individual's behaviours are local and random to an extent. Interaction between agents, however, causes the disclosure of intelligent actions, which are not known to the agents [4]. SI recently has attracted researchers in different fields. Several new algorithms in the base of mimicking the swarm and animal behaviours in nature have been developed by the researchers. The most popular SI algorithms include particle swarm optimization (PSO) proposed by Kennedy and Eberhart [5]. PSO is one of the significant improvements in the field. It mimics the behaviours of a school of bird or fish. A particle represents a single solution that has a position in the search space. Additionally, at the beginning of the 1990s, Marco Dorigo completed his PhD thesis on optimization and nature-inspired algorithms. In his thesis, he examined a novel idea known as ant colony optimization algorithm (ACO) [6]. Chu and Tsai developed Cat Swarm Optimization algorithm (CSO) based on the behaviours of cat [7]. Grey wolf optimizer introduced by Mirjalili et al. [8]. It mimics the hunting behaviour of wolfs. Later, in reference [9], a dragonfly optimization algorithm proposed by the same author. DA mainly inspired by the hunting and migration behaviours of dragonflies. Differential evolution algorithm (DE) developed in[10], is another example. DE is a search population-based technique, inspired by the evolution of living species. Donkey and Smuggler Optimization algorithm (DSO) proposed in [11]. DSO mimics the searching behaviours of donkeys. Searching and selecting routes by donkeys were utilized as an inspiration for the algorithm. Other examples of nature-inspired algorithms are Lion Optimization Algorithm (LOA) by Yazdani et al. [12], and artificial bee colony (ABC) [13]. The LOA mimics the lion's cooperation behaviour and their unique lifestyle. Based on a social organization, the lions divide into residents and nomads. The residents consist of several lions who live together, and they are called pride. Nomads, on the other hand, are mostly seen in pairs and sometimes singularly. Lions may change their lifestyle from nomads to residents or vice versa. However, ABC mimics the actions of honeybees. This algorithm provides well-balanced exploitation and exploration ability.

Dragonfly algorithm is one of the most recently developed and well-known algorithms. It has been successfully used in many different applications and gives satisfactory results. After publishing the algorithm in 2016 until the end of working on this survey (March 2019), almost 300 different works cited dragonfly algorithm in different areas. It produced satisfying results in almost all applications. Thus, in this paper, we centre our review on dragonfly algorithm as one of the most recently developed algorithms in the area.

This work first presents an overview of dragonfly algorithm in section II. Next, the variants of the algorithm are described in section III. The modifications of the DA are presented in section IV. Afterwards, in section IV, the authors address the hybridization versions of the algorithm with other algorithms. In section V, the authors address the applications that used DA in the field of engineering. Additionally, in section V, a comparison between the DA and some other metaheuristics is made. The algorithm is then evaluated using the traditional benchmark functions and the CEC-2019 "The 100-Digit Challenge" benchmark functions in section VII. The results of the evaluations are compared with the PSO and FA. The Wilcoxon rank-sum is used to test the significance of the results statistically. Furthermore, 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.

II. OVERVIEW OF DRAGONFLY ALGORITHM

In the last few decades, the natural behaviour of creatures has widely motivated meta-heuristic optimization algorithms. Swarm intelligence is the main inspiration for the meta-heuristic [5, 14]. DA is a meta-heuristic optimization algorithm. It mimics the dragonfly's swarming behaviours [9].

Dragonflies are small predators. They hunt other insects in nature. The reason for their swarming is hunting and migration (static swarm and dynamic swarm), respectively. In the static swarming phase, dragonflies create sub-swarms and search through different small areas. In dynamic swarming, however, dragonflies fly in a much bigger swarm. They fly in one direction towards the most promising global optimum location [9].

In dynamic swarming, dragonflies maintain a reasonable separation and cohesion (intensification or exploitation). In static swarming, on the other hand, alignment is too low and cohesion is quite high to attack preys (diversification or exploration). Therefore, when exploring the search space, high alignment and low cohesion weights will be assigned to individuals. However, they will be assigned to low alignment, and high cohesion weights while exploiting the search space. The radii of the neighbourhood enlarged proportionally to the number of iterations for changeover between intensification and diversification. Another way for balancing intensification and diversification is tuning the swarming weights adaptively during the optimization. The swarming weights are (separation (*s*), alignment (*a*), cohesion (*c*), attraction motion towards food (*f*), distraction outwards predators (*e*), and inertia weight (*w*)). Following are the equations for the swarming weights:

Equation (2) is for calculating separation, as mentioned by Reynolds [15]:

$$S_i = -\sum_{j=1}^{N} \mathrm{X} - \mathrm{X}_j \tag{1}$$

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

Equation (2) is for calculating 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 a j^{th} neighbouring dragonfly.

Equation (3) is for calculating cohesion:

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

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

Equation (4) is for calculating attraction motion towards food:

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

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, food is the dragonfly that has the best objective function so far.

Equation (5) is for calculating distraction outwards predator:

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

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.

The position of artificial dragonfly individuals in the search space is updated using two vectors: step vector ΔX and position vector X. The step vector in dragonfly algorithm is an analogy to the velocity vector in the PSO algorithm. The position updating procedure of individuals in DA is based mainly on the PSO algorithm framework. The step vector indicates the movement direction of dragonfly individuals, and it is defined in [9] as follows:

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

Where:
s is separation weight.
S_i represents the separation for the i^{th} dragonfly.
a is alignment weigh.
A_i is alignment for i^{th} dragonfly.
c is cohesion weight.
C_i represents the cohesion for i^{th} dragonfly.
f is a food attraction weight.
F_i represents the food source for i^{th} individual.
e is enemies distraction weight.
E_i represents the position of the enemy for i^{th} dragonfly.
w is inertia weight.
t indicates the iteration counter.

When the calculation of the step vector is finished, the calculation for the position vector starts, as follows:

$$X_{t+1} = X_t + \Delta X_{t+1} \quad (7)$$

Where t indicates the current iteration.

To increase the probability of exploring the whole decision space by an optimization technique, we should add a random move to the searching technique. When no neighbouring solutions are there, dragonflies are required to use a random walk (Lévy flight) to fly throughout the search space. In this case, the dragonfly's position updated as follows:

$$X_{t+1} = X_t + Lévy(d) \times X_t \quad (8)$$

Where t is the current iteration and (d) is the dimension of the position vector.

Lévy flight function calculates as follows [16]:

$$Lévy(x) = 0.01 \times \frac{r_1 \times \sigma}{|r_2|^{\frac{1}{B}}} \quad (9)$$

Where r_1 and r_2 are random numbers in [0,1], β is a constant.
σ calculates as follows:

$$\sigma = \left(\frac{\Gamma(1+\beta) \times \sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right) \times \beta \times 2^{\left(\frac{\beta-1}{2}\right)}} \right)^{1/\beta} \quad (10)$$

Where Γ (X) = (X-1)!

Reference [17] stated that although using the Lévy flight improves the performance of DA, however, it may cause very long steps. In the mentioned reference, to avoid this drawback, Brownian motion was used in place of Lévy flight. Brownian motion is another random motion mechanism. The movement of free liquid/gas molecules inspires it. The complexity of the modified DA was O (number of iterations * population size). The calculated complexity proved that using Brownian motion did not have an impact on the complexity time of the original DA. By using the Brownian motion, the massive jumps caused by the Lévy flight were corrected. However, occasionally sudden moves may still be required to avoid trapping into local optima. For objectives with local minima, the Brownian motion produced better solutions in a shorter time.

For the transition from intensification to diversification, dragonflies should adaptively change their weights. As optimization process progress, to adjust the flying path, the neighbourhood area should be expanded, hence at the final stage of optimization, the swarm become one group to converge to the 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 non-promising area of the search space, respectively.

III. VARIANTS OF DRAGONFLY ALGORITHM

DA has three different versions:

A. DA FOR SINGLE OBJECTIVE PROBLEMS

Like most SI-based optimization algorithms, DA initially creates a random set of solutions for optimization problem in hand. At first, the position and step vectors of artificial dragonflies assigned to random values between the upper and lower bounds of the variables. In each iteration, update the position and step vector for each dragonfly. For updating position and step vectors, the neighbourhood of each dragonfly is chosen by Euclidean distance calculation between all the dragonflies and selecting *N* of them. Iteratively, the position updating continues until the end criterion is met. Figure 1 shows the pseudo-code for DA for single-objective problems. The single DA is the most popular variant among the other versions of DA.

B. THE DA FOR BINARY PROBLEMS

Since in binary search space the position vector can only be assigned to 0 or 1, adding step vectors to position vector cannot update the position of search agents. The transfer function converts a continuous SI technique to a binary algorithm. The 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 significant velocity. To use the DA for binary problems (BDA) Equation (11) is used [18].

$$T(\Delta X) = \left|\frac{\Delta X}{\sqrt{\Delta X^2 + 1}}\right| \tag{11}$$

First, the above transfer function calculates the changing probability of position of all artificial dragonflies. In the next step, Equation (12) updates the search agent's location in binary search spaces.

Initialize the dragonflies population Xi (i = 1, 2, ..., n)
Initialize step vectors Xi (i = 1, 2, ..., 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 Eqs. (1) To (5)
  Update neighbouring radius
  ***if** a dragonfly has at least one neighbouring dragonfly*
    Update velocity vector using Eq. (6)
    Update position vector using Eq. (7)
  else
    Update position vector using Eq. (8)
  end if
  Check and correct the new positions based on the boundaries of variables
end while

Figure 1: Pseudo-code for DA [9]

$$X_{t+1} = \begin{cases} \neg X_t & r < T(\Delta X_{t+1}) \\ X_t & r \geq T(\Delta X_{t+1}) \end{cases} \tag{12}$$

Where *r* is a number in [0, 1]

BDA assumes that all of the artificial dragonflies are in one swarm. Hence, it adaptively tunes the swarming factors (*s*, *a*, *c*, *f*, and *e*) and the inertia weight (*w*) to simulate the intensification and diversification. Figure 2 shows the pseudo-code for the BDA.

Reference [19] used BDA for feature selection. This work proved the importance of the role of the transfer function (TF) to produce a discrete space from a continuous one and providing a better balance between exploration and exploitation phases. The proposed work stated that the current version of Equation (11) does not provide a right balance between exploration and exploitation, where at the beginning of the optimization the exploration rate should be higher than the exploitation rate. Hence, to improve the performance of the BDA and prevent trapping into local optima time-dependent TF was used. Equation (13) designed as a new model of TF.

$$T\left(v_i^k(t), \tau\right) = \frac{1}{1 + e^{\frac{-v_i^k(t)}{\tau}}} \tag{13}$$

Where τ refers to a time-varying variable. It initially starts with value and decreases gradually over iterations, as shown in Equation (14).

$$\tau = \left(1 - \frac{t}{T}\right)\tau_{max} + \frac{t}{T}\tau_{min} \tag{14}$$

Where *Tmax* and *Tmin* refer to the maximum and minimum values of τ, T is the maximum number of iterations.

The value of time-dependent TF linearly gets bigger as the step vector of the search agents gets more significant. Consequently, in the early stages of the algorithm, higher exploration is provided (when $\tau = \tau_{max}$). However, as time passes the probability of exploitation increases and the probability of exploration decreases (when $\tau = \tau_{min}$).

As proved in this work, the performance of the BDA improved by using the proposed TF. The main reason for this was providing the right balance between exploration and exploitation phases of the BDA.

The computational complexity of the BDA using the time-dependent TF is the same as the original BDA, and it is *O*(*ISD*).

Where *I* is the number of iterations, *S* is the number of solutions, and *D* is the number of dimensions.

Initialize the dragonflies population Xi (i = 1, 2, ..., n)
Initialize step vectors Xi (i = 1, 2, ..., 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 Eqs. (1) to (5)
    Update step vectors using Eq. (6)
    Calculate the probabilities using Eq (11)
    Update position vectors using Eq.(12)
end while

Figure 2: Pseudo-Code For BDA [9]

C. THE DA FOR MULTI-OBJECTIVE PROBLEMS

For multi-objective problems, there are multiple objectives, and the answer of multi-objective problems is often a set called Pareto Optimal Set. The best trade-offs between the objectives are presented in this set [20]. The Pareto optimal dominance compares two solutions in multi-objective search space [21]. DA is first provided with an archive to save and retrieve the best Pareto optimal solutions during the process of optimization. For updating the position, the food source comes from the archive, and the rest of the process is identical to that of the DA.

