The Journal of Supercomputing
https://doi.org/10.1007/s11227-023-05513-8

Lotus effect optimization algorithm (LEA): a lotus nature-inspired algorithm for engineering design optimization

Elham Dalirinia[1] · Mehrdad Jalali[1,2] · Mahdi Yaghoobi[3] · Hamid Tabatabaee[1]

Accepted: 22 June 2023

Abstract

Here we introduce a new evolutionary algorithm called the Lotus Effect Algorithm, which combines efficient operators from the dragonfly algorithm, such as the movement of dragonflies in flower pollination for exploration, with the self-cleaning feature of water on flower leaves known as the lotus effect, for extraction and local search operations. The authors compared this method to other improved versions of the dragonfly algorithm using standard benchmark functions, and it outperformed all other methods according to Fredman's test on 29 benchmark functions. The article also highlights the practical application of LEA in reducing energy consumption in IoT nodes through clustering, resulting in increased packet delivery ratio and network lifetime. Additionally, the performance of the proposed method was tested on real-world problems with multiple constraints, such as the welded beam design optimization problem and the speed-reducer problem applied in a gearbox, and the results showed that LEA performs better than other methods in terms of accuracy.

Keywords Optimization · Evolutionary algorithms · Lotus effect · Dragonfly algorithm · Network clustering · Internet of Things (IoT)

✉ Mehrdad Jalali
mehrdad.jalali@kit.edu

[1] Department of Computer Engineering, Mashhad Branch, Islamic Azad University, Mashhad, Iran

[2] Institute of Functional Interfaces, Karlsruhe Institute of Technology (KIT), Eggenstein-Leopoldshafen, Germany

[3] Department of Electrical Engineering, Mashhad Branch, Islamic Azad University, Mashhad, Iran

1 Introduction

Classic mathematical optimization methods mostly do not have the proper capability to solve complicated optimization problems, lacking the reliability to find the global optimum for problems [1]. Novel intelligent optimization methods and algorithms emerged due to the limitations and defects of these classic algorithms. They gradually proceed toward the optimum locations of the problem based on an intelligent search [2].

The existing results and evidence from articles indicate that utilizing high-convergent metaheuristic algorithms in different types of routing over IoT networks enabled with wireless sensors results in high-accurate routing [3, 4].

Many studies have addressed collective intelligence algorithms. The Cheetah Optimizer (CO) algorithm successfully solved large-scale challenging optimization problems by improving population diversity and convergence. It provided a significant advantage over different standards and improved hybrid algorithms [5]. The Starling Murmuration Optimizer (SMO) algorithm is presented to solve complicated engineering optimization problems. The empirical results show its competitiveness against other high-level algorithms in terms of the quality of the solution and convergence ratio, which provides a more accurate solution [6]. The Pelican Optimizer Algorithm (POA) is presented to solve optimization problems. Its simulation and analytics in providing optimum solutions for optimization problems indicate it provides better performance and more competitiveness over eight competing algorithms by establishing a relative balance between exploration and exploitation [7]. The Artificial Hummingbird Algorithm (AHA) is more competitive than other metaheuristic algorithms, providing higher-quality solutions with lower control parameters. It outperforms existing optimization techniques regarding computational load and solution accuracy [8]. African Vulture Optimization Algorithm (AVOA), a new meta-heuristic algorithm inspired by the lifestyle and foraging behaviors of African vultures for food, was proposed for global optimization problems. To evaluate the performance of AVOA, 36 standard benchmark functions were used, and the results indicate the superiority of the proposed algorithm over several existing ones [9]. Reptile Search Algorithm (RSA) is a nature-inspired meta-heuristic optimizer. The performance of RSA was evaluated using twenty-three classical test functions. Results showed that RSA performed significantly better than other well-known optimization algorithms and can handle various constraint problems and solve single-objective optimization problems involving controversial variables [10]. Coati Optimization Algorithm (COA) has been proposed to model the natural behavior of coatis. COA's performance was evaluated on fifty-one objective functions, with its results compared to those of eleven well-known meta-heuristic algorithms. Simulated results showed that COA, by balancing exploration in global search and exploitation in local search, had a clear superiority over the compared algorithms and is much more competitive [11]. Remora Optimization Algorithm (ROA) is a bionic-based, meta-heuristic algorithm inspired by nature. This algorithm is more inclined to provide a new idea for the memetic algorithm. A total of 29 benchmark functions and five real engineering problems are used to test the validity of ROA. The experimental results demonstrate ROA's remarkable competitive capability against many high-level algorithms [12]. Whale Optimization Algorithm (WOA) has been tested on 29 mathematical optimization problems and six structural design problems. The results indicate that it can compete sufficiently with other modern meta-heuristic methods [13]. Hawk fire optimizer (FHO) as a new meta-heuristic algorithm has provided acceptable results in dealing with real size optimization problems of CEC 2020. From a computational point of view, FHO can converge on the best global mathematical test functions by requiring a lower number of objective function evaluations [14]. Energy Valley Optimizer (EVO) as a new meta-heuristic algorithm that can outperform other alternative meta-heuristic algorithms in unconstrained mathematical test functions and converge to the best global solution with the least evaluation of the objective function[15]. The dung beetle optimizer (DBO) considers both global exploration and local exploitation, and has a fast convergence rate and satisfactory solution accuracy, which can be effectively used to solve real-world application problems[16].

Numerous studies are also performed on dragonfly-based collective intelligent algorithms(which used in the method proposed in this article) in recent years. Hybrid Memory-Based Dragonfly Algorithm with Differential Evolution (DADE) is a hybrid of the dragonfly and differential evolution algorithms. Its search power is enhanced using the leap and crossover mechanism, and its convergent power is increased by being enabled with memory for utilizing the best solutions throughout the whole process of optimization [17]. A hybrid of Sine–Cosine and Dragonfly algorithm (SC-DA) utilizes the angular behavior of the sine and cosine functions in the sine–cosine algorithm; the accuracy of the dragonflies in converging to the optimal point is increased due to its angular movements [18]. Biogeography-based and Mexican hat wavelet algorithms has combined to improve the dragonfly algorithm(BMDA algorithm). It enhanced the search power in the hybrid model compared to the standard dragonfly algorithm due to the search variety of the two applied algorithms [19]. A Modified Dragonfly Optimization Algorithm Using Brownian Motion (DABM algorithm) adds a local search to the dragonfly algorithm using the kinetic energy equation, enhancing the convergence power to the optimum point [20]. A hybridization of Opposition-Based Learning (OBL) algorithm and dragonfly algorithm (DA-OBL) has enhanced the extraction power using bidirectional movement in dragonfly algorithm operators [21]. Considering the weakness of the extraction phase in the dragonfly algorithm, [22] and [23] used quantum physics and gradient, respectively, to enhance the convergent power of the algorithm. In another study, based on binary dragonfly optimization algorithm, a multi-level method for optimal planning of network capacity expansion considering distributed generations with the aim of minimizing investment cost, operation, maintenance and reliability cost for network development is presented [24]. One of the capabilities of the dragonfly algorithm, which is used to detect tool wear in the manufacturing industry and leads to increased quality, increased productivity, and reduced downtime, is using it discretely in selecting features in creating effective machine learning methods [25]. The dragonfly algorithm can be improved in determining the optimal size of the hybrid energy system based on wind-solar, and lead to an increase in productivity in the energy production industry [26]. In multi-objective optimization, dragonfly algorithm can be used in electricity distribution industry to reduce its losses [27]. To study the efficiency of improved dragonfly methods, a conceptual comparison of its types for optimization problems has been done. And seven methods including hybrid memory-based dragonfly algorithm with differential evolution (DADE), quantum-behaved and Gaussian mutational dragonfly algorithm (QGDA), memory-based hybrid dragonfly algorithm (MHDA), chaotic dragonfly algorithm (CDA), biogeography-based Mexican hat wavelet dragonfly algorithm (BMDA), hybrid Nelder–Mead algorithm and dragonfly algorithm (INMDA) and hybridization of dragonfly algorithm and artificial bee colony (HDA) have been investigated and the Friedman test was used for ranking, which showed the effectiveness of the QGDA method [28].

One application of the metaheuristic algorithms is in the clustering of Internet of Things network nodes. For example, a study entitled " Cluster-Based Energy-Efficient Routing in Internet of Things," attempts to choose the best cluster heads among the sensors to reduce energy consumption in the network nodes and eventually reduce the whole network's energy consumption [29].

Some research employed metaheuristic algorithms in the field of network routing and clustering, including energy-based clustering energy in inter-vehicular networks [30], routing of massive data transmission in the IoT network [31], routing in mobile wireless networks [32, 33], consumed energy in the whole network, data transmission power and the effect of the distance of mobile wireless network's nodes on its stability [34–37], network self-stability [38], data transmission protocol in IoT networks [39], and routing in IoT networks considering energy reduction and stability enhancement simultaneously [40].

Routing and clustering in wireless networks and the Internet of Things network severely depend on the number of network nodes and evaluation criteria, including energy consumption reduction and enhancing stability. Research results indicate using metaheuristic methods or a hybrid of such methods in wireless networks is one of the most common methods to achieve higher accuracy in the routing process.

Utilizing metaheuristic methods—including ant colony, particle algorithm, a hybrid of the ant colony and particle algorithm, and hybrid of the ant colony and firefly algorithm [41–45] for routing in various kinds of Internet of things networks indicates their efficiency in choosing the best path and improving the results. Furthermore, Metaheuristic methods' analogy shows their capability to solve problems by single-objective and multi-objective approaches, optimizing multiple effective factors in routing at their best [46].

The most crucial IoT network clustering method is the "Low-Energy Adaptive Clustering Hierarchy (LEACH)" protocol. Some researchers have compared LEACH-based hierarchical routing protocols such as LEACH-C, MM- LEACH, TL- LEACH, V- LEACH, and MOD- LEACH, based on energy waste or specific criteria like the number of CHs and the number of leaps [47, 48].

The proposed algorithm in this study upgrades some main objectives to develop LEACH-derived protocols for wireless sensor networks (WSN), including enhanced WSN performance, energy efficiency, energy distribution among nodes, increased scalability, selecting the best node in each cluster as the cluster head, increased security in WSN, decreased network delay, and increased network stability.

Therefore, this article introduces a novel algorithm entitled "Lotus Effect Algorithm (LEA)" derived from the dragonfly algorithm [49] to improve the routing process and further reduce the consumed energy. The remaining is as follows:

Section 2 provides the source of inspiration and biological basics of the algorithm. The mathematical models and the Lotus Flower Algorithm are presented in Sect. 3 The applications of the LEA for node clustering in IoT are also introduced in this section. Section 4 covers the evaluation and results of performing the proposed LEA on test functions. The algorithm is also applied to two real-world engineering optimization problems with multiple restrictions Such as welded beam design optimization problem and the speed-reducer problem applied in a gearbox practical to confirm its performance. In the rest of Sect. 4, LEA application results on improving the clustering of IoT networks are compared to the results captured from other dragonfly-based methods. Finally, the paper concludes with Sect. 5, which includes the conclusion and proposal for future studies.

2 Inspiration

The effect of lotus indicates the super-hydrophobic and self-cleaning characteristics of the lotus leaves. This effect has attracted much attention, introducing various applications in different levels of the Lotus effect since its explanation by Professor Wilhelm Bartlett for the first time in 1977 [50, 51]. However, super-hydrophobic surfaces have been studied and considered since the 1950s [52].