Likewise, the multi-objective particle swarm optimization (MOPSO) algorithm [22], to find the well-spread Pareto optimal front, the food source is chosen from the least populated region of the produced Pareto optimal front. A hyper-sphere is defined to cover all the solutions. In each iteration, the hyper-spheres are divided into equal sub-hyper-spheres. When the segments are created, for every segment a roulette-wheel mechanism with the following probability is used for the selection process [23].

$$P_i = \frac{c}{N_i} \quad (15)$$

Where *c* is a constant number, and it is greater than one. N_i is the number of Pareto optimal solution obtained in the i^{th} segment. Equation (15) provides the MODA with a higher probability to choose the food source from the less populated segments. On the other hand, to select predators from the archive, select the worst (most populated) hyper-sphere, so that the artificial dragonflies are discouraged from searching around non-promising areas.
The roulette-wheel mechanism is used for the selection with the following probability for each segment:

$$P_i = \frac{N_i}{c} \quad (16)$$

Where *c* is a constant number and greater than one. N_i is the number of Pareto optimal solutions obtained in the i^{th} segment. In each iteration, the archive needs to be updated regularly, and it may become full during the optimization process. Hence, there should be a mechanism to handle that situation. If at least one of the archive residences dominates the solution, then it should not be allowed to enter the archive. If some of the Pareto optimal solutions dominated by the solution, then they all should be removed from the archive, and the solution should be added to the archive if it is non-dominated concerning to all the solutions in the archive. If the archive is full to accommodate new solution(s), then one or more solutions may be removed from the most populated segments [23]. MODA parameters are identical to those of DA parameters except that the MODA has two new parameters: one for defining the maximum number of hyper-spheres, and the second one for determining the archive size. Pseudo-code for MODA is presented in Figure 3.

In reference [24], the authors modified the multi-objective DA. In this work, the crowded distance selection mechanism from NSGA-III was used instead of the roulette wheel mechanism for the multi-objective DA. NSGA-III developed in [25]. Introducing a set of reference points makes NSGA-III to produce a set of non-dominated solutions that provides better distribution. The new algorithm is named multi-objective dragonfly algorithm based on the reference point (RMODA). The new mechanism gave a better distribution of the solutions and a better convergence to the improved multi-objective DA.

```
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 non-dominated solutions
    Update the archive with concerning the obtained
    non-dominated solutions
    If the archive is full
        Run the archive maintenance mechanism to omit
        one of the current archive members Add the new solution to
        the archive
    end if
    If any of the new added solutions to the archive is
        located outside the hyper-spheres
        Update and re-position all of the hyper-spheres to
        cover the new solution(s)
    end if
    Select a food source from the archive: =
    SelectFood(archive)
    Select an enemy from the archive: =
    SelectEnemy(archive)
    Update step vectors using Eq. (11)
    Update position vectors using Eq. (12)
    Check and correct the new positions based on the boundaries of variables
end while
```

Figure 3: Pseudo-code for MODA [9]

IV. MODIFICATIONS OF DA

In this section, some modifications of the DA are discussed.

A. ELITE OPPOSITION LEARNING AND EXPONENTIAL FUNCTION STEPS-BASED DRAGONFLY ALGORITHM FOR GLOBAL OPTIMIZATION

For enhancing the performance of optimization by DA, reference [26] proposed an improved version of DA. The proposed DA based on elite opposition-based learning strategy and exponential function adaptive steps. The elite individual presented to produce different solutions using elite opposition-based learning. The scope of the search area was expanded by using the mentioned mechanism and was useful to improve the capability of global exploration of the DA.

Furthermore, to replace the original stochastic step, an adaptive step with the exponential phase was designed. The improved proposed work was called dragonfly based on elite opposition-based learning and exponential function adaptive steps (EOEDA). The reason for this modification on the DA was that this algorithm sometimes had problems with solving complex optimization problems, it quickly fell into local optimum, and the speed of its convergence was low. The results of the proposed work compared to some other algorithms, it was concluded that the EOEDA had better convergence accuracy, and the speed of convergence was faster.

B. CHAOTIC DRAGONFLY ALGORITHM: AN IMPROVED METAHEURISTIC ALGORITHM FOR FEATURE SELECTION

In reference [27], a new chaotic dragonfly algorithm (CDA) used to select different features. In CDA, searching iterations of DA embedded to the chaotic maps. To alter the main parameters for movement in DA using the optimization process, ten chaotic maps utilized to increase the convergence rate and efficiency of dragonfly algorithm. The examined technique utilized to pick features in the extracted dataset from drug bank database. It had 6712 drugs. In this work, 553 bio-transformed medications were used. The examined technique used to assess the toxicity of hepatic medications. The proposed model, in general, consisted of three phases: data pre-processing, feature selection, and the classification phase. In step two, CDA was used to select the features. The k-NN classifier utilized to measure the goodness of the selected features. From this experiment, it was discovered that using Gauss chaotic map to adjust variables could significantly enhance DA in terms of stability quality, classification performance, the speed of convergence, and the number of selected features. For evaluation, the chosen features, SVM classifier with various kernel methods used. The results showed that CDA outperformed the other techniques in the literature of the work.

V. HYBRIDIZING DRAGONFLY ALGORITHM WITH OTHER TECHNIQUES

In the metaheuristic context, hybridization refers to merge the robust characteristics of two or more algorithms to provide a new robust algorithm based on the features of the merged ones [28]. In the following subsections, the hybridization versions of the DA are discussed.

A. MEMORY BASED HYBRID DRAGONFLY ALGORITHM FOR NUMERICAL OPTIMIZATION PROBLEMS

In [28], some features of DA and PSO were combined to produce a new hybrid DA called Memory based Hybrid Dragonfly Algorithm (MHDA) for numerical optimization problems. To enhance the performance of dragonfly algorithm, two more features added, they are: (1) to keep track of possible solution, internal memory added. This internal memory has an essential role in converging to global optima. (2) Iterative level hybridization with PSO, which runs on the set of saved solutions. In the following subsections, both of them will be discussed.

1. INTERNAL MEMORY IMPLEMENTATION

The internal memory made the dragonfly individuals keep track of their correlates in the problem space, which is related to the value of fitness. In the PSO algorithm, it calls *pbest*. In each iteration, the best fitness value compared to the search agent's fitness value of the current population. As a result, the DA-*pbest* was created from the better-saved solutions. The dragonfly individuals were also able to keep track of the best value founded so far by any of the dragonflies in the neighbourhood. The mentioned technique is similar to the concept of *gbest* in the PSO. Here DA-*gbest* used to store the best value. The concepts above (*pbest* and *gbest*) enhanced the capability of exploitation in DA. The internal memory feature gave a higher performance compared to the conventional algorithm and trapping into local optima was prevented [29].

2. ITERATION LEVEL HYBRIDIZATION WITH PSO

In the iteration level, to enhance the performance of optimization, two algorithms were executed iteratively in sequence [30]. Here to extend the search space and converge to a more promising area, DA used with internal memory. Then the previously limited area was exploited using PSO to find better solutions. Hence, the hybrid algorithm-MHDA provided better performance compared to the original DA. The reason for this was that the MHDA provided excellent stability between exploration and exploitation phases corresponding to the original DA and PSO. Then the *pbest* and *gbest* of PSO were initialized using DA-*pbest* and DA-*gbest* matrixes, respectively. The equations for velocity and position of PSO were modified, as shown in Equations (17) and (18).

$$V_{k+1}^{i} = wV_{k}^{i} + C_1 r_1 \left(DA - pbest_{k}^{i} - X_{k}^{i}\right) + C_2 r_2 \left(DA - gbest_{k}^{g} - X_{k}^{i}\right) \quad (17)$$

$$X_{k+1}^{i} = X_{k}^{i} + V_{k+1}^{i} \quad (18)$$

Where $DA - pbest_{k}^{i}$ and $DA - gbest_{k}^{g}$ are the *pbest* and *gbest* in PSO, respectively. k represents the size of the swarm.

Thus, to reach global optimal solutions in MHDA, the DA's exploration features in the initial stage, and PSO's exploitation features were combined in the final step. Figure 4 shows the pseudo-code of MHDA. Three constrained engineering problems considered to evaluate the performance of the MHDA. Penalty functions are utilized to handle the constraints. The idea behind using the penalty functions is to transform optimization problems from constrained to unconstrained one through subtracting or adding a specific value from/to the objective function depending on the amount of the violated constraint provided in a particular solution [31]. Results of the three real optimization problems proved the superiority of the method to find the global or near to global optimal solution. MHDA superior performance on unimodal functions showed rapid converge and proper diversification. The diversification capability of MHDA was improved by integrating internal memory, and iteration level hybridization with PSO. MHDA performed better compared to the other algorithms in the literature of the work, and it proved its ability to optimize hard problems.

B. A HYBRID DRAGONFLY ALGORITHM WITH EXTREME LEARNING MACHINE FOR PREDICTION

In [32], DA was used with extreme learning machine (ELM) to overcome the problems in gradient-based algorithms. Here to optimally choose the biases of the hidden layer, DA was used to maximize the ELM's overall performance. DA-ELM converges to the global minimum in a small number of iterations. Moreover, the over-fitting problems in traditional ELM overcame by using the DA-ELM model. In the experiment, the DA-ELM outperformed both GA-ELM and PSO-ELM. DA-ELM showed a good ability in searching the feature space adaptively. Moreover, the high exploration of the DA provided a good ability to the DA-ELM to avoid local minima that may cause premature convergence. Furthermore, DA could find an optimal feature combination with less prediction error. To some extent, the average computational time of DA-ELM was also compatible with PSO-ELM and GA-ELM. The compared and proposed models trained using a thousand iterations. The settings of parameters are shown in Table 1.

Table 1: *Parameter Settings For Algorithms [32]*

Algorithms	Parameters
DA	7 dragonflies 100 iterations
PSO	Inertia factor = 0.1 Individual-best factor = 0.1
GA	Crossover fraction = 0.8 Mutation Fraction = 0.2
ELM	7 input nodes 100 hidden nodes Activation function = Sigmoid 1 output nodes 1000 iterations

```
Initializing set of parameters:
Maximum iteration (Max-iter),
maximum number of search agents (Nmax)
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 Eqs. (2) To (6)
        Update neighbouring radius
        if a dragonfly has at least one neighbouring dragonfly
            Update velocity vector using Eq. (8) Update
            position vector using Eq. (9)
        else
            Update position vector using Eq. (10)
        end if
            Check and correct the new positions based on the boundaries of variables
    end
    ----------End of DA and Start of PSO----------
    For each particle
        Initialize particle with DA-pbest matrix Set PSO-gbest as 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 Eq. (17)
            Update particle position according to Eq. (18)
        end
    end while
    ----------End of PSO----------
    best-fitness = PSO-gbest
end while
```

Figure 4: Pseudo-code for MHDA [28]

C. POWER SYSTEM VOLTAGE STABILITY ASSESSMENT USING A HYBRID APPROACH COMBINING DRAGONFLY OPTIMIZATION ALGORITHM AND SUPPORT VECTOR REGRESSION

Reference [33] proposed a hybrid version of dragonfly algorithm with support vector regression (SVR) for online voltage stability assessment. The performance of SVR is highly dependent on its parameter selection. The critical parameters for SVR include non-sensitivity coefficient ε, penalty parameters C, and the kernel parameters. The DA involved parameter settings of SVR, which improved their performance. For training the proposed model (DFO-SVR), as input, the voltage magnitude produced from PMU buses used for various operating conditions, and the least values of voltage stability index (VSI) used as output variables. Three statistical indices utilized to evaluate the DFO-SVR model. Those statistical indices were correlation coefficient (R), root mean square error (RMSE), and the percentage of mean square error (PRMSE). Depending on the produced results in this work, the proposed model offered excellent performance for prediction.

D. HYBRID BINARY DRAGONFLY ENHANCED PARTICLE SWARM OPTIMIZATION ALGORITHM FOR SOLVING FEATURE SELECTION PROBLEMS

In reference [34], a hybrid version of the binary dragonfly algorithm with enhanced particle swarm optimization algorithm examined for solving the feature selection problem. The examined technique called Hybrid Binary Dragonfly Enhanced Particle Swarm Optimization Algorithm (HBDESPO). The proposed hybrid technique avoided excessive exploitation, and dual exploration used to obtain the solution. In the HBDESPO algorithm, the velocities of both participated algorithms (PSO and DA) updated independently. The velocities to work toward the same goal used different ways, which was worthwhile for the hybrid technique because it derived from the diversity of solutions. The examined system used the K-nearest neighbour (KNN) as a classifier for ensuring the robustness of the training data and reach better feature combinations. The proposed algorithm tested on 20 standard datasets from the UCI repository. A set of assessment indicators utilized to compare and evaluate the various optimizers. The datasets divided into three sets: testing, validation, and training. The value of K assigned to 5 based on trial and error.
The training set used for evaluating the KNN on the validation set. The training set, then, utilized for the final evaluation of the best-nominated feature combination. The minimization problem for this work shown in Equation (19).

$$fitness = \alpha E_R(D) + \beta \frac{|R|}{|C|} \quad (19)$$

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

The proposed technique compared to the results of binary DA from [9], and the enhanced PSO from [35]. The reference concluded that high classification accuracy provided while the ratio of feature selection kept to the minimum. Moreover, small fitness values reached, and across various runs, it kept its stability. Additionally, the value of standard deviation proved the robustness of the algorithm as it repeatedly could converge to a similar solution.