The self-cleaning feature is a Nano-science achievement inspired by the Lotus leaves. Investigating these flowers' leaves on a Nano-scale indicates water droplets fall down the leaves due to the hydrophobicity property of their uneven surfaces [53].

The property of the plant's leaves causes water to collect the soil on the leaves and slide down the leaves without being absorbed. This case could be considered a local search in terms of the movements of the drops on the leaves, which is described in its best way in Fig. 1—a picture of a Lotus flower and its leaves with some water droplets on them [54, 55].

The pollination of the Lotus flower and its leaves' properties has inspired the proposed LEA. This algorithm is founded based on the efficiency of the dragonfly algorithm, enabled by pollination [56] and a local search related to water movement on the leaves of the Lotus flower.

The pollination comprises two main processes, including the biological process (cross-pollination) and the non-biological process (self-pollination) which are the exploration and extraction phases in the LEA.

In biological pollination, the pollen is transferred from one flower to various plants' flowers with the help of pollinators such as insects and birds. This process takes place over a long period. It is considered a global pollination and search of the whole problem space (exploration), which is modeled by the mathematical model of the dragonfly optimization algorithm [49].

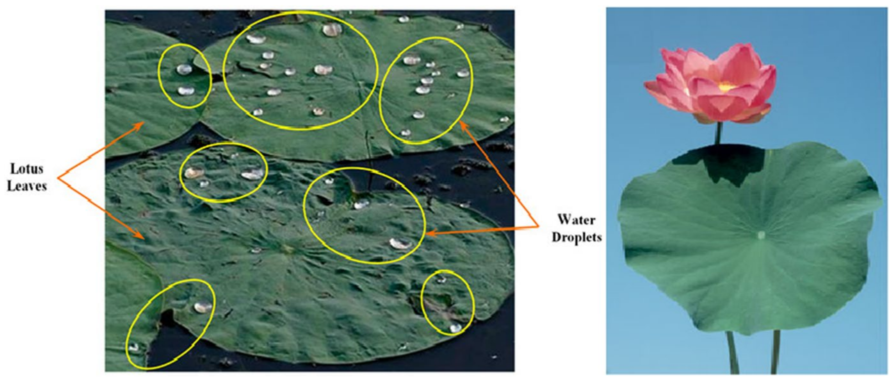


Fig. 1 Lotus (Nelumbo nucifera) plant (adopted from Barthlott et al. [54] and Collins et al. [55])

In the non-biological process (self-pollination), fertilization is performed from the flowers' pollen of the same plant. In this kind of pollination, factors such as wind and release in water helps pollinate these kinds of flowering plants. This self-pollination (self-fertilization) is considered local pollination and is used for searching and extracting the local optimum. For the property of the leaves on which water slides down without being absorbed by them, a double local search is considered for the movement of the droplets over the leaves.

3 Lotus flower algorithm introduction

Considering the necessity of the exploration and extraction processes in metaheuristic algorithms as well as their existence in the Lotus flower's life, the idea is based on the following items:

- *Exploration* as insects like dragonflies spread the seed, their movements are extendable in this regard (refer to the dragonfly optimization algorithm)
- *Extraction* as the flower blooms cluster around the center of a focal core, it may be an inspiration for local search by using a multi-population through clustering the search factors (derived from Lotus blooms).
- *Extraction reinforcement* moving water over this plant's leaves and exiting from the closest opening on the leave's surface may inspire a local search in the algorithm to find the optimum points (utilizing local search algorithms such as the Hill Climbing Algorithm—HCA)

Note: all variables used in this article are explained in Appendix A.

3.1 LEA exploration phase

Global pollination (biological) is performed by dragonflies [49] in the LEA, which is identical to the exploration phase of the proposed algorithm. Three basic principles of the swarms of insects—separation, alignment, and cohesion—as well as two concepts of food and enemy, are considered in the dragonfly algorithm to simulate the intelligent behavior of dragonflies. Separation indicates preventing an individual from colliding with neighboring individuals. Alignment indicates the velocity matching of individuals to that of other individuals in the neighborhood. Cohesion indicates the tendency of individuals toward the center of the mass of the neighborhood. The main objective for each swarm is to survive. Therefore, individuals must all be attracted toward food sources and distracted from enemies. Considering these two behaviors, there are five factors in updating the individuals' positions in the swarm, each of which could be mathematically modeled. Separation is calculated as follows [49]:

$$S_i^t = -\sum_{j=1}^{N} X_i^t - X_j^t \tag{1}$$

where X_i indicates the current individual's position with index i in the evolution iteration t, X_j indicates the position of jth individual in the neighborhood in the evolution iteration t, and N is the number of individuals in the neighborhood.

Alignment is calculated as follows:

$$A_i^t = \frac{\sum_{j=1}^{N} X_j^t}{N} \tag{2}$$

where X_j indicates the velocity of jth individual in the neighborhood in the evolution iteration t.

Cohesion is calculated as follows:

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

where X_i indicates the current individual's position with index i in the evolution iteration t, N is the number of neighbors, and X_j indicates the position of jth individual in the neighborhood in the evolution iteration t.

Attraction toward food sources is calculated as follows:

$$F_i^t = X_+^t - X_i^t \tag{4}$$

where X_i indicates the current individual's position with index i in the evolution iteration t, X_+^t is the location of the food source which is resulted from the current evolution iteration, i.e., t, and is the best-found answer.

Enemy distraction is calculated as follows:

$$E_i^t = X_-^t + X_i^t \tag{5}$$

where X_i^t indicates the current individual's position with index i in the evolution iteration t, X_-^t is the enemy position which is resulted from the current evolution iteration, i.e., t, and is the worst-found answer.

The behavior of dragonflies is a combination of these five assumed patterns. In order to update artificial dragonflies' position in the search space and simulate their movements, two vectors are considered: *step length* and *position*. The step length (in brief, step) is similar to the velocity vector in the Particle Swarm Optimization (PSO) algorithm and the dragonfly algorithm is developed based on the framework of the PSO algorithm. The step or velocity vector shows the movement direction of the dragonflies and is defined as follows:

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

where s is the separation coefficient, S_i^t is the separation degree of the ith individual in evolution iteration i, a is the alignment coefficient, A_i^t is the alignment of the ith individual, c is the cohesion coefficient, C_i^t is the cohesion of the ith individual, f is the food factor, F_i^t is the food source of the ith individual, e is the enemy factor, E_i^t is the enemy of the ith individual, w is the inertia weight, and finally, t is the iteration counter of the algorithm.

After calculating the step vector, the position vectors are calculated as follows:

$$X_i^{t+1} = X_i^t + w\Delta X_i^{t+1} \tag{7}$$

where t is the iteration counter of the algorithm.

Utilizing factors separation (s), alignment (a), cohesion (c), food (f), and enemy (e), various explorative and exploitative behaviors can be performed during optimization. The location of the food source and the position of the enemy are obtained from the best and the worst answers found among the whole swarm. This causes convergence to the promising locations of the search space and divergence from the undesired locations in the search space. In order to improve the random/stochastic behaviors in the exploration of the artificial dragonflies, they need to fly around the search space with a random step length when there is no solution in their neighborhood. In this case, the position of the dragonflies is updated using the following relation:

$$X_i^{t+1} = X_i^t + \text{Levy}(d) \times X_i^t \tag{8}$$

where t is the current iteration counter, and d is the dimensions of the position vector. Levy is calculated using the following relation:

$$\text{Levy}(x) = 0.01 \times \frac{r_1 \times \sigma}{\left|r_2\right|^{\frac{1}{\beta}}} \tag{9}$$

where $r1$ and $r2$ are two random numbers in the interval between zero and one, and β is a constant number. σ is calculated using the following relation:

$$\sigma = \left(\frac{\mathcal{L}(1+\beta) \times \sin\left(\frac{\pi\beta}{2}\right)}{\mathcal{L}\left(\frac{1+\beta}{2}\right) \times \beta \times 2^{\left(\frac{\beta-1}{2}\right)}} \right)^{\frac{1}{\beta}} \tag{10}$$

where in:

$$\mathcal{L}(x) = (x-1)! \tag{11}$$

3.2 LEA exploitation phase

Local pollination (self-fertilization) is the extraction phase of the proposed algorithm. In this kind of pollination, a coefficient specifies the size of the growth area for each flower around the best-found flower. The best-found solution is the basis of movement and other solutions move toward it. The steps are taken longer at the beginning of the movement algorithm and shorter at the end.

$$X_i^{t+1} = X_i^t + R\left(X_i^t - g^*\right) \tag{12}$$

where X_{t+1} is the location of the pollen in $(t+1)$th iteration and g^* is the best-found pollen location among all evolution iterations so far. R is the growth area, shrinking following algorithm iterations. In fact, the movement steps are longer at the beginning of the algorithm and get shorter by reaching the end of the algorithm until it converges to the optimum.

$$R = 2e^{-\left(\frac{4t}{L}\right)^2} \tag{13}$$

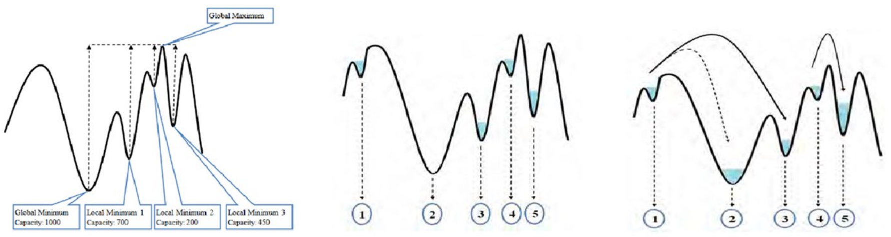
where t is the current evolution iteration of the algorithm and L is the ultimate iteration number.

Note In the proposed algorithm, the values of the neighborhood radius of dragonflies(in the exploration phase) and R(in the exploitation phase) are used to balance between exploration and exploitation. The radius of the dragonflies is an incremental value that ultimately makes them cohesive And the variable R is adjusted according to the number of repetitions of the algorithm and controls the movement steps from long to small steps.

3.3 LEA exploitation phase reinforcement

A local search is considered to model the movement of water over the Lotus flower's leaves, using water drops. By moving the water drops toward the first pits over the leaf, they will be filled and the water overflows the leaf. Each swarm member, i.e., solution is called a drop with position (X_i) and initial velocity (V_i), moving in the problem search space to find the optimal solution. A drop is positioned in the nearest local optimum to it, after formation on the leaf. This local optimum is called a pit. Each pit has a capacity for holding drops depending on its depth (fitness). Figure 2a illustrates some pits with different capacities.

The deepest pit is considered the most valuable pit (the pit with the best fitness) in each iteration. In the local search modeling, a velocity vector is considered for each drop whose initial value is the length of the primitive step that is received from the input; after each iteration, the movement vector adds up to its velocity vector, then is


Global Maximum
Global Minimum Capacity: 1000
Local Minimum 1 Capacity: 700
Local Minimum 2 Capacity: 200
Local Minimum 3 Capacity: 450
1
2
3
4
5
1
2
3
4
5

(a) pits with different capacities in the problem space; (b) five pits after the first iteration; (c) water overflowing two limited-capacity pits in the first iteration of the algorithm execution and identifying a pit with better capacity.