E. HYBRID NELDER–MEAD ALGORITHM AND DRAGONFLY ALGORITHM FOR FUNCTION OPTIMIZATION AND THE TRAINING OF A MULTILAYER PERCEPTRON

In DA, having an excessive number of social interactions may reduce the accuracy of the solution, falling easily into local optima, and imbalance among exploitation and exploration. To control these deficiencies, in reference [36] DA was hybridized with an improved version of the Nelder-Mead algorithm (INMDA) to make the capability of local exploration stronger, and prevent falling into local optima. INMDA consists of two stages. First, DA utilized to search the solution space. The required diversity of the individuals used to find the global optimum. Second, the improved version of the Nelder-Mead (INM) simplex technique used to find the worst and best points, and compute the population centroid. One of the main features of the INM was that the population's centroid used to update the position. Hence, the possibilities of jumping out of the local optima improved.

For single-objective functions, the proposed technique compared to 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 used. The number of agents and the maximum number of iterations were 30 and 1000, respectively.

The results showed high performance of the proposed work for solving high-dimensional problems comparing to DA and MHDA. At the same time, they cannot be used to solve high-dimensional problems because they quickly encounter "dimensional curse". The high performance of the proposed work came from the enhanced exploitation and exploration capabilities of reverse learning techniques.

F. DESIGN AND ANALYSIS OF TILT INTEGRAL DERIVATIVE CONTROLLER FOR FREQUENCY CONTROL IN AN ISLANDED MICROGRID: A NOVEL HYBRID DRAGONFLY AND PATTERN SEARCH ALGORITHM APPROACH

Reference [37] proposed a technique to control frequency in an islanded AC microgrid (MG). MG formed by integrating various sources, such as wind power generation, renewable sources of energy, and solar energy generation. The MG frequency is affected by any variations in the sources. Thus, controlling the frequency of MG is a challenge for the researchers. This work considered a tilt integral derivative (TID) controller for controlling the secondary frequency on the islanded MGs. For tuning control parameters, Hybrid Dragonfly Algorithm and Pattern Search (hDF-PS) utilized. The work concluded that the tilted behaviour of the examined

technique based TID controller produced superior results compared to other conventional controllers. And that an adequate balance provided between the power generation and load, thus, the MG frequency disturbances were overcome.

G. A COGNITIVELY INSPIRED HYBRIDIZATION OF ARTIFICIAL BEE COLONY AND DRAGONFLY ALGORITHMS FOR TRAINING MULTI-LAYER PERCEPTRONS

In reference [38], the strengths of DA combined with ABC. The aim of hybridizing these two metaheuristics was to eliminate slow convergence problem and falling into local optima by providing a better balance between local and global search components of the participated algorithms. The idea behind hybrid ABC/DA (HAD) was combining the exploration ability of ABC with exploration and exploitation ability of the DA. The examined algorithm consists of three components: the dynamic and static swarming behaviour phase in DA and the two phases of global search in ABC. The first component accomplished global search (DA phase), the second component accomplished local search (onlooker phase), and the third one accomplished global search (modified scout bee phase). The proposed hybrid algorithm adopted all the parameters from the original DA and ABC with one more parameter, which is *Prob*. The added parameter used for balancing the dragonfly bee phase, the onlooker bee phase, and balancing between exploitation and exploration. The *prob* parameter set to 0.1 in the experiments carried out in the proposed work, which based on another confirmed research. A population size of *N* and *D* dimensional solutions considered. The time complexity of the iterative process of HAD algorithm analysed as follows: In the first phase, the main operation was creating the initial population, and the complexity time was *O* (ND). In the second phase, the stopping criteria judged, and the time complexity was *O* (1). In the third phase, the value of the *rand* parameter judged. If *rand* is smaller than *prob*, then perform dragonfly bee phase, else perform onlooker bee phase, then perform a modified scout bee phase; the time complexity was *O* (N). In the fourth phase, the solution updated; 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 proposed algorithm was *O* (ND). In terms of convergence speed, the performance of the proposed hybrid method calculated using 50 iterations.

The results showed that HAD provided better performance or comparative performance in terms of convergence speed in almost all the benchmark functions with different dimensions compared to the ABC and DA. In another contribution, this work used the examined hybrid algorithm for training multi-layer perceptron (MLP) neural networks. It observed that the proposed hybrid algorithm was a better trainer for MLPs comparing to the original version of participated algorithms. The experimental results showed that the proposed method showed superior results compared to the standard DA, ABC algorithm, and other well-known algorithms. The reason for the superiority of the proposed technique was that it a better balance provided between exploration and exploitation. Thus, this improved the performance of the mechanism and reduced the probability of trapping into local optima.

H. OPTIMIZATION OF A FRAME STRUCTURE USING THE COULOMB FORCE SEARCH STRATEGY-BASED DRAGONFLY ALGORITHM

Reference [39] proposed an adaptive DA for the structural optimization of frame structures. In this article, the Coulomb force search strategy (CFSS) combined with DA. The exploratory constant parameter (k) is one of the essential parameters in the CFSS. This work examined the utilization of an adaptive value during the search process of the dragonfly algorithm. The dragonflies encouraged for searching in the search space with giant steps at the beginning of the process, and small steps at the end of the process. The above mentioned adaptive strategy improved the convergence of the algorithm. Such that, it provided an optimal result in a short time compared to the original algorithm. The comparing results to the DA and BDA from the proposed technique proved this. The proposed algorithm used to optimize the front axle of an automobile, which is a classic engineering optimization problem. In the examined problem, the front axle beam selected. The results proved the convergence speed of the CFSS-DA comparing to the BDA, and DA. The main reason for the better results was that the DA and BDA do not have any search step optimization strategy.

VI. APPLICATIONS OF DA IN ENGINEERING

Due to the power of DA, enormous research applications from different domains have been conducted, such as engineering, machine learning, image processing, wireless network applications, and some other areas. In this section, we present applications of dragonfly algorithm in Engineering and Physics.

A. MECHANICAL ENGINEERING

Network configuration is the process of changing the status of open/close switches to make changes in the distribution network's topological structure. In [40], a new reconfiguration schema developed to reduce the net deviation among the nominal voltage value, and the node voltages using a dragonfly optimization algorithm. Dragonfly optimization algorithm based reconfiguration method (DORM) enhanced the voltage profile (VP) by net voltage deviation minimization (NVD). The proposed technique examined without making any thermal violations. It also kept the radial structure. In this study, the results obtained using DORM compared to some other nature-inspired algorithms for solving configuration problems, such as PSO [41], GA [42], and BBO [43]. According to the study, the obtained results proved that the DORM provided better configuration that minimized NVD and provided good VP. To share the loads with the conventional power plant, distribute generation units utilized. The mentioned units also used to give the power to the loads individually. Photovoltaic (PV), gas turbine (GT), wind turbine (WT), storage battery (SB), and micro turbine (MT) are the most typical distributed generation units in this type of applications.

In reference [44], a new optimal design of a new hybrid power generation system (HPGS) generated. The introduced design consisted of a combination of PV, WT, GT, and SB. Natural gas distribution network utilized to fuel the GT of the system. To find the optimal design of the proposed work, two metaheuristic techniques; DA [9] and GWO [8] examined. The system considered different weather conditions. Both metaheuristic algorithms in this work used to minimize the total emission functions of the system and the annual cost. It concluded that the DA produced better results in respect of the total yearly cost comparing to GWO. In contrast, regarding the system pollution, GWO technique produced better results than the DA technique.

Perforated plates are part of many industrial applications in recent years. Perforated plate cutouts mostly used to decrease the structure weight or to build a point of exit and entry. Cutouts in the plates can change the geometry of the plate, which leads to severe local stresses (called stress concentration) throughout the cutouts. Stress concentration can cause a reduction in strength and premature failure in structures. Therefore, knowing useful parameters to reduce stress concentration in various structures is crucial. In reference [45], DA used to optimize the involved parameters in analysing stress of the perforated orthotropic plates. The aim was to achieve the minimum stress value around the quasi-triangular cutout located in an infinite orthotropic plate. Distribution of stress calculated using the proposed method based on Lekhnitskii's analytical solution. The variables that designed using the proposed technique included load angle, fibre angle, bluntness, material properties and orientation angle of the cut-out. The results compared to those of the GA and PSO. The parameters of the mentioned algorithms shown in Table 2. The results proved that the average of optimum values of stress produced from the DA was smaller than the other algorithms. It concluded that the values of both average and standard deviation for the DA were smaller than the GA and PSO [46]. The comparison of these techniques proved that DA showed an excellent performance to solve the problem mentioned above, and also it acted more steadily. It was also determined that the high exploration and exploitation rates in the DA were reasons that made the algorithm to perform better. Moreover, DA converged much earlier (18th iteration), whereas PSO and GA converged in the iterations 95th and 146th, respectively. Additionally, depending on the results, it was observed that the most significant levels of stress in all cutout bluntness (w) happened at a load angle of 45°.

Table 2: *Parameter Settings Of The Algorithms [45]*

DA	GA	PSO
Population size = 30	Population size = 30	Population size = 30
Max. No. Of iterations = 200	Max. No. Of Iterations = 200	Max. No. Of Iterations = 200
Random values = $r_1 = r_2 = [0, 1]$	Probability of crossover (Pc) = 0.8	Cognitive component = $c_1 = 2$
Separation weight (s′) = 0.1	Probability of Mutation (Pm) = 0.03	Social component = $c_2 = 2$
Alignment weight (a′) = 0.1	ncrossover = 2*round(npop *Pc/2)	$\omega = \frac{0.1}{\left\lvert 1\frac{c}{2}\sqrt{\frac{\lvert c^2-4c\rvert}{2}}\right\rvert}$, $c = c_1 + c_2$
Cohesion weight (c′) = 0.7	nmut = npop*Pm	
Food factor (f′) = 1		
Enemy factor (e′) = 1		
Inertia factor (δ+ = 0.9-0.2		
Constant (ζ) = 1.5		

The strong non-linear relationship between the elements of the array and array factor makes the problem of concentric circular antenna array (CCAA) synthesis challenging. High maximum side lobe level (MSL) is a problem of CCAAs. Reference [47] used DA to design CCAA in a way that was able to get low side lobes. Sub-structured neural network (SSANN) was used instead of a single artificial neural network (ANN), which improved the prediction accuracy of the efficiency of retraining sub-ANNs and the engine working process. The proposed work aimed at observing and exploring the effectiveness of the DA technique. Moreover, in this work, four different CCAA design cases used to study DA efficiency. Then, the results compared to the existing methods, such as BBO [48], SOS [49], SQP [48], CSO [50], OGSA [51], EP [52], and FA [53]. The proposed work utilized two three-ring designs, CCAA with 4-, 6-, 8- and 8-, 10-, 12-, and two cases considered for each model: CCAA without, and with centre element. For each design experiment, the spacing between neighbouring elements in every ring was fixed to 0.55, 0.606, and 0.75 from the centre to the outermost ring. The results of the DA compared with the techniques in the literature of the work using the uniform array. The results proved that the DA had better performance for the mentioned problem, and it was competitive with other methods for decreasing MSL.

In reference [54], automatic generation control of an interconnected two-area multi-source hydrothermal power system considered. Performance of the examined system was analysed and studied with proportional integral derivative (PID), proportional integral (PI), and 2 degrees of freedom PID (2DOF PID). The DA was used to optimize the controller gains. It concluded that the DA provided superior results compared to classical methods. Furthermore, the 2DOF PID controller optimized by DA produced less overshoot (OS), settling time (ST), undershoot (US). Moreover, smaller values of objective function provided comparing to 2DOF PID controller optimized by DE.