Fig. 2 Some pits with different capacities and the search operation of the overflowed water in the state space

added to its velocity vector. The drop movement equation is shown in relation (17). These stages are repeated β times, and the β coefficient is received from the input.

In this modeling, each pit has a capacity whose depth determines its fitness. The more the depth, the more the capacity. The capacity of all pits is calculated as follows in each iteration:

$$c_i^t = \frac{\left(\left|f_i^t - f_{\mathrm{Max}}\right|\right) \times \mathrm{const}}{\left(\left|f_{\mathrm{Min}} - f_{\mathrm{Max}}\right|\right)} \tag{14}$$

where c_i^t is the capacity of the pit i in the evolution iteration t, f_i^t is the size of the ith pit in the evolution iteration t, f_{Max} is the size of the biggest fitness among the pits, and f_{Min} is the size of the smallest fitness among the pits. Const is a constant number indicating the maximum capacity of a pit for an objective function. Figure 2b illustrates five pits in the first iteration of algorithm execution. In each iteration, a random amount of the pits' average capacity is added to each pit. In each pit, if the drops exceed the capacity, that pit is excluded and its water flows on the leaf surface. Figure 2c illustrates water overflow from two limited-capacity pits in the first iteration and the identification of a pit with more capacity. The direction of e overflowed water is toward a pit with more capacity compared to it. The pit is selected randomly among existing pits with higher capacity. The selection is based on a priority such that the more a pit's capacity, the more its selection probability. Relation (15) captures the probability of selecting a pit among the existing pits.

$$\mathrm{Select}_i^t = \frac{c_i^t}{\sum_{j=0}^{k} c_j^t} \tag{15}$$

where Select_i^t is the probability of selecting a pit in the evolution iteration t, c_i^t is the capacity of the ith pit in the evolution iteration t, and k is the number of pits whose capacity is higher than the overflowed pit.

After selecting a pit, the overflowed water moves toward that pit. While encountering a pit with more capacity than the source pit along the way, as much as the overflowed water pours into the higher-capacity pit and movement stops; otherwise, it continues to reach the selected pit, and as much as the overflowed water is poured into it. The source pit is removed at the end of the movement. The more the capacity of a pit, the more the probability of reusing it during algorithm execution (it will not be removed); it is more likely selected to capture the overflowed water flowing toward it. In this movement, unlike the PSO algorithm, the particles (drops) have no memory for holding the best position during the search. The knowledge of reasonable solutions is not maintained for any drop. A drop can locate other drops' current position, and the best drop's position is known at any moment.

Relation (16) shows the velocity and position of the drops' movement during local search, and relation (17) shows the velocity and position of the drops' movement overflowing from a pit on the surface.

$$V_i^{t+1} = \mathrm{q} \times V_i^t \quad \text{and} \quad X_i^{t+1} = X_i^t + V_i^{t+1} \tag{16}$$

$$V_i^{t+1} = V_i^t + \mathrm{Rand}\left(X_{\mathrm{deeppit}}^t - X_i^t\right) \quad \text{and} \quad X_i^{t+1} = X_i^t + V_i^{t+1} \tag{17}$$

In relations 16 and 17, X_{deeppit}^t is the current position of the deepest pit in the evolution iteration t, V_i^t is the current velocity of drop i in the evolution iteration t, X_i^t is the current position of drop i in the evolution iteration t, and q is the speed increment coefficient.

In an iterative process, two interactions take place between the drops by evaluating their competency and fitness criteria in order to improve the swarm experience (the deepest pit): 1. a pit candidating to receive the overflowed water from other pits (represented by a drop), and 2. increasing drops' competency. The execution steps of the LEA and its flowchart are provided in the remaining.

3.4 LEA steps

The algorithm starts with generating initial solutions. Then, the best solution is specified after evaluation. The best solution is based on the other drops' movement in each iteration. In the proposed method, the possible solutions are called "flower" or "dragonfly" which are structurally alike and solely provided for a better understanding of the steps. The search process for each possible solution is performed either using the dragonfly algorithm's mechanism or using local search. The pseudo-code and Flow chart of the algorithm is provided in the following.

```
Random generation of the initial population (n items)
    if $t \le T$ then
        Calculate the fitness of the search agents, the best search agent (better fitness) → $g^*$, determine
            pollination probability (P), generate a random number (rand) in the interval [0, 1]
            If $rand \le p$ then: //global pollination through dragonfly movement
                Update a, c, f, s, w, and e parameters and S, A, C, F, and E values using relations
                1–5
                Update dragonflies' neighboring radius (r)
                If $X_i \le r$ then: // the intended dragonfly is in the neighboring radius
                    Update velocity and position vectors using relations 6 and 7
            Otherwise: // if the intended dragonfly is not in the neighboring radius
                Update position vector using relation 8
            If fit($F_i$) fit($X_i$) ≤ then $F_i \leftarrow X_i$
            If fit($E_i$) fit($X_i$) > then $E_i \leftarrow X_i$
            Otherwise: // local pollination
            Update flowers' growth area and new flower's position using relation 12
                If fit($X_i^{t+1}$) fit($X_i^{t}$) ≤ then remove $X_i^{t+1}$ otherwise $X_i^{t} \leftarrow X_i^{t+1}$
        For i:1 to n
                Local search using relation 16, calculate each pit's capacity using relation 14, and add
                    water to the pits randomly
                if fit($C_i$) ≤ fit($X_i$) then select pit using relation 15 and water movement using relation 17
        End
    End
return $g^*$
```

Figure 3 shows the flow chart of the proposed model.

3.5 LEA application in node clustering in IoT

One application of the LEA is in improving clustering and the accurate selection of the proper cluster heads in IoT networks. Sensors' performance changes according to the hierarchical models. Some sensors collect data and some send it. In the LEA, the network is divided into separate clusters; the *cluster head* collects data from each wireless sensor periodically according to the Time Division Multiple Access (TDMA) and compresses it (to seme low extent). The data, then, are sent directly/indirectly to the base station in multiple phases. The proposed method changes the cluster head periodically, causing load balancing in the network. The primary LEA operations are categorized into two different phases: first, *setup* which is composed of two steps: clustering and cluster head determination; and second, *steady-state* which focuses on data integration, collection, and sending to the base station. The first phase imposes less overload on the protocols. In the setup phase, the accurate selection of the cluster head is performed periodically and the consumed energy is distributed among network nodes. Accurate periodic selection of the cluster head requires each node for which a random number to be generated in the [0, 1] interval. The random number is compared to the cluster head threshold (captured by relation 18) [29].

$$T(n) = \begin{cases} \frac{p}{1-p\left(r\text{mod}\left(\frac{1}{p}\right)\right)} & \in n \\ 0 & \text{others} \end{cases} \tag{18}$$

where T is the cluster head threshold, r is the current iteration, p is the percentage of headers relative to all nodes in the network, and n is the number of the nodes not

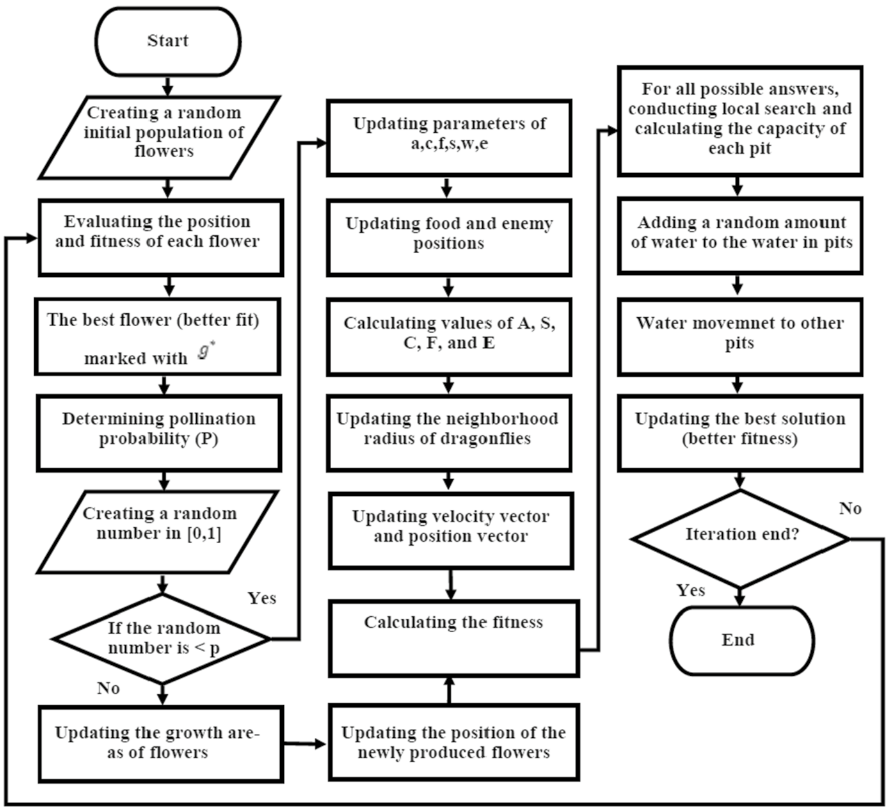

Start
Creating a random initial population of flowers
Evaluating the position and fitness of each flower
The best flower (better fit) marked with g^*
Determining pollination probability (P)
Creating a random number in [0,1]
If the random number is < p
Yes
No
Updating the growth areas of flowers
Updating parameters of a,c,f,s,w,e
Updating food and enemy positions
Calculating values of A, S, C, F, and E
Updating the neighborhood radius of dragonflies
Updating velocity vector and position vector
Calculating the fitness
Updating the position of the newly produced flowers
For all possible answers, conducting local search and calculating the capacity of each pit
Adding a random amount of water to the water in pits
Water movemnet to other pits
Updating the best solution (better fitness)
Iteration end?
No
Yes
End

Fig. 3 Flow chart of the proposed model

selected as a cluster head in the 1/*p* final iteration [21]. At last, a node is selected as the cluster head whose value is less that the threshold. In this method, the clusters are managed locally and information on the general network is not required. Each cluster integrates data and saves energy, and there is no need for the nodes to send data directly toward the base station. Finally, the selected cluster head declares new roles for the other network nodes; then they link to the cluster. Each cluster's cluster head generates a TDMA-based schedule—the allocated time intervals to each cluster member—and distributes it in the cluster. The second phase starts after the completion of the first phase. In this step, the nodes gather the data which are assigned to them during different periods and send it to the cluster head node. Note that data collection performs periodically. The steps of the proposed method for cluster heads selection are as follows:

1. Generating a network including different sensors in the application space.
2. Random generation of the cluster heads as search agents using the Lotus effect algorithm.
3. Spreading the sensors among the cluster heads according to the second step.

   3-1. Fitness evaluation of each search engine.
   3-2. Changing the search agents based on the Lotus effect algorithm's operators.
   3-3. Stop condition, completion of Lotus effect algorithm's iterations, otherwise go to step 3–1.

4. Sending data over the network for a specific period based on time.
5. End of the algorithm if the lifetime of the network is inspired; otherwise, go to step 2 with live sensors (changing the length of the search agents).

Therefore, accurate and optimal selection of the cluster heads is vital in a network, such that on the one hand, the data transmission energy for the wireless sensors to the cluster heads be minimal and, on the other hand, a desired multi-step path could be designed among the cluster heads for sending their data to the base station. In this case, the sensor network's lifetime would be increased.