One of the most used operations in the industries is grinding. Optimization can significantly affect the process of grinding by improving the quality of products and reduce operational costs and time of production. Optimizing the grinding process is a challenging process in the engineering field because of the complexity and nonlinearity of the process. In reference [55], multi-objective DA used to get non-dominated Pareto optimal solutions. In this work, an experimental example in [56] was used. Then, the results were compared to the results of an experimental model using NSGA-II in [56]. The Pareto optimal solutions produced by MODA dominated the obtained solutions by NSGA-II. The results proved that in solving the multi-objective mathematical

model of the grinding process, MODA performed better compared to the NSGA-II. The reason for this superiority was due to the MODA's efficient operators compared to the simple operators of NSGA-II (crossover and mutation). The solutions produced by MODA improved surface roughness significantly and reduced the costs and the total grinding time. The results proved that all the objectives were optimized by MODA simultaneously using the algorithm's efficient operators. MODA used 30 individuals and 1000 iterations to examine the mathematical model of tri-objective of the grinding process. On the other hand, NSGA-II utilized 100 chromosomes and 1000 iterations, which resulted in 100,000 number of function evaluations. The results proved that the MODA's computational cost was much lower than the NSGA-II's.

Reference [57] used MODA for optimizing the performance of switched reluctance motor (SRM) powered by autonomous stacked proton exchange membrane fuel cells (PEMFC). MODA used to produce the optimal settings of turn-on and turn-off angles of the driving circuit. As mentioned the optimal settings generated by DA could improve the savings in energy and improve the performance of isolated PEMFC-SRM. The ability of DA in developing the initial stochastic population and the high exploration and exploitation of the algorithm were the reasons for the superiority of the algorithm for solving this problem. Furthermore, In the case of multi-objective problems, DA offers a set of high uniformly distributed Pareto optimal solutions [61].

B. ELECTRICAL ENGINEERING

A new method for designing, modelling and optimizing a uniform serpentine meander based on MEMS switch incorporating beam perforation effect discovered in [62]. In this paper, for pull-in voltage, a new analytical model was proposed. An optimization technique presented to find the optimum configuration of the switch to accomplish minimum pull-in voltage. The analytical model considered as an objective function. For this purpose, the author utilized several high-performance evolutionary optimization algorithms to get the optimum dimensions with computationally less cost and more simplicity. The conducted algorithms included PSO, DE, a hybrid PSO with differential evolution (DEPSO), DA, WOA, and human behaviour based PSO (HBPSO). A comparison among the applied algorithms showed that the DA had the best minimum pull-in voltage with the smallest errors. The parameter settings for DA in the proposed work were: dimension = 8, search agents = 50, alignment weight, separation weight, and cohesion weight were random between -0.2 and 0.2, food attraction weight was a random, and enemy distraction weight was a value between -0.1 and 1. The results showed that the DA performance was the best to minimize pull-in voltage with minimum errors.

In the power transmission system, the stability of voltage is a significant concern due to inconsistency between demand and power generation. Reference [63] utilized the eigenvalue decomposition (EVD) method and DA in partitioned Y-admittance matrix to identify weak buses for implementing the compensators of reactive power. In this work, DA used to optimize the cost and size of the static VAR compensator. For the objective function, voltage deviation, line flows, and reactive power limit was examined as the design constraints. The results proved that the proposed technique maximized the cost of static VAR compensator and the cost of installation with the loading condition. Also, the real power loss and the voltage deviation in the DA were much lesser comparing to the PSO. Moreover, for the IEEE 30 bus system, the DA could show its superiority in reducing real power loss comparing to the other algorithms. Additionally, DA converged earlier.

Atomic generation control (AGC) problem was examined in reference [64] by using DA. In this work, the DA optimized the control parameters, such as PID gains and scaling factors of fuzzy logic. The criterion of integral of time multiplied absolute error (ITAE) was used to minimize the settling time with a minimized peak overshoot. The ITAE employed for optimizing the scaling factor and PID gains controller. The examined control strategy tested with two equal non-reheat thermal interconnected power system areas. The work extended to two hydro-thermal power system areas connected via a high voltage direct current (HVDC) transmission link and an AC tie line. To deal with non-linearity, the generation rate constraint (GRC) effect counted. Moreover, three-area interconnected reheat power system used with an adequate GRC non-linearity in all areas. The results proved that in terms of minimum damping oscillations, settling time, peak undershoots and overshoot in the interconnected three-area power system with GRC non-linearity, the proposed meta-heuristic algorithm based fuzzy PID controller provided superior results compared to other control techniques. The results proved that the DA as an optimization technique produced a better optimal AGC solution for frequency regulation of non-linear and linear interconnected power systems. Furthermore, the combined fuzzy PID controller proposed in this work proved its superiority over the fuzzy logic and optimized PID controller.

C. OPTIMAL PARAMETERS

Reference [65] optimized the parameters in the analysing stress of perforated orthotropic plates. In this work, the DA utilized to reach the smallest stress value around the quasi-triangular cutout in an infinite orthotropic plate. The DA was used to compute the distribution of the stress based on Lekhnitskii's analytical solution. Fibre angle, load angle, orientation cutout, bluntness, and material properties included in the study design variables. In this study, the results obtained from the dragonfly algorithm compared to those of PSO [5] and GA [66]. The results proved that in comparison to the PSO and GA, the DA converged earlier. Besides, avoiding local optimum and producing better results showed the superiority of the DA comparing to the other two algorithms. The DA also produced smaller average values of optimum stress compared to the other algorithms. Furthermore, by using the DA, a standard deviation closer to zero was produced, which was smaller to the ones produced using the PSO and GA.

Providing reliable and continuous supply to customers is a critical ambition of utility, and also meets the expectations of power balance and the loss of transmission when the generators operate within a specified limit. For achieving this purpose, fuel cost and emission value should be as small as possible. The allowed deviation in feasible tolerance and fuel cost is named as emission constrained economic dispatch (ECED) problem. Reference [67] used DA to find a solution for ECED problem. ECED is a multi-objective problem. In this work, fuel cost and emission value together with quadratic function were considered as a multi-objective problem. To convert the problem to single-objective, price penalty factor technique used. The consequences of penalty factors, such as "Min-Min", "Min-Max", "Max-Max", "Max-Min" emission value of different gas exhalation, and price penalty factors mentioned in this work. As the results in this work showed, using "Min-Max" as a price penalty factor produced less fuel cost comparing to the other penalty factors. In "Common", however, increasing ECED fuel cost by 17% could reduce emission by almost 23% in comparison with the price penalty factor of "Min-Max". The author mentioned that nowadays having a small amount of ECED fuel cost to operate thermal power plant with "Min-Max" price penalty creates contamination in the environment and causes premature death in humans leaving near the thermal power plant.

The trend of automotive in the industry is towards electric vehicles (EV). However, for the next coming years, these industries will still use gasoline engines. Reference [68] introduced a new technique to participate in online engine calibration, and to control increasing the performance of the engine, and decrease gas emission of the greenhouse. For this purpose, the mentioned reference used a robust model based on sub-structural neural network (SSANN), multi-objective dragonfly algorithm, multi-objective genetic algorithm (NSGA-II), and fuzzy dependent on inference system. The inputs used for SSANN were injection angle, injection time, throttle angle, and engine rpm. The outputs were: CO, NOx, fuel flow (FF), and torque. Initially, the data from GT-POWER used to train SSANN. Based on various engine speeds, 15 working points were selected randomly to examine the accuracy of SSANN. Linear regression was used to assess the linear relationship between the measured and predicted outputs. For this problem, MODA converged earlier (at 40^{th} generation), and it had better inverse generation distance (IGD). However, NSGA-II converged after the 80^{th} generation. Also, it was discovered that increasing the number of iterations MODA showed better convergence. It was because of the use of food/enemy selection technique in the MODA. In MODA, the search agents move towards the region of the search space that has Pareto optimal solutions with low distribution and ignores the areas with high distribution in the Pareto front.

In reference [69], the dynamic stability of hybrid energy distributed power system (HEDPS) studied. The HEDPS was subject to wind power and load variations. A controller with three degrees of freedom (3-DOF) proportional-integral-derivative (PID) implemented and designed in the HEDPS to balance power and frequency fluctuations after the perturbation. Unlike the single-degree-of-freedom (1-DOF) controller, the 3-DOF controllers own the ability of an outstanding set-point tracking, and it produces superior regulations for the input disturbance. DA used to optimize the parameters of 3-DOF PID controllers. Also, to optimize the 3-DOF controller gains, integral time absolute error (ITAE) used as an objective function. The obtained results compared to the results of Zeigler-Nichols (ZN) and other popular meta-heuristic algorithms. To evaluate the performance of the proposed controller, interconnected, isolated modes of hybrid energy, and distributed power system used. For qualitative assessment, the convergence of DA compared to other algorithms. The results proved that the DA found the global optimum value at a faster rate and that smaller minimum value for the fitness function produced compared to the other participated algorithms. For this work, all the algorithms produced the optimal global point between 60 to 70 generations, which gave the choice of having 100 iterations. Furthermore, the results concluded that the DA outperformed the other mentioned algorithms in terms of faster convergence and minimum fitness value.

D. ECONOMIC LOAD DISPATCH

Wind integrated system with valve-point effect considered in [70]. DA used to overcome the problem of economic load dispatch (ELD) along with valve-point effect. The Weibull distribution function used to model the stochastic nature of wind. Furthermore, closed integral function was used to analyse overestimation/underestimation cost. In the proposed work, the optimization technique started by generating a set of random solutions for the given problem. The dragonfly's vectors (position and step) randomly initialized within upper and lower bounds of generators. The results showed that the DA successfully solved the problem of the power system of economic dispatch of wind thermal integrated system. Two cases and IEEE-30 bus system used to calculate the performance of the algorithm. The problem of non-convex economic dispatch solved in the first case. The obtained results from this case compared to a sequential quadratic programming particle swarm optimization (SQP-PSO) technique. ELD with wind power penetration was solved using DA in the second case. Moreover, the performance of the work in case 2 compared to SQP-PSO [71]. In both cases, 1200MW considered as a load demand. The results showed that the DA found a global optimum solution, and it was remarkably free from trapping into local optima.

In reference [72], dragonfly algorithm used to develop a new technique to resolve economic dispatch incorporating solar energy. In carrying out the economic dispatch, the mentioned reference considered prohibited operating zone and valve-point loading constraints. Beta distribution function used to model the solar energy system, and the objective function. The output forecasted for four different seasons. Different loading conditions considered for each. The proposed work addressed that comparing to other optimization methods dragonfly algorithm gave a low cost, minimum power loss, and converges in the minimum running time. It is concluded that more power could be generated if the availability of sun was abundant in the chosen location. Moreover, in the case of using the produced system power correctly, the economy will maximize, and the system loss will minimize.

The proposed work considered three different cases. In case 1, the system used for testing consisted of six generators and 1263 MW. The results from the first case study compared to the most recent optimization methods. The results proved that the DA was the best regarding the convergence time, the minimum value of the objective function, power loss, and evaluations. Similarly, in terms of generations, cost, and transmission loss, DA was the best. Concerning case 2, the number of used generators was 15, and 2630 MW considered. Here, the total cost generation for the DA was minimum. In case 3, the south Indian 86 bus test system used. It consisted of 7 generators, 131 lines, and 86 buses. This case considered transmission loss, ramp rate constraint, up the reserve, and down reserve constraints. Here, the obtained results proved that the optimal cost of DA was much smaller than the compared algorithms.

E. LOSS REDUCTION

The research work [73] based on the BDA. In this work, a new method proposed for wrapper-selection. The proposed technique aimed at minimizing the number of characteristics compared to the original feature set and obtain better accuracy in classification at the same time. The K-Nearest Neighbourhood (KNN) classifier was used for evaluating the selected feature subset. Feature subset selection is a multi-objective problem. Multi-objective problems examine two different objectives. The proposed work aimed at maximizing the classification accuracy, and minimizing the number of features. Equation (20) shows the objective function. The proposed approach evaluated using 18 UCI datasets. A comparison made between the proposed technique and the similar techniques that used GA, and PSO. The comparison concerned on the accuracy of classification and number of the selected attributes. The results proved that BDA had a better ability in searching the feature space and choosing the features with more information for the classification task.

$$Fitness = a y_R(D) + \beta \frac{|R|}{|C|} \quad (20)$$

Where:

$y_R(D)$ shows the error rate of the classification used.

|R| represents the selected subset's cardinality.

|C| represents the whole number of characteristics in the dataset. a And β are parameters representing the classification importance and length of the subset, respectively. $a \in [0, 1]$ and β = (1- a), the author adopted these from [74].

Reference [75] solved a nearly-zero-energy-building design problem. A comparison made in terms of performance among seven multi-objective algorithms. The comparison included controlled non-dominated sorting genetic algorithm with a passive archive (pNSGA-II), a multi-objective particle swarm optimization, a two-phase optimization using genetic algorithm (PR_GA), an elitist non-dominated sorting evolution strategy (ENSES), a multi-objective evolutionary algorithm based on the concept of epsilon dominance (evMOGA), a multi-objective differential evolution algorithm (spMODE-II), and MODA. The results from other algorithms are out of the scope of this paper. In this work, only the results of MODA are discussed. In most of the cases, the obtained solutions improved with increasing the number of generations. Each algorithm ran 20 times with moderately raising the number of evaluations. The optimization results in most running cases proved that the results of MODA were uncompetitive. In terms of contribution, and running time, MODA was not competitive, and it was slow. According to this work, MODA did not have any outstanding features.

Power loss, electric distribution system's maximum loadability, and voltage stability margin (VSM) are greatly affected by inadequate reactive power generation. To solve these problems, in reference [76], optimal concurrent as well as multiple separate installations of distributed generation (DG), and capacitor were examined. For this work, minimizing the total of reactive power loss (Q_L) counted as the primary objective, and DA used to optimize the problem. Standard 33-bus distribution systems utilized to test the methodology proposed in this work. The proposed work handled different capacitor, and DG installation cases. The results of the proposed work compared to weight improved particle swarm optimization (WIPSO) algorithm. The results proved that the primary behaviour of DA for updating the individual's position provided an enhance Q_L reduction comparing to the other methods. The results also showed a better convergence speed by producing fitter optimized solutions in 15 to 20 iterations.

VII. A COMPARISON BETWEEN DRAGONFLY ALGORITHM AND OTHER ALGORITHMS

In reference [77], whale optimization algorithm (WOA), Moth-Flame optimization (MFO), and DA were compared. The proposed work implemented the above mentioned algorithms to examine the optimal sizing and location of distributed generation in radial distribution systems. The aim of this work was reducing the power loss in the network. Multiple-DG units allocated simultaneously and analysed 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 used to perform the simulation. The results proved that the MFO algorithm is superior comparing to WOA and DA. It performed better and converged earlier for the mentioned objective function.

Reference number [78] used DA, MFO, and WOA to optimize a nonlinear and stochastic optimization problem. The markets had non-identical 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 of this problem was optimally allocating capacity by generation companies (GenCo) 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. The authors in this article discovered that the maximum value of GenCos's revenue, the capacity price, and the smallest value of ISO's purchase cost increased with the cost of recall, probability of recall, and the load forecasting error. It concluded that various algorithms required a different number of iterations to converge, and none of the examined algorithms proved its superiority.

Reference [79] focused on court case assignment that has a great impact on improving judicial system's efficiency. The efficacy of judicial system highly depends on a timely manner and the efficient operation of the court case. In this work, mixed-integer linear programming (MILP) utilized to examine the case assignment issue in the justice court. The objective of the assignment problem was assigning *N* cases to *M* teams. Each team could do all the cases. Nevertheless, due to the case specification, personal capability, and other assigned cases, the teams needed to a spent different time to solve the same case. To find the optimal solution of the assignment problem, the proposed work used DA and firefly algorithm (FA) [51]. Two problems were examined to a uniform distribution. In the form P1: effectiveness rate (μ_i) = (1,90), Lower bound (L_l) = (1, 30), Upper bound (U_i) = (1, 90), and P2: effectiveness rate (μ_i) = (1,90), Lower bound (L_l) = (1, 60), Upper bound (U_i) = (1, 90). The produced results proved that the DA required less time to find the optimal solution and an average percentage deviation for maximizing effectiveness comparing to the FA. The results showed that for 50 cases and three 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 [80], DA, MFO, and GWO techniques examined to optimize the capacitor's optimum sitting in different radial distribution systems (RDSs). The factor of loss sensitivity considered to determine the candidate buses. The authors considered 33-, 69-, and 118-bus RDSs to validate the efficiency and effectiveness of the examined optimization techniques. This study aimed at minimizing power loss and total cost with voltage profile enhancement. The results of the optimization methods mentioned above compared to the PSO to prove the superiority of the techniques. The same initial population was selected for MFO, DA, GWO, and PSO for the 33-bus distribution system. The results showed that DA-, GWO-, and MFO-based optimization methods were much superior compared to the 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 higher convergence rate for 69-bus distribution system case. However, PSO could be 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 69-bus system accurately.

Additionally, the three algorithms, DA, GWO, and MFO, were evaluated using statistical tests. The parameter settings implemented as the original references. For this evaluation, 35 iterations used, population size was 20, and each algorithm ran 30 times for each case. The results showed that DA, GWO, and MFO had a tolerable root-mean-square error (RMSE).

Reference [81] introduced a novel binary multi verse optimization algorithm. In the article, the authors compared the new algorithm to some other binary optimization algorithms, including binary DA. Binary DA ranked as the second-best algorithm, among others, this was because of the excellent balance between exploration and exploitation of the algorithm. Furthermore, the sudden changes in the variables cause a quick convergence.

Reference [82] compared DA with Harris hawks optimization algorithm (HHS). The algorithms utilized to optimize the performance of multi-layer perceptron (MLP), which was used to analyse the stability of two-layered soil. The work compared the accuracy and computational time of the algorithms. Mean square error (MSE), mean absolute error (MAE) and the area under the receiving operating characteristic curve (AUC) used to evaluate the performance of the predictive models. In general, both algorithms helped to improve the applicability accuracy of the MLP. However, the DA reached the lowest error within 500 iterations, whereas the HHS needed 1000 iterations for the same task. Hence, the DA provided a better convergence comparing to the HHS for the problem mentioned above.

VIII. RESULTS AND EVALUATIONS

The results of the applications in the literature showed that dragonfly algorithm is suitable to optimize various problems in the field of engineering. The provided results proved the superiority of the algorithm. In this section, to demonstrate the ability of the algorithm, it is tested on the traditional benchmark functions. Moreover, to further evaluate the algorithm, it was examined on the IEEE Congress of Evolutionary Computation Benchmark Test Functions (CEC-2019), also known as "the 100-digit challenge" [83]. The Wilcoxon rank-sum test functions are utilized to show the significance of the results statistically.

To examine the performance of the algorithm, three groups of traditional benchmark functions utilized with various characteristics in the original work. The groups of the traditional test functions are unimodal (F1-F7), multi-modal (F8-F13), and composite test functions (F14-F23). The results of F1-F19 in Tables 3 and 4 are from the original work. However, the authors of this work examined both PSO and DA for the results of F20-F23 in both tables. Moreover, the authors tested the FA on all the benchmark functions (F1-F23).

Table 3: Classical Benchmark Results of *DA, PSO, and FA*

F	Measurements.	DA	PSO	FA
F_1	Mean	**2.85E-18**	4.2E-18	1.72e-10
	Std.	**7.16E-18**	1.31E-18	9.43e-10
F_2	Mean	**1.49E-05**	0.003154	6.01e-07
	Std.	**3.76E-05**	0.009811	3.29e-06
F_3	Mean	**1.29E-06**	0.001891	**1.58e-10**
	Std.	**2.1E-06**	0.003311	**8.66e-10**
F_4	Mean	**0.000988**	0.001748	5.913-03
	Std.	**0.002776**	0.002515	0.029813
F_5	Mean	**7.600558**	63.45331	**2.383765**
	Std.	**6.786473**	80.12726	**1.350716**
F_6	Mean	4.17E-16	**4.36E-17**	1.9e-10
	Std.	1.32E-15	**1.38E-16**	1.04e-09
F_7	Mean	0.010293	**0.005973**	**1.57e-04**
	Std.	0.004691	**0.003583**	**1.01e-04**
F_8	Mean	-2857.58	**-7.1E+11**	-3566.452419
	Std.	383.6466	**1.2E+12**	239.113661
F_9	Mean	16.01883	**10.44724**	7.462188
	Std.	9.479113	**7.879807**	4.41686
F_{10}	Mean	**0.23103**	0.280137	8.47e-07
	Std.	**0.487053**	0.601817	4.64e-06
F_{11}	Mean	0.193354	**0.083463**	0.053309
	Std.	0.073495	**0.035067**	0.053615
F_{12}	Mean	0.031101	**8.57E-11**	1.92e-12
	Std.	0.098349	**2.71E-10**	1.05e-11
F_{13}	Mean	0.002197	0.002197	8.21e-12
	Std.	0.004633	0.004633	4.5e-11
F_{14}	Mean	**103.742**	150	**0.99800**
	Std.	**91.24364**	135.4006	**1.700065e-16**
F_{15}	Mean	193.0171	**188.1951**	**3.77e-04**
	Std.	80.6332	**157.2834**	**1.853-04**
F_{16}	Mean	458.2962	**263.0948**	**-1.031628**
	Std.	165.3724	**187.1352**	**1.06e-15**
F_{17}	Mean	596.6629	**466.5429**	3.0
	Std.	171.0631	**180.9493**	6.05e-15
F_{18}	Mean	229.9515	**136.1759**	**-3.862782**
	Std.	184.6095	**160.0187**	**2.79e-15**
F_{19}	Mean	**679.588**	741.6341	-3.259273
	Std.	**199.4014**	206.7296	0.059789
F_{20}	Mean	**-3.32199**	-3.27047	-9.316829
	Std.	**-3.38E-06**	0.059923	2.21393
F_{21}	Mean	-10.1532	**-7.3874**	-10.147907
	Std.	**6.60E-15**	3.11458	1.396876
F_{22}	Mean	-10.4029	**-8.5305**	-9.398946
	Std.	**1.51E-06**	3.038572	1.99413
F_{23}	Mean	-10.5364	**-9.1328**	-10.2809
	Std.	**2.97E-07**	2.640148	1.39948

As shown in Table 3, the results of DA on unimodal test functions outperformed the PSO. The results of the unimodal test functions are evident that the DA has an excellent exploitation and convergence speed comparing to the PSO. On the other hand, the results of the FA for the unimodal test functions were superior compared to the DA and PSO. However, the results from the references mentioned above are another evidence for the convergence speed of the DA. Reference [51] utilized the FA and DA to optimize the same problem. The results showed that the DA converged earlier. Furthermore, the *p*-value in Table 4 for this group of test functions is less than 0.05, which means that the results are statistically significant. Moreover, in the references above, DA proved its high convergence speed.

The results for the multi-modal test functions proved that the exploration of the DA is high, which assists in discovering the search space. The PSO showed better results in general in this group of the test functions. Furthermore, as shown in Table 4, these results again are statistically significant because most of the *p*-values are less than 0.05.

For the composite test function, the FA outperformed the DA and PSO. Similar to the unimodal test functions, the results of the PSO and the DA were competitive. However, in some cases, PSO provided better results, which means that the balance of exploration and exploitation in the FA is better than the DA and PSO. The reason for this is that the exploration of the DA algorithm is much higher compared to the exploitation. Moreover, the majority of the statistical results for this group of benchmark functions are significant and less than 0.05, as shown in Table 4.

Table 4: The Wilcoxon Rank-Sum Test Overall Runs For The Classical Benchmark Functions

F	DA	PSO
F1	N/A	0.045155
F2	N/A	0.121225
F3	N/A	0.003611
F4	N/A	0.307489
F5	N/A	0.10411
F6	0.344704	N/A
F7	0.021134	N/A
F8	0.000183	N/A
F9	0.364166	N/A
F10	N/A	0.472676
F11	0.001008	N/A
F12	0.140465	N/A
F13	N/A	0.79126
F14	N/A	0.909654
F15	0.025748	0.241322
F16	0.01133	N/A
F17	0.088973	N/A
F18	0.273036	0.791337
F19	N/A	0.472676
F20	0.938062	0.938062
F21	N/A	N/A
F22	0.256157	0.256157
F23	0.59754	0.59754

Furthermore, in the original work, the CEC benchmark functions were not used to evaluate the DA. Hence, in this work, the CEC-2019 test functions are used to assess the DA further. This group of test functions utilize in annual optimization competition. Professor Suganthan and his colleges improved these benchmark functions to optimize single objective problems [83]. All the CEC-2019 benchmark functions are scalable. However, only function number 4 to function number 10 are shifted and rotated. Whereas, the functions (1 to 3) are not. The functions (1 to 3) have different dimensions. However, functions (4 to 10) set as minimization problems with 10 dimensions. See Table 5 for more details about the functions.