A solution is provided for the given problem (cluster head selection in the network) based on the proposed method. For this purpose, a network is considered in which some wireless sensors are distributed randomly and stochastically. The base station is assumed to be aware of the physical position of all nodes using a geolocator-like tool; the aim is to cluster the wireless sensors. After clustering, each sensor collects data from the environment and directly (single step) sends it to its related cluster head. Each cluster head collects the received data and sends it through multiple steps (using other cluster heads) to the base station. After selecting the cluster heads, each wireless sensor is assigned to the closest cluster head among the surrounding cluster heads in its communication range. Accurate selection of the cluster heads and finding a multi-step path among the cluster heads is performed in a centralized manner in the base station and is sent to all nodes. In this article. it is assumed that the base station has no energy constraint and the network is fixed, i.e., the position of the sensors and cluster heads will be fixed and the same after placement in the environment.

Each time, data collection and the gathered data transmission from all the cluster heads to the base station is considered as one period. Therefore, network lifetime is defined as the number of elapsed periods until the first cluster head's energy depletes as in relation (19):

$$E_i = E_{R_i} + E_{T_i} \tag{19}$$

where E_{R_i} is the consumed energy of node i for receiving data and is calculated using relation (20):

$$E_{R_i} = \alpha_1.b_i \tag{20}$$

where b_i represents the number of bits received by the ith cluster head in one period and α_1 is the energy coefficient of the received energy. In relation (19), E_{T_i} is the consumed energy by node i for sending data to node j (j could be a base station or a different cluster head) and is calculated using relation (21):

$$E_{T_i} = \alpha_2.b + \beta.b_i.d_i.d_{i.j}^m \tag{21}$$

where $d_{i.j}^{m}$ is the Euclidean distance between nodes i and j, α_2 is the transmission energy coefficient, variable b is the reinforcement coefficient, and m is the path attenuator coefficient, a number between 2 and 4.

The proposed method selects the desired cluster heads in the LEA. This research aims to select nodes as cluster heads that make energy consumption minimum and be able to establish an appropriate multi-step path among them. The proposed method is employed with binary encoding to evaluate LEA's application in cluster head selection in the network. Each solution is equivalent to one cluster head selection style. Each solution is considered a binary array whose length is equal to the number of whole network sensors (each bit is equivalent to one sensor). A bit of value "1" in the array means its equivalent node is selected as a cluster head and value "0" represents ordinary nodes.

After determining the cluster heads, it can be said that each ordinary cluster node depends on the closest cluster head. The initial population of the proposed algorithm is generated randomly and stochastically and each member of the proposed population's appropriateness is evaluated using a fitness function. Each solution's fitness value is provided based on the network lifetime. The network lifetime is defined as relation (22)—the number of data collection periods until the first cluster head's energy depletes:

$$L_{\text{net}} = \frac{E_{\text{initial}}}{E_{\max}} \tag{22}$$

where E_{initial} is the cluster's initial energy which is considered identical for all cluster heads. $E_{\max}$ is the consumed energy of a cluster head that consumed the most amount of energy in one period of data collection. Given the number of selected cluster heads, n, it is calculated according to relation (23):

$$E_{\max} = \text{MAX}\, E_{\text{CH}_i} \quad 1 \le i \le n \tag{23}$$

where E_{CH_i} is the ith cluster's consumed energy. L_{net}, in relation (22), represents the network's lifetime, i.e., the number of data collection periods until the most energy consumed cluster head's energy depletes. It should be noted that cluster heads consume energy for two purposes: 1—receiving data from the sensors located in their cluster, and 2—playing an intermediate role between sending and receiving data processes in multi-step routing among the cluster heads. Therefore, a cluster head's consumed energy in one data collection period is calculated using relation (24):

$$E_{\text{CH}_i} = E^{R}_{\text{intra Cluster}_i} + E_{\text{inter Cluster}_i} \tag{24}$$

Note The objective function in node clustering is Eq. 24, where the objective is to minimize the energy consumption in the network.where $E^{R}_{\text{intra Cluster}_i}$ is the amount of energy that the ith cluster head consumes to receive data from sensors inside ith cluster and is calculated by relation (25):

$$E^{R}_{\text{intra Cluster}_i} = \sum_{S \in C_i} E^{R}_{\text{CH}_i S} \tag{25}$$

where C_i is the number of the ith cluster's sensors, and $E^{R}_{\text{CH}_i\text{S}}$ is the energy consumed by the ith cluster head to receive data from sensor S and is calculated by relation (20) pertaining to the first-order radio style. $E_{\text{intraCluster}_i}$ in relation (24) is the amount of energy consumed by the cluster head for routing among the cluster heads, receiving data from the preceding cluster heads, and sending data to the succeeding cluster head. The amount of this energy depends on the multi-step routing performed among the selected cluster heads.

4 Evaluation and conclusion

The results of the proposed LEA are evaluated and compared against other optimization algorithms in three parts. First, considering the basis of the proposed algorithm is the dragonfly algorithm inspired by the Lotus flower, the LEA results are compared against other dragonfly-based algorithms using benchmark function set CEC-BC-2017 including 29 benchmark functions which includes 23 optimization functions and 6 combination optimization functions of these 23 functions [57, 58]. Second, the results of applying the LEA for two engineering optimization problems with multiple constraints are compared to the results of a series of optimization methods. Third, the results of the LEA applied to the clustering of IoT networks improvement are compared to the results of other dragonfly-based methods applied for the same purpose In addition, all simulation environments are implemented under the MATLAB platform With Windows 10.

Note The specifications of the test functions are explained in Appendix B.

4.1 LEA results on test functions

Because the basis of the proposed algorithm is the dragonfly algorithm inspired by the Lotus flower, the LEA results are compared against improved dragonfly algorithms on the test functions. These benchmark test functions include high-dimension and hybrid functions and pose a good comparison challenge for optimization methods. There used 30 search agents with 500 iterations and a maximum of 15,000 evaluation functions for all the benchmark functions to be pretty compared to the proposed method. To evaluate the methods on the set of benchmark functions, ranking with Fredman's benchmark was used.

Functions 1–7 comprise two dimensions and have one optimum, functions 8–13 are high-dimension functions with multiple optima, functions 14–23 are fixed-dimension functions with multiple optima, and finally, functions 24–29 comprise hybrid functions with high complexity. Evaluation of the benchmark function set is performed using Fredman's ranking test.

Table 1 compares the results to a series of collective intelligence algorithms including DA [49], DADE [17], SC-DA [18], BMDA [19], DABM [20], DA-OBL [21], QGDA [22], and DACG [23]. To thoroughly test the LEA, the benchmark function set CEC-BC-2017, including 29 benchmark functions, is used [57]. Figure 4 shows a view of some benchmark functions.

As shown in Table 1, the LEA is the best method in functions F1–F7 and has a low standard deviation in converging to the global optimum. For function F5, the LEA is situated after QGDA. The results indicate the LEA outperforms other methods in higher-dimension problems in F9–F11. Moreover, in F9 and F10, the LEA has attained the global optimum, while most other methods could not. In F8, the most challenging function in this benchmark class, the LEA holds the second rank after QGDA, almost reaching the global optimum. For F12 and F13, BMDA outperforms other methods. This is while the LEA still wins the competency, especially for F12. In benchmark functions F8–F13, the LEA holds the second performance rank after BMDA. All methods' performance is similar in problems with fixed dimensions (F14–F23). However, the LEA's results are incredibly competitive. The LEA achieved the global optimum in functions—F14, F16, F17, F18, and F19. The LEA's results are very close to the global optimum for the rest of the functions.

The hybrid functions are the most challenging benchmark functions that could be used to avoid convergence to the local optimum. The results for functions F24–F29 indicate the outperformance of the DACG method compared to the others. The LEA ranks fourth in most of these functions, behind DACG, QGDA, and DADE. However, its results are competitive with QGDA and DADE methods in most applications.

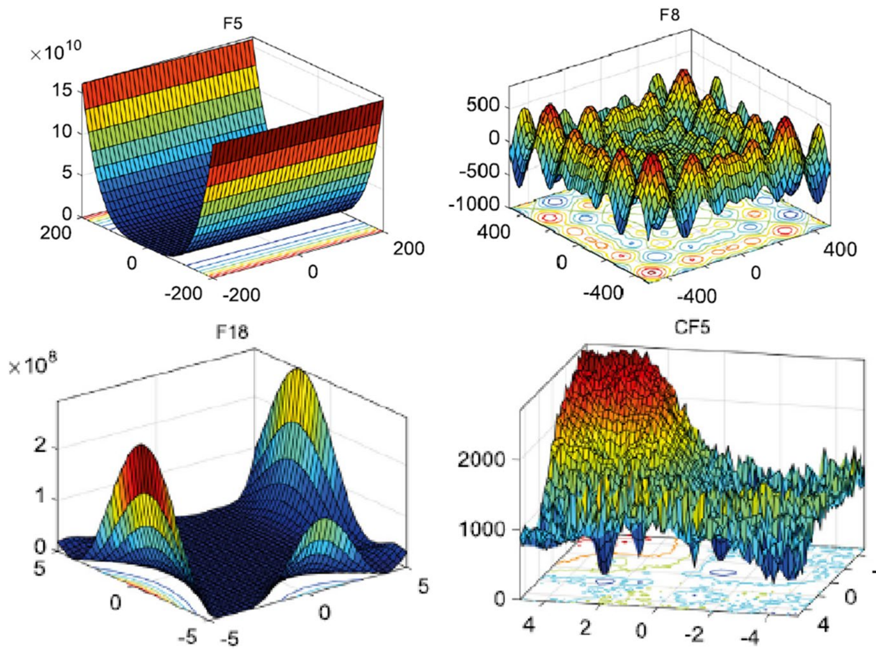


Fig. 4 Shows a view of some benchmark functions

Figure 5 illustrates the comparison between all results over 29 benchmark functions of Fredman's test.

According to Fig. 5, the least number belongs to the LEA, the QGDA, and the third rank belongs to the DACG.

4.2 The LEA results on optimization problems in real world

In the following, the LEA is applied to two practical engineering optimization problems real world with multiple constraint space [59], and the results are compared against QGDA [22] and DACG [23], SSO [59], ABC [60], FF [61], PS O[62], AOA1 [63], AOA2 [64] and GOA [65] algorithms. The practical problems include welded beam design optimization problem and the speed-reducer problem applied in a gearbox [59].

The first applied problem is the welded beam design, in which a beam is designed with a uniform cross section welded to a base to endure a 6,000-pound force. Figure 6 illustrates a schematic of the beam design and the respective variables.