Table 5: CEC-2019 Benchmark Functions "The 100-Digit Challenge" [83]

No.	Functions	Dimension	Range	f_{min}
1	STORN'S CHEBYSHEV POLYNOMIAL FITTING PROBLEM	9	[-8192, 8192]	1
2	INVERSE HILBERT MATRIX PROBLEM	16	[-16384, 16384]	1
3	LENNARD-JONES MINIMUM ENERGY CLUSTER	18	[-4, 4]	1
4	RASTRIGIN'S FUNCTION	10	[-100, 100]	1
5	GRIENWANK'S FUNCTION	10	[-100, 100]	1
6	WEIERSRASS FUNCTION	10	[-100, 100]	1
7	MODIFIED SCHWEFEL'S FUNCTION	10	[-100, 100]	1
8	EXPANDED SCHAFFER'S F6 FUNCTION	10	[-100, 100]	1
9	HAPPY CAT FUNCTION	10	[-100, 100]	1
10	ACKLEY FUNCTION	10	[-100, 100]	1

In the original paper, the DA compared to the PSO; hence, for this group of test functions, DA again will be compared to the PSO. The default parameter settings were not changed during the optimization. For this evaluation, the authors used 100 iterations and 30 agents. As shown in Table 6, DA outperformed the PSO in three CEC-2019 benchmark functions (1, 2, and 3), and the results were competitive in two functions (3 and 10).

IX. DISCUSSION AND FUTURE WORKS

DA is simple and easy to implement. To explore the search space, assign low cohesion weight and high alignment to individuals. Contrarily, to exploit the search space, assign individuals to low alignment and high cohesion weights. Tuning the swarming weights *(s, a, c, f, e,* and *w)* adaptively during the process of optimization is another technique to balance exploration and exploitation. To switch between exploitation and exploration the radii of neighbourhood enlarged proportionally to the iteration numbers can be used. It usually can produce superior results for small to medium-scale problems. However, for large scale optimization problems, more affords are required, and it causes an increase in convergence time and a reduction in performance, which may cause falling into local optima.

Table 6: The IEEE CEC-2019 Benchmark Results For DA, and PSO

Function No.	Measurements	DA	PSO
1	Mean	**46835.63679**	1.47127E+12
	Std.	**8992.755502**	1.32362E+12

2	Mean	**18.31681239**	15183.91348
	Std.	**0.041929318**	3729.553229
3	Mean	12.70240422	12.70240422
	Std.	1.50E-12	9.03E-15
4	Mean	103.3295366	**16.80077558**
	Std.	20.00405422	**8.199076134**
5	Mean	1.177303105	**1.138264955**
	Std.	0.057569859	**0.089389848**
6	Mean	**5.646572343**	9.305312443
	Std.	**4.27E-08**	1.69E+00
7	Mean	898.5188217	**160.6863065**
	Std.	4.023921424	104.2035197
8	Mean	6.210996106	**5.224137165**
	Std.	0.001657324	0.786760649
9	Mean	2.601134198	**2.373279266**
	Std.	0.233292964	**0.018437068**
10	Mean	20.0506995	20.28063455
	Std.	0.070920925	0.128530895

With growing the complexity of optimizing real-world problems, computing demands are hard to be satisfied with the single version of optimization algorithms. One of the obstacles that may face the users of the DA is that the position updating in this algorithm is not so much correlated with the algorithm's population centroid in the preceding generations. Consequently, the produced solutions have low accuracy, and premature convergence to local optima may occur. Additionally, it may cause it difficult to find the global optimal solution. Furthermore, as mentioned earlier, alignment, separation, cohesion, and attraction toward food sources and distraction toward enemy sources mainly determine the exploration and exploitation of the DA. This searching technique maximizes solution diversity and makes the capability of exploration of the DA stronger to some extent. Nevertheless, the performance of the algorithm reduces with a large number of exploration and exploitation operators because they cause an increase in the convergence time, which leads to falling into local optima.

Similar to other metaheuristic algorithms, DA has several strong points, as well as some weak points. It owns powerful optimization capability. The DA has few parameters for adjusting. Most of the time, it can keep a reasonable convergence rate to the global optima. DA is one of the most recently developed algorithms in the area. However, as discussed in the literature, it has been utilized for optimizing a vast number of problems. The simplicity of the algorithm is one of the main reasons for contributing the DA in various applications. Also, selecting the predators from the archive, the worst (most populated) hyper-sphere prevents the artificial dragonflies from searching around non-promising areas. Another advantage is that DA has few parameters for tuning. Similarly, the convergence time of the algorithm is reasonable. Over other optimization algorithms, it is firmer, and it easily can be merged with different algorithms.

On the other hand, as discussed in [36], for complex optimization problems, one of the limitations of the DA is that it easily falls into local optima and the convergence speed is low. It does not have an internal memory that can lead to premature convergence to the local optimum. This disadvantage was overcome in reference [30] by proposing a novel memory-based hybrid dragonfly algorithm (MHDA). Additionally, as presented earlier, DA uses Levy flight as a search process when the neighbourhood does not exist. Nevertheless, the giant steps of the Levy flight mechanism caused an interruption. The original work used step control mechanism to prevent overflowing. However, this distorts the characteristics of the swarm. Also, it is a reason for falling into local optima. Hence, utilizing other searching techniques instead of the Levy flight and comparing the results of the various methods is highly recommended. Moreover, using an adaptive step instead of the original stochastic step will help in balancing the exploration and exploitation phases and improving the performance of the algorithm. Position updating technique is another way to prevent trapping into local optima. Using the population's centroid technique, as discussed in [35] can reduce the probability of locating into local optima.

Furthermore, after evaluating the algorithm in the previous section, it was noted that the DA does not provide a right balance between exploration and exploitation phases, this was because the exploration of the algorithm is high. This high rate of the exploration in the early steps of the optimization process is good, however, in the final iterations of the algorithm, it should be decreased, and the exploitation rate should be increased. For binary dragonfly algorithm, for example, using time-dependent transfer function can improve the balance of exploration and exploitation of the algorithm. Hence, at the beginning of the optimization, the exploration rate is high. The exploration rate gradually decreases during the process and the exploitation rate increases. The mentioned technique will provide a better performance, and it prevents trapping into local optima. Tuning parameters automatically improves the performance of different algorithms. Moreover, it improves the balance between exploration and exploitation phases, and the diversity of the population [84]. On the other hand, the results of the traditional benchmark function for the unimodal benchmark functions and the results of most of the works in literature proved the superior convergence of the algorithm. The superior convergence of the algorithm makes it outperform most of the mentioned algorithms in the literature for solving small to medium problems.

In general, the results from the previous section and the applications in the literature proved that the DA has a significant level of exploration and exploitation. The reason for this is the static swarming behaviour of the algorithm, which increases the exploration level of the algorithm and helps in avoiding local optima for simple problems. Furthermore, increasing the number of iterations will result in high exploitation degree and improves the accuracy in finding approximate global optimum.

However, hybridizing the algorithm with other techniques will give power to the algorithm to overcome the bottlenecks. As discussed, some hybrid versions of DA proposed to overcome the weakness of this algorithm. For example, MHDA was examined for overcoming the shortage that may cause premature convergence to local optima. Moreover, reference [27] utilized Gauss chaotic map to adjust variables. The results showed that the hybridized algorithm in terms of stability quality, classification performance, the speed of convergence, and the number of selected features provided better results. Although the DA and the hybridized DAs provided some good results in solving several 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. The correlation of position updating rule of DA with the centroid of the population from the previous generation is less. Thus, it may result in a solution with low accuracy, premature convergence to local optima, and difficulties in finding the global optima. Hence, researchers are encouraged to find new techniques to update the positions of dragonflies. Another point that will help in improving the algorithm is balancing the exploration and the exploitation phases of the algorithm. A good balance between exploration and exploitation will prevent the algorithm from trapping into the local optima. Furthermore, combining new search techniques with the DA and examining new transfer functions with the binary DA are highly recommended. Moreover, tuning parameters dynamically during the optimization process will have a significant effect on improving the balance of the algorithm's exploration and exploitation.

X. CONCLUSIONS

In this paper, one of the most recently developed algorithms was reviewed. The different versions of the algorithm, including the hybridization versions with other algorithms, were discussed. And most of the optimization problems in engineering and physics that used DA were discussed. From the reviewed works, the authors discovered that DA is one of the useful algorithms in the area. The simplicity of the algorithm was one of the reasons that encouraged the researchers to use the algorithm to optimize the problems in hand. Moreover, the convergence speed and the accuracy of the algorithm are other reasons. As shown, in general, for small to medium problems, the algorithm provided good results. However, similar to other algorithms, for some problems (especially complex problems) DA cannot produce reasonable results. The exploration of the algorithm is high, which may cause trapping into local optima, mainly for the complex problems. Moreover, the results of the benchmark functions (F20-F23) showed that the results of the DA and PSO are competitive. Additionally, the results of the CEC-2019 benchmark function also showed that the DA is comparative with the PSO and FA showed better results than both DA and PSO. Finally, reviewing the DA and its applications proved that we could utilize the mentioned algorithm successfully to optimize almost all the problems in the real world.

As an extension to this work, the authors are willing to find a technique to improve the balance of exploration and exploitation of the algorithm. Likewise, the enactment of the DA can be additionally assessed and compared with other popular algorithms, such as WOA-BAT Optimisation Algorithm [85], Donkey and Smuggler Optimisation Algorithm [11], Fitness Dependent Optimiser [86], and Modified Grey Wolf Optimiser [87].

Funding:

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

REFERENCES

[1] Dorigo M., (1992) “Optimization, Learning and Natural Algorithms”, PhD thesis [in Italian], Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy.

[2] Bonabeau E, Dorigo M, Theraulaz G. “Swarm Intelligence: From Natural to Artificial Systems”. Journal of Artificial Societies and Social Simulation. 1999;4: 320.

[3] Kothari, V., Anuradha, J., Shah, S. and Mittal, P. (2012). A Survey on Particle Swarm Optimization in Feature Selection. *Communications in Computer and Information Science*, [online] pp.192-201. Available at: https://link.springer.com/chapter/10.1007/978-3-642-29216-3_22 [Accessed 2 Feb. 2018].

[4] Zhang, Y., Wang, S. and Ji, G. (2015). *A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications*. [online] Available at: https://www.hindawi.com/journals/mpe/2015/931256/ [Accessed 4 Feb. 2018].

[5] Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. *Proceedings of ICNN'95 - International Conference on Neural Networks*. [online] Available at: https://ieeexplore.ieee.org/document/488968 [Accessed 7 Feb. 2018].

[6] Dorigo, M. and Carro, D. (2002). Ant colony optimization: a new meta-heuristic. In: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99. [online] Washinghton, DC, USA: IEEE. Available at: https://ieeexplore.ieee.org/document/782657/ [Accessed 1 May 2018].

[7] Chu, S., Tsai, P. and Pan, J. (2006). Cat Swarm Optimization. *Lecture Notes in Computer Science*, [online] pp.854-858. Available at: https://link.springer.com/chapter/10.1007/978-3-540-36668-3_94 [Accessed 6 Jan. 2018].

[8] Mirjalili, S., Mirjalili, S. and Lewis, A. (2014). Grey Wolf Optimizer. *Advances in Engineering Software*, [online] 69, pp.46-61. Available at: https://www.sciencedirect.com/science/article/pii/S0965997813001853 [Accessed 3 Jan. 2018].

[9] Mirjalili, S. (2015). Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. *Neural Computing and Applications*, [online] 27(4), pp.1053-1073. Available at: https://link.springer.com/article/10.1007/s00521-015-1920-1 [Accessed 2 Jan. 2018].

[10] Storn, R. and Price, K. (1997). Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *Journal of Global Optimization*, [online] 11(4), pp.341-359. Available at: https://link.springer.com/article/10.1023/A:1008202821328 [Accessed 2 Jan. 2018].

[11] Shamsaldin, A., Rashid, T., Al-Rashid Agha, R., Al-Salihi, N. and Mohammadi, M. (2019). Donkey and smuggler optimization algorithm: A collaborative working approach to path finding. *Journal of Computational Design and Engineering*, [online] 6(4), pp.562-583. Available at: https://www.sciencedirect.com/science/article/pii/S2288430018303178 [Accessed 1 May 2019].

[12] Yazdani, M. and Jolai, F. (2016). Lion Optimization Algorithm (LOA): A nature-inspired metaheuristic algorithm. *Journal of Computational Design and Engineering*, [online] 3(1), pp.24-36. Available at: https://www.sciencedirect.com/science/article/pii/S2288430015000524 [Accessed 8 Mar. 2019].

[13] Karaboga, D. and Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, [online] 39(3), pp.459-471. Available at: https://link.springer.com/article/10.1007/s10898-007-9149-x [Accessed 6 Feb. 2018].

[14] Yang, X. (2014). *Cuckoo search and firefly algorithm: Theory and Applications*. Cham: Springer.