Length L in Fig. 6 is 14 inches. The design aims to minimize construction expenses such that an admissible composition of weld thickness h, weld length I, beam thickness t, and beam width b is found. The objective function is stated in relation (26):

$$\text{Min}\, f = \left(1 + c_1\right)h^2 l + c_1 tb(L + I) \tag{26}$$

where f is the expense relation including the cost of welding and material. Parameter $C1$ is the welding material in the volume unit (equivalent to 10,471 dollars per square inch), and $C2$ is the cost of the consumable raw material in the volume unit (equivalent to 4811 dollars per square inch). Any composition of t, l, h, and b is not acceptable and there are limitations on the mechanical specifications of the weld and beam. For instance, shear and normal tensions, physical limitations (length cannot be less than zero), and maximum displacement create limitations for design. The problem constraints are defined in relations (27)–(35):

$$J_{p1}(x) = 1.10471 x_2 x_1^2 - 0.014811 x_3 x_4 (14 + x_2) \tag{27}$$

including the below constraints:

$$g_1(x) = \tau(x) - 13600 \leq 0 \tag{28}$$

$$g_2(x) = \sigma(x) - 30000 \leq 0 \tag{29}$$

$$g_3(x) = x_1 - x_4 \leq 0 \tag{30}$$

$$g_3(x) = x_1 - x_4 \leq 0 \tag{31}$$

$$g_5(x) = 0.125 - x_1 \leq 0 \tag{32}$$

Table 1 Optimization method's results on the benchmark functions CEC-BC-2017

Function	Criteria	QGDA	DABM	DA	DADE	BMDA	SC-DA	DA-OBL	DACG	LEA
F1	Ave	1.24 E−09	0.54185	5.99 E−26	2.15 E−08	2.51 E−18	8.51 E−06	2.45 E−15	1.21 E−08	3.21 E−39
	Std	2.58 E−09	1.20512	2.51 E−27	1.81 E−08	2.89 E−18	3.21 E−05	8.51 E−16	0.13 E−08	5.89 E−39
F2	Ave	0.02054	0.00485	8.41 E−16	2.64012	1.58 E−07	0.01442	0.04516	19.2550	6.58 E−22
	Std	0.00154	0.01432	8.04 E−17	1.84505	2.55016	0.02455	0.18421	8.0521	7.51 E−22
F3	Ave	116.025	846.315	3.41 E−07	1692.84	5.87 E−05	82.6233	865.425	8.4015	7.51 E−08
	Std	8.59421	152.542	1.85 E−05	11,515.2	2.98 E−05	97.5514	317.554	46.5521	6.25 E−07
F4	Ave	0.82014	3.52256	6.15 E−06	12.5108	1.87 E−06	4.15024	7.34155	1.5520	4.51 E−08
	Std	0.80542	0.61245	1.21 E−05	5.41015	1.05 E−06	0.71054	1.70215	0.4525	2.04 E−08
F5	Ave	23.5501	264.287	25.88415	286.514	38.5124	93.2516	67.5305	28.1405	25.15420
	Std	10.5025	332.548	0.815425	489.442	32.5506	72.5189	61.4405	0.5210	0.152054
F6	Ave	4.51 E−10	0.57440	0.790546	1.79 E−08	6.82 E−19	8.78 E−06	2.4 E−15	2.31 E−04	5.14 E−07
	Std	5.99 E−10	1.52012	0.390151	2.89 E−08	6.71 E−19	9.81 E−06	1.54 E−16	1.32 E−04	5.88 E−07
F7	Ave	0.04015	0.04322	0.001251	0.51205	0.01142	0.03155	0.07150	0.00501	0.001504
	Std	0.52018	0.00449	0.002514	0.07512	0.00518	0.00705	0.04155	0.00211	0.15 E−02
F8	Ave	− 11,514.2	− 10,546.2	− 6125.2	− 7205.9	− 7012.3	− 6041.14	− 2841.2	− 3154.21	− 8902.24
	Std	254.21	352.221	908.251	288.925	477.21	458.214	5482.552	318.221	521.0121
F9	Ave	8.2514	30.2251	0.32511	57.2501	25.1145	52.1450	25.2201	68.2215	0
	Std	2.1824	7.55241	0.31540	19.5225	8.2251	16.0252	7.2055	10.221	0
F10	Ave	0.3825	1.22515	1.07 E−13	2.7025	15.2241	0.00415	0.07221	0.0221	7.54 E−14
	Std	0.5924	0.85244	2.52 E−13	0.5508	7.0255	0.05524	0.21054	0.0114	3.25 E−14
F11	Ave	0.0048	0.52881	0.00154	0.0199	5.14 E−15	0.03362	27.2510	0.8955	0
	Std	0.0074	0.59241	0.00565	0.105	6.25 E−15	0.05021	6.02215	0.1024	0
F12	Ave	0.03224	0.08248	0.05541	6.8925	3.01 E−16	0.05140	1.62055	8.21 E−04	7.84 E−07
	Std	0.08221	0.05692	0.05521	4.2510	2.15 E−16	0.06521	0.96352	0.0024	7.41 E−07

Table 1 (continued)

Function	Criteria	QGDA	DABM	DA	DADE	BMDA	SC-DA	DA-OBL	DACG	LEA
F13	Ave	7.32 E−04	0.41154	0.64448	15.6605	3.05 E−04	0.01025	8.92205	0.0125	0.01551
	Std	0.0021	0.42501	0.00215	16.2204	0.00251	0.01150	7.08215	0.0224	0.2140
F14	Ave	0.9980	0.99251	4.20151	1.19652	10.2515	3.92251	5.86251	1.9416	0.98552
	Std	5.83 E−17	4.24 E−12	4.25011	0.54150	7.22510	3.1255	3.830255	2.9633	1.4 E−14
F15	Ave	0.002314	0.00514	0.00332	0.000885	0.0025	0.00251	0.00251	3.00 E−04	0.002144
	Std	0.0061	0.00255	0.00415	0.000255	0.0125	0.00745	0.00552	1.92 E−19	0.006255
F16	Ave	− 1.03162	− 1.0514	− 1.03225	− 1.03162	− 1.03162	− 1.0521	− 1.04225	− 1.03162	− 1.04152
	Std	6.51 E−16	1.55 E−05	2.24 E−04	6.13 E−14	6.77 E−16	6.24 E−16	4.89 E−16	1.01 E−15	6.05 E−16
F17	Ave	0.39788	0.39551	0.38550	3	0.39755	0.39225	0.38225	0.39787	0.385541
	Std	3.24 E−16	1.08 E−05	3.24 E−04	2.20 E−13	0	0	0	0	0.0214
F18	Ave	3	3.00151	3.0084	− 3.86278	8.4155	3	3	3	3
	Std	1.87 E−15	4.22 E−06	1.24 E−04	1.47 E−10	20.855	1.98 E−15	4.16 E−15	1.25 E−15	1.54 E−15
F19	Ave	− 3.86278	− 3.86278	− 3.86263	− 3.86278	− 3.86278	− 3.86263	− 3.8622	− 3.86278	− 3.86278
	Std	2.69 E−15	1.62 E−07	0.0051	0.0616	2.7 E−15	2.54 E−15	2.29 E−15	2.7 E−15	2.14 E−15
F20	Ave	− 3.27051	− 3.2744	− 3.28654	− 9.6334	− 3.2903	− 3.26654	−3.3177	− 3.28234	− 3.2687
	Std	0.0589	0.04155	0.1025	1.8104	0.0535	0.062251	0.02514	0.0570	0.05714
F21	Ave	− 9.2505	− 5.02215	− 8.1425	− 9.6334	− 5.6624	− 5.02250	− 5.95512	− 9.4735	− 8.24451
	Std	2.4151	3.32655	2.6925	1.8104	3.3543	3.55204	3.73707	1.7626	2.8705
F22	Ave	− 1.1479	− 6.05514	− 9.0278	− 9.0295	− 8.4434	− 7.3305	− 10.4015	− 1.2258	− 9.3320
	Std	1.3969	3.22051	1.60889	2.3922	3.3388	3.02551	2.01552	0.9704	2.1455
F23	Ave	− 10.2809	− 7.02251	− 10.3326	− 9.0333	− 8.0715	− 8.02251	− 10.22605	− 10.5364	− 9.34115
	Std	1.3995	3.80591	0.00415	2.9645	3.5961	2.55010	2.6 E−15	1.77 E−15	2.26651
F24 (CF1)	Ave	63.213	86.251	90.224	43.215	209.47	151.18	20.159	3.3415	44.2015
	Std	80.899	94.258	105.61	67.559	215.05	123.49	48.251	18.250	55.2150

Table 1 (continued)

Function	Criteria	QGDA	DABM	DA	DADE	BMDA	SC-DA	DA-OBL	DACG	LEA
F25 (CF2)	Ave	40.507	142.72	163.77	31.144	189.72	204.51	185.95	0.0000	88.251
	Std	61.225	119.14	88.145	51.269	169.25	118.98	65.224	0.0000	57.221
F26 (CF3)	Ave	138.25	214.67	210.99	235.12	274.20	273.14	218.44	104.21	160.251
	Std	33.258	77.215	99.554	79.025	201.54	110.54	114.66	14.266	31.250
F27 (CF4)	Ave	316.62	447.01	418.11	232.55	372.99	487.45	492.14	278.63	355.210
	Std	94.510	111.54	159.22	43.669	151.22	151.66	98.554	7.0501	112.89
F28 (CF5)	Ave	39.515	91.447	143.88	27.589	224.56	214.77	231.01	2.02 E−17	52.308
	Std	51.244	73.894	158.21	40.224	287.57	180.54	74.591	6.48 E−17	91.201
F29 (CF6)	Ave	684.50	811.21	837.47	628.97	845.24	794.50	845.55	539.21	768.48
	Std	201.23	173.58	136.25	185.67	132.58	175.62	80.221	123.55	191.20

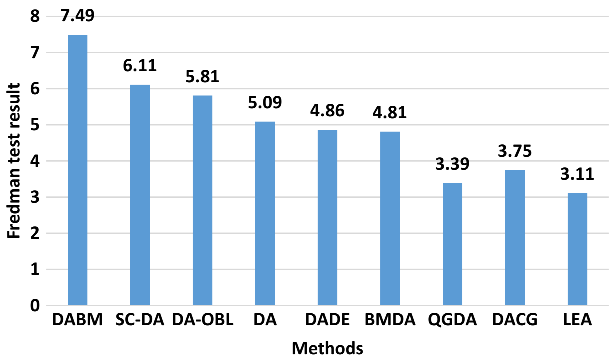


Fig. 5 Ranking results of the optimization methods using Fredman's test

$$g_6(x) = \delta(x) - 0.25 \le 0 \tag{33}$$

$$g_7(x) = 6000 - p_c(x) \le 0 \tag{34}$$

with the below variable domains:

$$0,1 \le x_1 \le 20.1 \le x_2 \le 10.0.1 \le x_3 \le 10.0.1 \le x_4 \le 2 \tag{35}$$

The second applied problem is the speed-reducer in a car's gearbox system that is applied in many other applications. The problem includes seven variables (x_1–x_7) composing the size of different parts and aims to minimize the objective function for constraint number 9 for all variables [59]. Its mathematical equation is captured in relation (36):

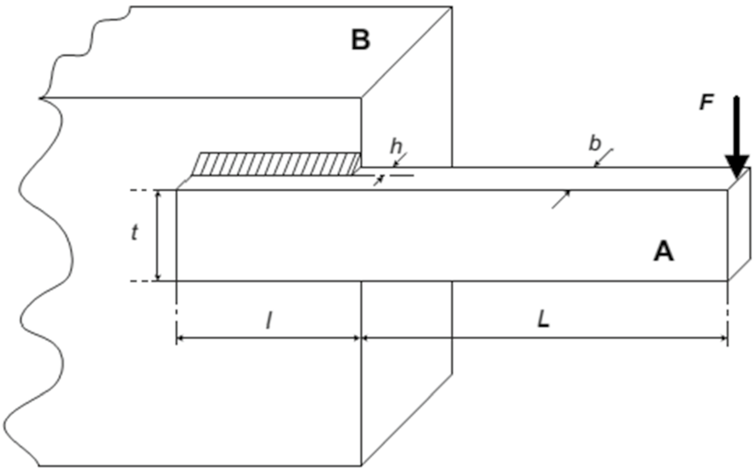


Fig. 6 The variables of the welded beam design problem [59]

$$\begin{aligned}J_{p2}(x) =&\,0.7854x_1x_2^2\left(3.3333x_3^2 + 14.9334x_3 43.0934\right)\\&- 1.508x_1\left(x_6^2 + x_7^2\right) + 7.477\left(x_6^3x_7^3\right)\\&+ 0.7854\left(x_4x_6^2 + x_5x_7^2\right)\end{aligned} \tag{36}$$

including the below constraints:

$$g_1(x) = \frac{27}{x_1x_2^2x_2} - 1 \le 0 \tag{37}$$

$$g_2(x) = \frac{397.5}{x_1x_2^2x_3^2} - 1 \le 0 \tag{38}$$

$$g_3(x) = \frac{1.93}{x_2x_3x_4^3x_6^4} - 1 \le 0 \tag{39}$$

$$g_4(x) = \frac{1.93}{x_2x_3x_5^3x_7^4} - 1 \le 0 \tag{40}$$

$$g_6(x) = \frac{\sqrt{\left(\frac{745x_4}{x_2x_3}\right)^2 + 157.5 * 10^6}}{85x_7^3} - 1 \le 0 \tag{42}$$

$$g_7(x) = \frac{x_2x_3}{40} - 1 \le 0 \tag{43}$$

$$g_8(x) = \frac{5x_2}{x_1 - 1} - 1 \le 0 \tag{44}$$

$$g_9(x) = \frac{x_1}{12x_2} - 1 \le 0 \tag{45}$$

with the below variable domains:

$$\begin{aligned}6 \le x_1 \le 3.0.7 \le x_2 \le 0.8.17 \le x_3 \le 28.7.3 \le x_4 \le 8.37.3 \le x_5\\ \le 8.32.9 \le x_6 \le 3.9.5 \le x_7 \le 5.5\end{aligned} \tag{46}$$

The results of the proposed method over 30 different executions in terms of the average of the objective function and standard deviation are provided against the results of the other methods.

According to Table 2, the least amount over 30 different executions is better for the LEA compared to the other methods. In the welding beam design problem, after the LEA, the AOA$_2$ and SOS methods are situated, respectively.

In the speed-reducer problem, after the LEA, the AOA_1 and AOA_2 methods are situated with an insignificant difference, respectively. As it is known, the results of the best methods in Table 1, which include DACG and QGDA methods, could not have better results than the proposed method.

4.3 The LEA results on IoT network clustering

This section provides the results comparison of the LEA with recent dragonfly-based algorithms in different Internet of Things network applications, including BDA [24], CH-DA [29] and DA-FA [66].

Different networks with 5–100 nodes and the parameters indicated in Table 3 are generated using MATLAB and then evaluated.

4.3.1 Investigating the proposed method in terms of packet delivery ratio

The performance of the proposed method is investigated in terms of three parameters packet delivery ratio, energy consumption, and network lifetime of up to 200 nodes. The packet delivery rate is defined as the number of packets successfully sent over the number of whole packets injected into the network. Figure 7 shows the linear diagram of the packet delivery ratio for 20 different networks in Gigabits with 10–200 nodes. The X-axis is the number of nodes, and the Y-axis is the number of delivered packets over whole packets in the network.

According to Fig. 7, generally, in a network with a high number of nodes, the packet delivery ratio is higher due to existing more paths among the nodes. However, approximating the current network situation and tuning the transmission power of the nodes are more challenging with increasing the number of nodes in the network and require more accuracy. The LEA has a more packet delivery ratio than DA-FA, CH-DA, and BDA methods, according to Fig. 6. The evaluation of the proposed method and the other methods over 20 networks of different numbers of nodes is provided in Table 4. The results indicate that increasing the number of nodes increases the packet delivery ratio linearly, and the proposed method could deliver 0.9317 percent of the packets on average.

4.3.2 Investigating the proposed method in terms of energy consumption

Figure 8 compares energy consumption in the network using the LEA and the other mentioned methods. The X-axis shows the number of nodes, and the Y-axis shows the amount of consumed energy in millijoules.

According to Fig. 8, the consumed energy rate with increasing the number of nodes is lower in the proposed method compared to the others. In the LEA, the best path selection to send messages is approximated considering the current situation of the network; the optimal paths include passing through lower nodes but with high energy; because the transmission power of the nodes is also considered to consume lower energy and accurate transmission of the packet. Table 5 shows

Table 2 Results comparison in terms of result precision over 30 different executions

Function	Criterion	Methods									
		PSO	ABC	FF	SSO	GOA	AOA1	AOA2	DACG	QGDA	LEA
J_{p1}	Ave	2.0111	2.1673	2.1974	1.7464	1.7495	1.7525	1.7425	1.7536	1.7520	1.7250
	Std	0.1085	0.2542	0.1952	0.0257	0.0054	0.0890	0.0225	0.0148	0.0105	0.0002
J_{p2}	Ave	3079.2623	2998.0628	3000.0054	2996.1132	2996.5212	2995.5402	2995.5102	2996.1161	2996.0201	2994.7831
	Std	26.2173	6.3545	8.3565	1.3351	1.9051	1.0456	1.2250	1.0509	1.1502	1.0037

Table 3 Simulation parameters

Parameter	Abbreviation	Value
Number of nodes of the network	*N*	5–100
Environment width and length	Area	100*100
Primary energy	Eo	0.5 J
Electric energy	Eelec	50 n J/bit
Transmission energy	Etx	10 p J/bit/m^2
Retrieval energy	Erx	0.0013 p J/bit/m^4
Data collection energy	EDA	5 n J/bit/signal
Packet size	Packet size	4000 Bits

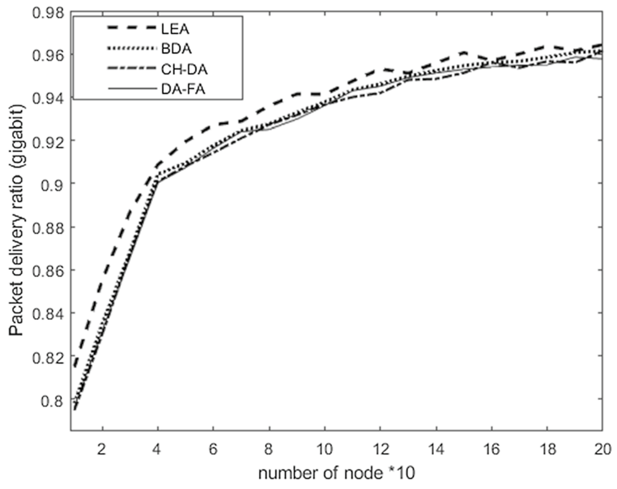


Fig. 7 Comparing the packet delivery ratio between the LEA and other methods

the consumption comparison for four different methods. The results are compared in terms of the consumed energy in KJ in networks with different numbers of nodes over 300 message transmission iterations in the network, and the amount of consumed energy of the network is measured for the number of iterations of fixed message transmission in the network.

According to Table 5, the average consumed energy of the LEA in 20 different networks is 94.95 KJs and is lower than the other methods.

4.3.3 Investigating the proposed method in terms of the network lifetime

Figure 9 shows the network lifetime for the LEA and the other methods. The X-axis shows the number of nodes, and the Y-axis shows the network lifetime in seconds.

Table 4 Comparing the packet delivery ratio among the methods

Number of network nodes	BDA	LEA	DA-FA	CH-DA
10	0.7980	0.8148	0.7963	0.7948
20	0.8347	0.8550	0.8314	0.8299
30	0.8684	0.8868	0.8655	0.8671
40	0.9043	0.9087	0.9004	0.9009
50	0.9095	0.9195	0.9073	0.9080
60	0.9177	0.9271	0.9163	0.9143
70	0.9245	0.9288	0.9240	0.9210
80	0.9275	0.9358	0.9250	0.9271
90	0.9329	0.9414	0.9297	0.9316
100	0.9377	0.9413	0.9360	0.9365
110	0.9433	0.9474	0.9429	0.9399
120	0.9461	0.9530	0.9450	0.9418
130	0.9497	0.9511	0.94905	0.9479
140	0.9523	0.9558	0.9511	0.9483
150	0.9546	0.9606	0.9528	0.9512
160	0.9560	0.9569	0.9538	0.9559
170	0.9566	0.9600	0.9547	0.9535
180	0.9585	0.9636	0.9549	0.9565
190	0.9606	0.9615	0.9585	0.9560
200	0.9615	0.9643	0.9577	0.9614

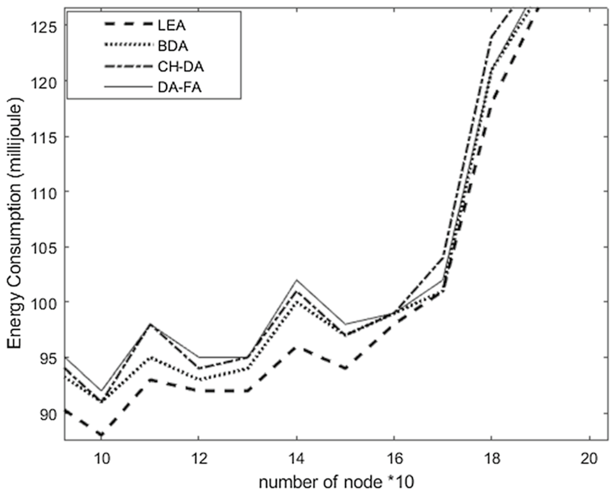


Fig. 8 Comparing energy consumption between the proposed method and the other methods

According to Fig. 9, the network lifetime is more for the LEA compared to the others. Table 6 shows the time when all network nodes are dead for 20 different networks. According to Fig. 9, the LEA has a higher network lifetime than the three methods DA-FA, CH-DA, and BDA. For the network lifetime, this is considered that the method which consumes more energy for sending messages through the nodes and comprises nodes with higher energy consumption certainly depletes its energy earlier. Therefore, the energy consumed by the nodes should be less to improve the network lifetime. It should be noted that the nodes with lower energy are not able to send/receive data and are considered dead. Table 6 shows the comparison between the proposed method and the others.

The lifetime values inserted in Table 6 are calculated in seconds for different networks; the calculation criterion is the death of the last node of the network. The results indicate the network average lifetime in the proposed method is 1656.9 s which is higher compared to the other methods.

4.3.4 Investigating the proposed method in terms of the death time of the nodes

In this section, the LEA is compared to the BDA [24], CH-DA [29] and DA-FA [66] methods. The death time of the first node is the number of the round in which the first node of the network stops working due to energy depletion, the death time of half of the nodes is the number of the round in which half of the network nodes stop working due to energy depletion. Finally, the death time of the last node of the network is the number of rounds in which the last node stops working due to energy depletion.

Figure 10 shows the performance comparison of the methods in terms of the number of live nodes over rounds' number.

According to Fig. 10, the network lifetime based on the death time of the first node comes with a lower delay in the proposed method compared to the other methods in 100 nodes, but in the BDA method [24], the death of the last node comes with more delay.

Table 7 provides the results comparison of the proposed method and the other methods in terms of the death of the nodes. The results are yielded by averaging over the statistical society for 20 executions per algorithm in a network with 100 nodes and 400 rounds.