[15] Reynolds, C. (1987). Flocks, herds and schools: A distributed behavioral model. *Proceedings of the 14th annual conference on Computer graphics and interactive techniques - SIGGRAPH '87*, [online] 21(4), pp.25-34. Available at: https://dl.acm.org/citation.cfm?id=37406 [Accessed 14 Feb. 2018].

[16] Yang, X. (2010). “Nature-inspired metaheuristic algorithms”. 2nd ed. Frome (U.K.): Luniver Press.

[17] Acı, Ç. and Gülcan, H. (2019). A Modified Dragonfly Optimization Algorithm for Single- and Multiobjective Problems Using Brownian Motion. *Computational Intelligence and Neuroscience*, [online] 2019, pp.1-17. Available at: https://www.hindawi.com/journals/cin/2019/6871298/ [Accessed 21 Sep. 2019].

[18] Mirjalili, S. and Lewis, A. (2013). S-shaped versus V-shaped transfer functions for binary Particle Swarm Optimization. *Swarm and Evolutionary Computation*, [online] 9, pp.1-14. Available at: https://www.sciencedirect.com/science/article/abs/pii/S2210650212000648 [Accessed 14 Feb. 2018].

[19] Mafarja, M., Aljarah, I., Heidari, A., Faris, H., Fournier-Viger, P., Li, X. and Mirjalili, S. (2018). Binary dragonfly optimization for feature selection using time-varying transfer functions. *Knowledge-Based Systems*, [online] 161, pp.185-204. Available at: https://www.sciencedirect.com/science/article/abs/pii/S095070511830399X?via%3Dihub [Accessed 21 Sep. 2019].

[20] Mirjalili, S. and Lewis, A. (2015). Novel performance metrics for robust multi-objective optimization algorithms. *Swarm and Evolutionary Computation*, [online] 21, pp.1-23. Available at: https://www.sciencedirect.com/science/article/abs/pii/S2210650214000777 [Accessed 22 Feb. 2018].

[21] Coello Coello, C. (2009). Evolutionary multi-objective optimization: some current research trends and topics that remain to be explored. *Frontiers of Computer Science in China*, [online] 3(1), pp.18-30. Available at: https://link.springer.com/article/10.1007/s11704-009-0005-7 [Accessed 22 Feb. 2018].

[22] Coello Coello, C. and Lechuga, M. (2002). MOPSO: a proposal for multiple objective particle swarm optimization. *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*. [online] Available at: https://ieeexplore.ieee.org/document/1004388 [Accessed 22 Feb. 2018].

[23] Coello, C., Pulido, G. and Lechuga, M. (2004). Handling multiple objectives with particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, [online] 8(3), pp.256-279. Available at: https://ieeexplore.ieee.org/document/1304847 [Accessed 9 Mar. 2018].

[24] Li, J., Lu, J., Yao, L., Cheng, L. and Qin, H. (2019). Wind-Solar-Hydro power optimal scheduling model based on multi-objective dragonfly algorithm. *Energy Procedia*, [online] 158, pp.6217-6224. Available at: https://www.sciencedirect.com/science/article/pii/S1876610219304990 [Accessed 21 Sep. 2019].

[25] Deb, K. and Jain, H. (2014). An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints. *IEEE Transactions on Evolutionary Computation*, [online] 18(4), pp.577-601. Available at: https://ieeexplore.ieee.org/document/6600851 [Accessed 25 Nov. 2019].

[26] Song, J. and Li, S. (2017). Elite opposition learning and exponential function steps-based dragonfly algorithm for global optimization. *2017 IEEE International Conference on Information and Automation (ICIA)*. [online] Available at: https://ieeexplore.ieee.org/abstract/document/8079080 [Accessed 2 Mar. 2019].

[27] Sayed, G., Tharwat, A. and Hassanien, A. (2019). Chaotic dragonfly algorithm: an improved metaheuristic algorithm for feature selection. *Applied Intelligence*, [online] 49(1), pp.188-205. Available at: https://link.springer.com/article/10.1007%2Fs10489-018-1261-8.

[28] K.S., S. and Murugan, S. (2017). Memory based Hybrid Dragonfly Algorithm for numerical optimization problems. *Expert Systems with Applications*, [online] 83, pp.63-78. Available at: https://www.sciencedirect.com/science/article/pii/S0957417417302762 [Accessed 9 Mar. 2018].

[29] Parouha, R. and Das, K. (2016). A memory based differential evolution algorithm for unconstrained optimization. *Applied Soft Computing*, [online] 38, pp.501-517. Available at: https://www.sciencedirect.com/science/article/pii/S1568494615006602 [Accessed 5 Mar. 2018].

[30] Ma, H., Simon, D., Fei, M., Shu, X. and Chen, Z. (2014). Hybrid biogeography-based evolutionary algorithms. *Engineering Applications of Artificial Intelligence*, [online] 30, pp.213-224. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0952197614000189 [Accessed 5 Mar. 2018].

[31] Di Pillo, G. and Grippo, L. (1989). Exact Penalty Functions in Constrained Optimization. SIAM Journal on Control and Optimization, [online] 27(6), pp.1333-1360. Available at: https://epubs.siam.org/doi/abs/10.1137/0327068 [Accessed 24 Nov. 2019].

[32] Salam, M., Zawbaa, H., Emary, E., Ghany, K. and Parv, B. (2016). A hybrid dragonfly algorithm with extreme learning machine for prediction. *2016 International Symposium on INnovations in Intelligent SysTems and Applications (INISTA)*. [online] Available at: https://ieeexplore.ieee.org/document/7571839 [Accessed 11 Mar. 2018].

[33] Amroune M, Bouktir T, Musirin I (2018) Power system voltage stability assessment using a hybrid approach combining dragonfly optimization algorithm and support vector regression. *Arab J Sci Eng*. Available at: https://doi.org/10.1007/s13369-017-3046-5 [Accessed 24 Feb. 2019].

[34] A. Tawhid, M. and B. Dsouza, K. (2018). Hybrid binary dragonfly enhanced particle swarm optimization algorithm for solving feature selection problems. *Mathematical Foundations of Computing*, [online] 1(2), pp.181-200. Available at: http://aimsciences.org/article/doi/10.3934/mfc.2018009?viewType=html [Accessed 2 Mar. 2019].

[35] Mirjalili, S. and Lewis, A. (2013). S-shaped versus V-shaped transfer functions for binary Particle Swarm Optimization. *Swarm and Evolutionary Computation*, [online] 9, pp.1-14. Available at: https://www.sciencedirect.com/science/article/abs/pii/S2210650212000648 [Accessed 2 Mar. 2019].

[36] Xu, J. and Yan, F. (2018). Hybrid Nelder–Mead Algorithm and Dragonfly Algorithm for Function Optimization and the Training of a Multilayer Perceptron. *Arabian Journal for Science and Engineering*. [online] Available at: https://link.springer.com/article/10.1007%2Fs13369-018-3536-0 [Accessed 2 Mar. 2019].

[37] Khadanga, R., Padhy, S., Panda, S. and Kumar, A. (2018). 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*, [online] 43(6), pp.3103-3114. Available at: https://link.springer.com/article/10.1007/s13369-018-3151-0 [Accessed 12 Mar. 2019].

[38] Ghanem, W. and Jantan, A. (2018). A Cognitively Inspired Hybridization of Artificial Bee Colony and Dragonfly Algorithms for Training Multi-layer Perceptrons. *Cognitive Computation*, [online] 10(6), pp.1096-1134. Available at: https://link.springer.com/article/10.1007/s12559-018-9588-3 [Accessed 12 Mar. 2019].

[39] Yuan, Y., Lv, L., Wang, X. and Song, X. (2019). Optimization of a frame structure using the Coulomb force search strategy-based dragonfly algorithm. *Engineering Optimization*, [online] pp.1-17. Available at: https://www.tandfonline.com/doi/full/10.1080/0305215X.2019.1618290 [Accessed 21 Sep. 2019].

[40] K., T. and Aravindhababu, P. (2017). Dragonfly Optimization based Reconfiguration for Voltage Profile Enhancement in Distribution Systems. *International Journal of Computer Applications*, [online] 158(3), pp.1-4. Available at: https://www.semanticscholar.org/paper/Dragonfly-Optimization-based-Reconfiguration-for-in-Abhiraj/6970179fb3b97a55dc881e8aa1c42e4dac44dc5d [Accessed 11 Mar. 2018].

[41] Andervazh, M., Haghifam, M. and Olamaei, J. (2013). Adaptive multi-objective distribution network reconfiguration using multi-objective discrete particles swarm optimisation algorithm and graph theory. *IET Generation, Transmission & Distribution*, [online] 7(12), pp.1367-1382. Available at: https://ieeexplore.ieee.org/document/6674159 [Accessed 11 Mar. 2018].

[42] Gupta, N., Swarnkar, A. and Niazi, K. (2014). Distribution network reconfiguration for power quality and reliability improvement using Genetic Algorithms. *International Journal of Electrical Power & Energy Systems*, [online] 54, pp.664-671. Available at: https://www.sciencedirect.com/science/article/pii/S0142061513003578 [Accessed 11 Mar. 2018].

[43] Aruul, S. and Santhi, R. (2016). New Reconfiguration Method for Improving Voltage Profile of Distribution Networks. *International Journal of Computer Applications*, [online] 135(7), pp.25-29. Available at: https://www.semanticscholar.org/paper/New-Reconfiguration-Method-for-Improving-Voltage-of-Santhi-Jabr/382de46c554feb5d53a550579f7aa48af9ddd6ce [Accessed 11 Mar. 2018].

[44] Algabalawy, M., Mekhamer, S. and Abdelaziz, A. (2017). Optimal Design of a New Configuration of the Distributed Generation Units using Grey Wolf and Dragonfly Optimizers. *MASK International Journal of Science and Technology*. [online] 2(1). Available at: https://www.academia.edu/31230265/Optimal_Design_of_a_New_Configuration_of_the_Distributed_Generation_Units_using_Grey_Wolf_and_Dragonfly_Optimizers. [Accessed 11 Mar. 2018].

[45] Jafari, M. and Bayati Chaleshtari, M. (2017). Using dragonfly algorithm for optimization of orthotropic infinite plates with a quasi-triangular cut-out. *European Journal of Mechanics - A/Solids*, [online] 66, pp.1-14. Available at: https://www.sciencedirect.com/science/article/pii/S0997753817304370 [Accessed 12 Feb. 2019].

[46] Bloomfield, M., Herencia, J. and Weaver, P. (2010). Analysis and benchmarking of meta-heuristic techniques for lay-up optimization. *Computers & Structures*, [online] 88(5-6), pp.272-282. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0045794909002648 [Accessed 12 Feb. 2019].

[47] Babayigit, B. (2017). Synthesis of concentric circular antenna arrays using dragonfly algorithm. *International Journal of Electronics*, [online] 105(5), pp.784-793. Available at: https://doi.org/10.1080/00207217.2017.1407964 [Accessed 13 Feb. 2019].

[48] Bloomfield, M., Herencia, J. and Weaver, P. (2010). Analysis and benchmarking of meta-heuristic techniques for lay-up optimization. *Computers & Structures*, [online] 88(5-6), pp.272-282. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0045794909002648 [Accessed 12 Feb. 2019].

[49] Dib, N. (2017). Design of planar concentric circular antenna arrays with reduced side lobe level using symbiotic organisms search. *Neural Computing and Applications*, [online] 30(12), pp.3859-3868. Available at: https://link.springer.com/article/10.1007/s00521-017-2971-2 [Accessed 13 Feb. 2019].

[50] Ram, G., Mandal, D., Kar, R. and Ghoshal, S. (2015). Circular and Concentric Circular Antenna Array Synthesis Using Cat Swarm Optimization. *IETE Technical Review*, [online] 32(3), pp.204-217. Available at: https://www.tandfonline.com/doi/abs/10.1080/02564602.2014.1002543?journalCode=titr20 [Accessed 14 Feb. 2019].

[51] Ram, G., Mandal, D., Kar, R. and Prasad Ghoshal, S. (2015). Opposition-based gravitational search algorithm for synthesis circular and concentric circular antenna arrays. *scientia Iranica*, [online] 22(6). Available at: http://scientiairanica.sharif.edu/article_3796_e57ae2fe002d2736cf943f020f5ac2fc.pdf [Accessed 14 Feb. 2019].

[52] Mandal, D., Ghoshal, S. and Bhattacharjee, A. (2010). Design of Concentric Circular Antenna Array with Central Element Feeding Using Particle Swarm Optimization with Constriction Factor and Inertia Weight Approach and Evolutionary Programing Technique. *Journal of Infrared, Millimeter, and Terahertz Waves*, [online] 31(6), pp.667-680. Available at: https://link.springer.com/article/10.1007/s10762-010-9629-9 [Accessed 14 Feb. 2019].