According to Table 7 and Fig. 10, the death time of the first node in the LEA comes with more delay compared to the other methods, but the death time of the last node in the BDA method [24] comes with more delay. Accordingly, the proposed method guarantees the network lifetime better than the CH-DA [29] and DA-FA [66] methods. This feature is of special privilege in applications in which network coverage is important. Moreover, the death time of the first node in the LEA indicates that it can be a superior model for monitoring or tracking applications that require very accurate data. In these applications, delaying the first death time is far more important than the last. On the contrary, in some applications like periodic weather monitoring, increasing the overall network lifetime (delaying the last death time) is more desirable, whereas the BDA method is superior.

Table 5 Comparing the consumed energy between the proposed method and the other methods

Number of network nodes	BDA	LEA	DA-FA	CH-DA
10	76	74	78	77
20	85	81	88	85
30	91	86	92	94
40	88	85	89	91
50	87	84	88	89
60	88	86	88	90
70	91	90	94	92
80	90	89	92	93
90	94	91	96	95
100	91	88	92	91
110	95	93	98	98
120	93	92	95	94
130	94	92	95	95
140	100	96	102	101
150	97	94	98	97
160	99	98	99	99
170	101	101	102	104
180	121	118	121	124
190	128	127	129	130
200	139	134	140	139

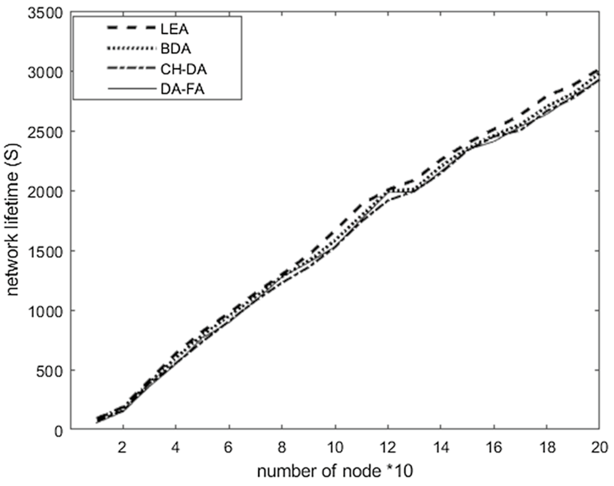


Fig. 9 Comparing the network lifetime between the proposed method and the other methods

Table 6 Comparing the network lifetime between the proposed method and the other methods

Number of network nodes	BDA	LEA	DA-FA	CH-DA
10	89	92	74	57
20	181	189	158	151
30	392	411	385	360
40	601	637	555	547
50	790	821	735	767
60	950	974	906	895
70	1112	1141	1078	1079
80	1280	1298	1228	1277
90	1410	1460	1358	1398
100	1580	1662	1526	1529
110	1790	1881	1739	1757
120	1991	2005	1915	1980
130	2011	2086	1994	1987
140	2201	2256	2144	2158
150	2353	2395	2334	2340
160	2461	2512	2451	2409
170	2552	2632	2503	2535
180	2701	2790	2665	2637
190	2812	2881	2775	2793
200	2980	3015	2927	2926

5 Conclusion

A novel algorithm inspired by the Lotus flower is presented in this article. This plant's propagation method, along with its leaves' self-cleaning property, is the main inspiration for a new optimization method. Since the Internet of Things is considered one of the most important applications in today's world, reducing energy consumption in the network and increasing the lifetime of nodes (considering the energy limitation of nodes) are important challenges in this field. This research presents a new method for clustering energy-based sensors, and the cluster heads are determined using the proposed algorithm. The proposed method is compared to the recently improved methods of the dragonfly algorithm in clustering Internet of Things networks. The results indicate the LEA is better than other methods in terms of network throughput, packet delivery rate, and network lifetime (in terms of the death of the first node). It also decreases energy consumption.

The experiment results of a set of different types of benchmark functions indicate that the proposed method is the best in single optimal functions and is one of the best methods in methods with multiple optimums and high dimensions. The results of the proposed method on the hybrid benchmark functions indicate that the proposed method is not the best method among all the functions in this field. However, it provides competitive results and generally shows the best performance in Fredman's ranking test over 29 benchmark functions. Also, the efficiency of the proposed method is

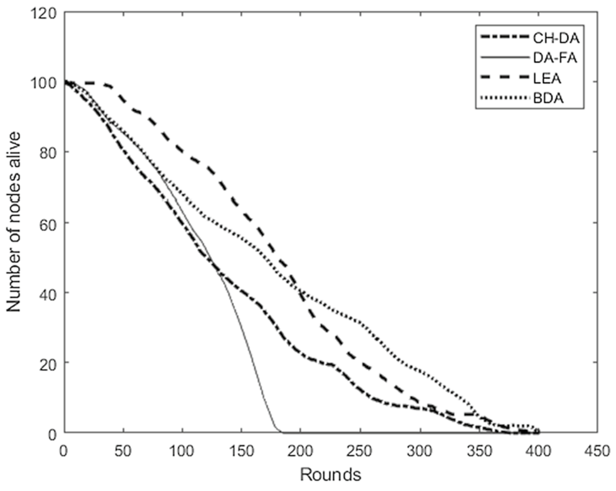


Fig. 10 Comparing the method's performance in terms of the number of live nodes over the rounds' number

Table 7 Comparing the results of the methods in terms of the death time of the nodes

Method	First node death time	Half nodes death time	Last node death time
Number of nodes: 100			
DA-FA	2	131	179
LEA	33	191	393
CH-DA	3	165	362
BDA	5	203	399

tested on two real-world optimization problems with multiple constraints. One example is the welded beam design optimization problem and the speed-reducer problem applied in a gearbox. It turns out that it has higher accuracy than other methods.

For future work, one may use the proposed method in discrete applications, including feature selection in data mining. Moreover, improving this method by using the fuzzy inference system to control the exploration and extraction phases could be investigated. Multi-objective optimization and high-objective optimization problems are other areas where this method could be developed. Furthermore, multi-modal optimization has numerous applications for which this algorithm could be investigated and developed.

Appendix A

See Table 8.

Table 8 Symbols

Row	Symbol	Description
The proposed method		
1	V_i^{t+1}	Speed of drop i in the next evolution iteration
2	V_i^t	Speed of drop i in current evolution iteration t
3	X_i^t	Possible solution i in current evolution iteration t
4	X_i^{t+1}	Possible solution i in the next evolution iteration
5	$X_{\text{deep pit}}^t$	Deepest drop pit
6	$\text{Rand}.r_1.r_2$	Random numbers between 0 and 1
7	q	Decremental coefficient of drop movement
8	Select_i^t	Pit selection
9	c_i^t	Pit capacity
10	t	Evolution iteration counter
11	L	Maximum number of evolution iteration
12	f_i^t	Value of the fitness function for possible solution i
13	f_{Max}	Maximum value of fitness function
14	f_{Min}	Minimum value of fitness function
15	σ, β, const	Constants
16	ΔX_i^{t+1}	Step length in the next evolution iteration’s movement
17	w	Movement step coefficient
18	ΔX_i^t	Step length in current evolution iteration’s movement
19	R	Decremental coefficient of flowers’ growth area
20	s	Separation movement coefficient
21	S_i^t	Separation movement variable for search agent i in evolution round t
22	a	Alignment movement coefficient
23	A_i^t	Alignment movement variable for search agent i in evolution iteration t
24	c	Cohesion movement coefficient

Table 8 (continued)

Row	Symbol	Description
25	C_i^t	Cohesion movement variable for search agent *i* in evolution iteration *t*
26	f	Food attraction coefficient
27	F_i^t	Food attraction coefficient for search agent *i* in evolution iteration *t*
28	e	Enemy distraction movement coefficient
29	E_i^t	Movement variable of enemy distraction for search agent *i* in evolution iteration *t*
30	X_-^t	The worst-found solution (enemy)
31	X_+^t	The best-found solution (food)
Calculations on networks		
32	E_{R_i}	Consumed energy by node *i*
33	b_i	Number of transmission bits in the network
34	α_1	Received energy coefficient
35	E_{T_i}	Consumed energy by node i for sending data to node *j*
36	$d_{i,j}^m$	Euclidean distance between nodes *i* and *j*
37	α_2	Transmission energy coefficient
38	b	Reinforcement coefficient
39	m	Attenuator coefficient
40	E_{initial}	Cluster head's initial energy
41	$E_{\max}$	Cluster head's consumed energy
42	E_{CH_i}	*i*th cluster's consumed energy
43	L_{net}	Network lifetime
44	$E_{\text{intra Cluster}_i}^R$	Amount of energy that the ith cluster head consumes to receive data from the *i*th cluster's sensors
45	C_i	Number of sensors of the *i*th cluster
46	$E_{\text{CH}_i S}^R$	Data retrieval energy from sensor *S*

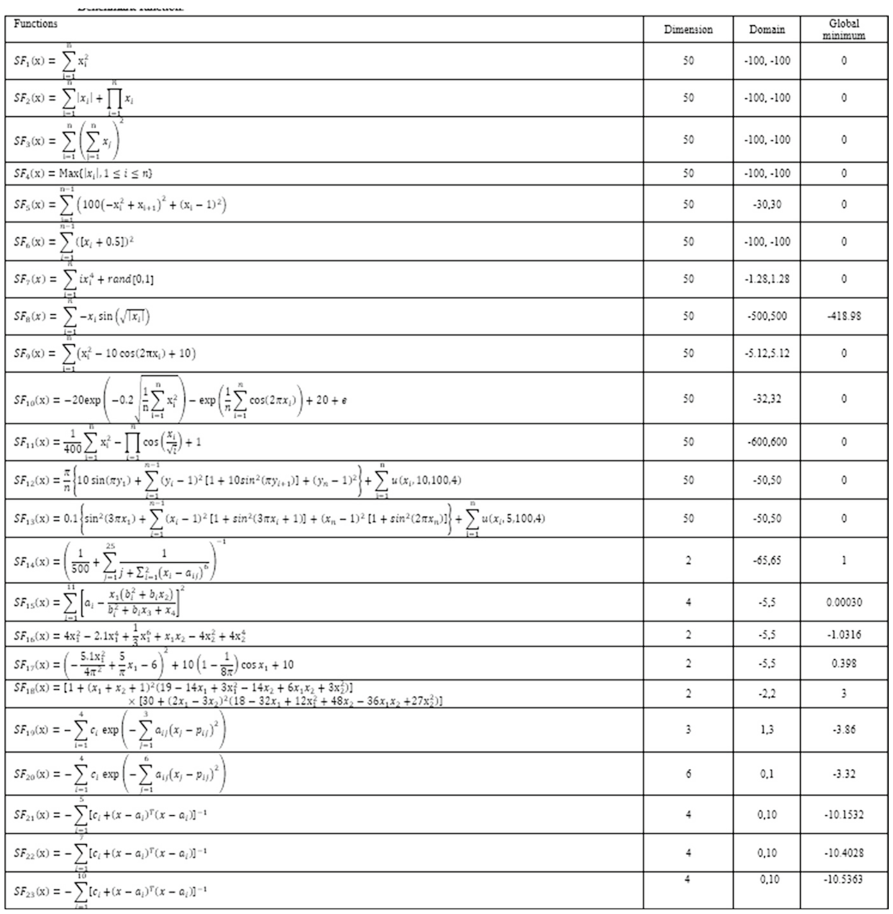
Table 8 (continued)

Row	Symbol	Description
47	*p*	Percentage of cluster heads relative to the whole network nodes
In the welding beam		
48	*L*	External length of the welding beam
49	*h*	Outer thickness of the welding beam
50	*l*	Inner thickness of the welding beam
51	*b*	Width of the welding beam
52	*t*	Inner thickness of the welding beam
53	C1	Expense of thewelding material per volume unit
54	C2	Expense of the consumable raw material per volume unit

Appendix B

See Table 9.