[53] Sharaqa, A. and Dib, N. (2013). Circular antenna array synthesis using firefly algorithm. *International Journal of RF and Microwave Computer-Aided Engineering*, [online] 24(2), pp.139-146. Available at: https://onlinelibrary.wiley.com/doi/abs/10.1002/mmce.20721 [Accessed 15 Feb. 2019].

[54] Simhadri, K., Mohanty, B. and Mohan Rao, U. (2018). Optimized 2DOF PID for AGC of Multi-area Power System Using Dragonfly Algorithm. *Advances in Intelligent Systems and Computing*, [online] pp.11-22. Available at: https://link.springer.com/chapter/10.1007/978-981-13-1819-1_2 [Accessed 20 Feb. 2019].

[55] Khalilpourazari, S. and Khalilpourazary, S. (2018). Optimization of time, cost and surface roughness in grinding process using a robust multi-objective dragonfly algorithm. *Neural Computing and Applications*, [online] pp.1-12. Available at: https://link.springer.com/article/10.1007/s00521-018-3872-8 [Accessed 19 Feb. 2019].

[56] Gholami, M. and Azizi, M. (2014). Constrained grinding optimization for time, cost, and surface roughness using NSGA-II. *The International Journal of Advanced Manufacturing Technology*, [online] 73(5-8), pp.981-988. Available at: https://link.springer.com/article/10.1007/s00170-014-5884-6 [Accessed 20 Feb. 2019].

[57] El-Hay, E., El-Hameed, M. and El-Fergany, A. (2018). Improved performance of PEM fuel cells stack feeding switched reluctance motor using multi-objective dragonfly optimizer. *Neural Computing and Applications*. [online] Available at: https://link.springer.com/article/10.1007%2Fs00521-018-3524-z [Accessed 21 Sep. 2019].

[58] Shilaja, C. and Ravi, K. (2017). Optimal Power Flow Using Hybrid DA-APSO Algorithm in Renewable Energy Resources. *Energy Procedia*, [online] 117, pp.1085-1092. Available at: https://www.sciencedirect.com/science/article/pii/S1876610217324736?via%3Dihub [Accessed 21 Sep. 2019].

[59] Suresh, V. and Sreejith, S. (2016). Generation dispatch of combined solar thermal systems using dragonfly algorithm. *Computing*, [online] 99(1), pp.59-80. Available at: https://link.springer.com/article/10.1007%2Fs00607-016-0514-9 [Accessed 21 Sep. 2019].

[60] Veeramsetty, V., Venkaiah, C. and Kumar, D. (2017). Hybrid genetic dragonfly algorithm based optimal power flow for computing LMP at DG buses for reliability improvement. *Energy Systems*, [online] 9(3), pp.709-757. Available at: https://link.springer.com/article/10.1007%2Fs12667-017-0268-2 [Accessed 21 Sep. 2019].

[61] Amroune, M., Bouktir, T. and Musirin, I. (2018). Power System Voltage Stability Assessment Using a Hybrid Approach Combining Dragonfly Optimization Algorithm and Support Vector Regression. *Arabian Journal for Science and Engineering*, [online] 43(6), pp.3023-3036. Available at: https://link.springer.com/article/10.1007%2Fs13369-017-3046-5 [Accessed 21 Sep. 2019].

[62] Guha, K., Laskar, N., Gogoi, H., Borah, A., Baishnab, K. and Baishya, S. (2017). Novel analytical model for optimizing the pull-in voltage in a flexured MEMS switch incorporating beam perforation effect. *Solid-State Electronics*, [online] 137, pp.85-94. Available at: https://www.sciencedirect.com/science/article/pii/S0038110117300229 [Accessed 14 Apr. 2018].

[63] Vanishree, J. and Ramesh, V. (2017). Optimization of size and cost of static var compensator using dragonfly algorithm for voltage profile improvement in power transmission systems. *International Journal of Renewable Energy Research (IJRER)*, [online] 8(1), pp.56-66. Available at: https://ijrer.org/ijrer/index.php/ijrer/article/view/6933 [Accessed 20 Feb. 2019].

[64] Kouba, N., Menaa, M., Hasni, M. and Boudour, M. (2018). A Novel Optimal Combined Fuzzy PID Controller Employing Dragonfly Algorithm for Solving Automatic Generation Control Problem. *Electric Power Components and Systems*, [online] pp.1-17. Available at: https://www.tandfonline.com/doi/abs/10.1080/15325008.2018.1533604?af=R&journalCode=uemp20 [Accessed 23 Feb. 2019].

[65] Jafari, M. and Bayati Chaleshtari, M. (2017). Using dragonfly algorithm for optimization of orthotropic infinite plates with a quasi-triangular cut-out. *European Journal of Mechanics - A/Solids*, [online] 66, pp.1-14. Available at: https://www.sciencedirect.com/science/article/pii/S0997753817304370 [Accessed 2 Mar. 2018].

[66] Mathworks.com. (2018). "Genetic Algorithm". [online] Available at: https://www.mathworks.com/discovery/genetic-algorithm.html [Accessed 27 May 2018].

[67] Bhesdadiya, R., Pandya, M., Trivedi, I., Jangir, N., Jangir, P. and Kumar, A. (2016). Price penalty factors based approach for combined economic emission dispatch problem solution using Dragonfly Algorithm. *2016 International Conference on Energy Efficient Technologies for Sustainability (ICEETS)*. [online] Available at: https://ieeexplore.ieee.org/document/7583794 [Accessed 1 Mar. 2018].

[68] Guo, S., Dooner, M., Wang, J., Xu, H. and Lu, G. (2017). Adaptive engine optimisation using NSGA-II and MODA based on a sub-structured artificial neural network. *2017 23rd International Conference on Automation and Computing (ICAC)*. [online] Available at: https://ieeexplore.ieee.org/document/8082008 [Accessed 15 Feb. 2019].

[69] Guha, D., Roy, P. and Banerjee, S. (2018). Optimal tuning of 3 degree-of-freedom proportional-integral-derivative controller for hybrid distributed power system using dragonfly algorithm. *Computers & Electrical Engineering*, [online] 72, pp.137-153. Available at: https://www.sciencedirect.com/science/article/pii/S0045790618304609 [Accessed 19 Feb. 2019].

[70] Pathania, A., Mehta, S. and Rza, C. (2016). Economic load dispatch of wind thermal integrated system using dragonfly algorithm. *2016 7th India International Conference on Power Electronics (IICPE)*. [online] Available at: https://ieeexplore.ieee.org/document/8079422 [Accessed 14 Apr. 2018].

[71] Zhang, Y., Yao, F., Iu, H., Fernando, T. and Wong, K. (2013). Sequential quadratic programming particle swarm optimization for wind power system operations considering emissions. *Journal of Modern Power Systems and Clean Energy*, [online] 1(3), pp.231-240. Available at: https://link.springer.com/article/10.1007/s40565-013-0030-2 [Accessed 14 Apr. 2018].

[72] Suresh, V. and Sreejith, S. (2016). Generation dispatch of combined solar thermal systems using dragonfly algorithm. *Computing*, [online] 99(1), pp.59-80. Available at: https://link.springer.com/article/10.1007/s00607-016-0514-9 [Accessed 16 May 2018].

[73] Mafarja, M., Eleyan, D., Jaber, I., Hammouri, A. and Mirjalili, S. (2017). Binary Dragonfly Algorithm for Feature Selection. *2017 International Conference on New Trends in Computing Sciences (ICTCS)*. [online] Available at: https://ieeexplore.ieee.org/document/8250257 [Accessed 10 May 2018].

[74] Emary, E., Zawbaa, H. and Hassanien, A. (2016). Binary ant lion approaches for feature selection. *Neurocomputing*, [online] 213, pp.54-65. Available at: https://www.sciencedirect.com/science/article/pii/S0925231216307263 [Accessed 2 Jul. 2018].

[75] Hamdy, M., Nguyen, A. and Hensen, J. (2016). A performance comparison of multi-objective optimization algorithms for solving nearly-zero-energy-building design problems. *Energy and Buildings*, [online] 121, pp.57-71. Available at: https://www.sciencedirect.com/science/article/pii/S0378778816301724 [Accessed 1 Mar. 2018].

[76] Arulraj, R. and Kumarappan, N. (2018). Simultaneous Multiple DG and Capacitor Installation Using Dragonfly Algorithm for Loss Reduction and Loadability Improvement in Distribution System. *2018 International Conference on Power, Energy, Control and Transmission Systems (ICPECTS)*. [online] Available at: https://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=8501352&filter=issueId%20EQ%20%228521560%22&pageNumber=2 [Accessed 19 Feb. 2019].

[77] Saleh, A., Mohamed, A., Hemeida, A. and Ibrahim, A. (2018). Comparison of different optimization techniques for optimal allocation of multiple distribution generation. *2018 International Conference on Innovative Trends in Computer Engineering (ITCE)*. [online] Available at: https://ieeexplore.ieee.org/abstract/document/8316644/figures#figures [Accessed 23 Feb. 2019].

[78] Parmar, A. and Darji, P. (2018). Comparative Analysis of Optimum Capacity Allocation and Pricing in Power Market by Different Optimization Algorithms. *Advances in Intelligent Systems and Computing*, [online] pp.311-326. Available at: https://link.springer.com/chapter/10.1007/978-981-13-1595-4_25 [Accessed 23 Feb. 2019].

[79] Wongsinlatam, W. and Buchitchon, S. (2018). The Comparison between Dragonflies Algorithm and Fireflies Algorithm for Court Case Administration: A Mixed Integer Linear Programming. *Journal of Physics: Conference Series*, [online] 1061, p.012005. Available at: https://iopscience.iop.org/article/10.1088/1742-6596/1061/1/012005 [Accessed 23 Feb. 2019].

[80] Diab, A. and Rezk, H. (2018). 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*, [online] 43(1), pp.77-96. Available at: https://link.springer.com/article/10.1007/s40998-018-0071-7 [Accessed 24 Feb. 2019].

[81] Al-Madi, N., Faris, H. and Mirjalili, S. (2019). Binary multi-verse optimization algorithm for global optimization and discrete problems. *International Journal of Machine Learning and Cybernetics*. [online] Available at: https://www.springerprofessional.de/en/binary-multi-verse-optimization-algorithm-for-global-optimizatio/16442930 [Accessed 21 Sep. 2019].

[82] Moayedi, H., Abdullahi, M., Nguyen, H. and Rashid, A. (2019). Comparison of dragonfly algorithm and Harris hawks optimization evolutionary data mining techniques for the assessment of bearing capacity of footings over two-layer foundation soils. *Engineering with Computers*, [online] pp.1-11. Available at: https://link.springer.com/article/10.1007%2Fs00366-019-00834-w [Accessed 21 Sep. 2019].

[83] K. V. Price, N. H. Awad, M. Z. Ali, P. N. Suganthan. (2018). The 100-Digit Challenge: Problem Definitions and Evaluation Criteria for the 100-Digit Challenge Special Session and Competition on Single Objective Numerical Optimization. *Nanyang Technological University*, Singapore.

[84] Tejani, G., Savsani, V. and Patel, V. (2016). Adaptive symbiotic organisms search (SOS) algorithm for structural design optimization. *Journal of Computational Design and Engineering*, [online] 3(3), pp.226-249. Available at: https://www.scopus.com/record/display.uri?eid=2-s2.0-84991722784&origin=inward&txGid=94c67ebb12be33b0fb72098991bc1ffc [Accessed 24 Nov. 2019].

[85] Mohammed, H., Umar, S. and Rashid, T. (2019). A Systematic and Meta-Analysis Survey of Whale Optimization Algorithm. *Computational Intelligence and Neuroscience*, [online] 2019, pp.1-25. Available at: https://www.hindawi.com/journals/cin/2019/8718571/ [Accessed 14 Dec. 2019].

[86] Abdullah, J. and Ahmed, T. (2019). Fitness Dependent Optimizer: Inspired by the Bee Swarming Reproductive Process. *IEEE Access*, [online] 7, pp.43473-43486. Available at: https://ieeexplore.ieee.org/document/8672851 [Accessed 14 Dec. 2019].

[87] Rashid, T., Abbas, D. and Turel, Y. (2019). A multi hidden recurrent neural network with a modified grey wolf optimizer. *PLOS ONE*, [online] 14(3), p.e0213237. Available at: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0213237 [Accessed 14 Dec. 2019].