Table 9 Test function specifications



Functions	Dimension	Domain	Global minimum
$SF_1(x) = \sum_{i=1}^{n} x_i^2$	50	-100, -100	0
$SF_2(x) = \sum_{i=1}^{n} \lvert x_i\rvert + \prod_{i=1}^{n} x_i$	50	-100, -100	0
$SF_3(x) = \sum_{i=1}^{n}\left(\sum_{j=1}^{n} x_j\right)^2$	50	-100, -100	0
$SF_4(x) = \mathrm{Max}\{\lvert x_i\rvert, 1 \le i \le n\}$	50	-100, -100	0
$SF_5(x) = \sum_{i=1}^{n-1}\left(100(-x_i^2 + x_{i+1})^2 + (x_i - 1)^2\right)$	50	-30,30	0
$SF_6(x) = \sum_{i=1}^{n-1}([x_i + 0.5])^2$	50	-100, -100	0
$SF_7(x) = \sum_{i=1}^{n} i x_i^4 + rand[0,1]$	50	-1.28,1.28	0
$SF_8(x) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i\rvert}\right)$	50	-500,500	-418.98
$SF_9(x) = \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i) + 10)$	50	-5.12,5.12	0
$SF_{10}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	50	-32,32	0
$SF_{11}(x) = \frac{1}{400}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	50	-600,600	0
$SF_{12}(x) = \frac{\pi}{n}\left\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$	50	-50,50	0
$SF_{13}(x) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2[1 + \sin^2(3\pi x_i + 1)] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	50	-50,50	0
$SF_{14}(x) = \left(\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\right)^{-1}$	2	-65,65	1
$SF_{15}(x) = \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$	4	-5,5	0.00030
$SF_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	-5,5	-1.0316
$SF_{17}(x) = \left(-\frac{5.1x_1^2}{4\pi^2} + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$	2	-5,5	0.398
$SF_{18}(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	2	-2,2	3
$SF_{19}(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right)$	3	1,3	-3.86
$SF_{20}(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right)$	6	0,1	-3.32
$SF_{21}(x) = -\sum_{i=1}^{5}[c_i + (x - a_i)^T(x - a_i)]^{-1}$	4	0,10	-10.1532
$SF_{22}(x) = -\sum_{i=1}^{7}[c_i + (x - a_i)^T(x - a_i)]^{-1}$	4	0,10	-10.4028
$SF_{23}(x) = -\sum_{i=1}^{10}[c_i + (x - a_i)^T(x - a_i)]^{-1}$	4	0,10	-10.5363

Acknowledgements We sincerely express our gratitude to Dr. Matthias Mail from the Karlsruhe Institute of Technology (KIT) for his invaluable contributions to introducing the properties of the lotus effect, which have greatly influenced the development of our optimization method. His insights have served as a significant source of inspiration throughout this research.Furthermore, the authors would like to acknowledge the generous support from the KIT-Publication Fund of the Karlsruhe Institute of Technology, Germany. This support has been instrumental in the successful completion of this work.

Author contributions The authors confirm their contribution to the paper as follows: study conception and design: ED, MJ, HT, and MY; theorem analysis and interpretation of results: ED, MJ, HT, and MY; draft manuscript preparation: ED, MJ, HT, and MY; all authors reviewed the results and approved the final version of the manuscript.

Funding Open Access funding enabled and organized by Projekt DEAL. Not applicable.

Data availability Data are available on request from the authors.

Declarations

Conflict of interest The authors declare no conflict of interest.

Ethical approval Not applicable.

References

1. Mahfoud SW (1995) A comparison of parallel and sequential niching methods. In: Conference on Genetic Algorithms, Vol 136, p 143)
2. Heidari AA, Mirjalili S, Faris H, Aljarah I, Mafarja M, Chen H (2019) Harris hawks optimization: algorithm and applications. Future Gener Comput Syst 97:849–872
3. Zhang Y, Wang Y (2020) A novel energy-aware bio-inspired clustering scheme for IoT communication. J Ambient Intell Humaniz Comput 11:4239–4248
4. Sivakumar P, Radhika M (2018) Performance analysis of leach-ga over leach and leach-c in wsn. Procedia Comput Sci 125:248–256
5. Akbari MA, Zare M, Azizipanah-Abarghooee R, Mirjalili S, Deriche M (2022) The cheetah optimizer: a nature-inspired metaheuristic algorithm for large-scale optimization problems. Sci Rep 12(1):10953
6. Zamani H, Nadimi-Shahraki MH, Gandomi AH (2022) Starling murmuration optimizer: a novel bio-inspired algorithm for global and engineering optimization. Comput Methods Appl Mech Eng 392:114616
7. Trojovský P, Dehghani M (2022) Pelican optimization algorithm: a novel nature-inspired algorithm for engineering applications. Sensors 22(3):855
8. Zhao W, Wang L, Mirjalili S (2022) Artificial hummingbird algorithm: a new bio-inspired optimizer with its engineering applications. Comput Methods Appl Mech Eng 388:114194
9. Abdollahzadeh B, Gharehchopogh FS, Mirjalili S (2021) African vultures optimization algorithm: a new nature-inspired metaheuristic algorithm for global optimization problems. Comput Ind Eng 158:107408

10. Abualigah L, Abd Elaziz M, Sumari P, Geem ZW, Gandomi AH (2022) Reptile search algorithm (RSA): a nature-inspired meta-heuristic optimizer. Expert Syst Appl 191:116158
11. Dehghani M, Montazeri Z, Trojovská E, Trojovský P (2023) Coati optimization algorithm: a new bio-inspired metaheuristic algorithm for solving optimization problems. Knowl-Based Syst 259:110011
12. Jia H, Peng X, Lang C (2021) Remora optimization algorithm. Expert Syst Appl 185:115665
13. Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67
14. Azizi M, Talatahari S, Gandomi AH (2023) Fire Hawk optimizer: a novel metaheuristic algorithm. Artif Intell Rev 56(1):287–363
15. Azizi M, Aickelin U, Khorshidi HA, Baghalzadeh Shishehgarkhaneh M (2023) Energy valley optimizer: a novel metaheuristic algorithm for global and engineering optimization. Sci Rep 13(1):226
16. Xue J, Shen B (2023) Dung beetle optimizer: a new meta-heuristic algorithm for global optimization. J Supercomput 79(7):7305–7336
17. Debnath S, Baishya S, Sen D, Arif W (2021) A hybrid memory-based dragonfly algorithm with differential evolution for engineering application. Eng Comput 37:2775–2802
18. Alshinwan M, Abualigah L, Shehab M, Elaziz MA, Khasawneh AM, Alabool H, Hamad HA (2021) Dragonfly algorithm: a comprehensive survey of its results, variants, and applications. Multimed Tools Appl 80:14979–15016
19. Shirani MR, Safi-Esfahani F (2020) BMDA: applying biogeography-based optimization algorithm and Mexican hat wavelet to improve dragonfly algorithm. Soft Comput 24(21):15979–16004
20. Acı ÇI, Gülcan H (2019) A modified dragonfly optimization algorithm for single-and multiobjective problems using Brownian motion. Computational intelligence and neuroscience
21. Reynolds AM, Rhodes CJ (2009) The Lévy flight paradigm: random search patterns and mechanisms. Ecology 90(4):877–887
22. Yu C, Cai Z, Ye X, Wang M, Zhao X, Liang G, Chen H, Li C (2020) Quantum-like mutation-induced dragonfly-inspired optimization approach. Math Comput Simul 178:259–289
23. Khaleel LR, Mitras BA (2020) A novel hybrid Dragonfly algorithm with modified conjugate gradient method. Int J Comput Netw Commun Secur 8(2):40–48
24. Kakueinejad MH, Heydari A, Askari M, Keynia F (2020) Optimal planning for the development of power system in respect to distributed generations based on the binary dragonfly algorithm. Appl Sci 10(14):4795
25. Shah M, Borade H, Sanghavi V, Purohit A, Wankhede V, Vakharia V (2023) Enhancing tool wear prediction accuracy using walsh-hadamard transform, DCGAN and Dragonfly algorithm-based feature selection. Sensors 23(8):3833
26. George DT, Raj RE, Rajkumar A, Mabel MC (2023) Optimal sizing of solar-wind based hybrid energy system using modified dragonfly algorithm for an institution. Energy Convers Manage 283:116938
27. Singh H, Sawle Y, Dixit S, Malik H, Márquez FPG (2023) Optimization of reactive power using dragonfly algorithm in DG integrated distribution system. Electr Power Syst Res 220:109351
28. Joshi M, Kalita K, Jangir P, Ahmadianfar I, Chakraborty S (2023) A conceptual comparison of Dragonfly algorithm variants for CEC-2021 global optimization problems. Arab J Sci Eng 48(2):1563–1593
29. Dhumane A, Chiwhane S, Mangore Anirudh K, Ambala S (2022) Cluster-based energy-efficient routing in Internet of Things. In: ICT with Intelligent Applications: Proceedings of ICTIS 2022, Vol 1. Springer Nature Singapore, Singapore pp 415–427
30. Arafat MY, Moh S (2018) A survey on cluster-based routing protocols for unmanned aerial vehicle networks. IEEE Access 7:498–516
31. Al-Turjman F, Mostarda L, Ever E, Darwish A, Khalil NS (2019) Network experience scheduling and routing approach for big data transmission in the Internet of Things. Ieee Access 7:14501–14512
32. Mohsin AH, Bakar KA, Zainal A (2018) Optimal control overhead based multi-metric routing for MANET. Wirel Netw 24:2319–2335
33. Yang H, Li Z, Liu Z (2019) A method of routing optimization using CHNN in MANET. J Ambient Intell Humaniz Comput 10:1759–1768
34. Wang NC, Huang YF, Chen JC (2007) A stable weight-based on-demand routing protocol for mobile ad hoc networks. Inf Sci 177(24):5522–5537

35. Kumar S, Sinha DK, Kumar V (2020) An approach to improve lifetime of MANET via power aware routing protocol and genetic algorithm. In: 2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA). IEEE, pp 550–553
36. Nivetha SK, Asokan R, Senthilkumaran N (2019) Metaheuristics in Mobile AdHoc network route optimization. In: 2019 TEQIP III Sponsored International Conference on Microwave Integrated Circuits, Photonics and Wireless Networks (IMICPW). IEEE, pp 414–418
37. Panda N, Pattanayak BK (2020) ACO-based secure routing protocols in MANETs. In: New Paradigm in Decision Science and Management: Proceedings of ICDSM 2018. Springer Singapore, pp 195–206
38. Le T, Hu W, Corke P, Jha S (2009) ERTP: energy-efficient and reliable transport protocol for data streaming in wireless sensor networks. Comput Commun 32(7–10):1154–1171
39. Rajesh G, Mercilin Raajini X, Ashoka Rajan R, Gokuldhev M, Swetha C (2020) A multi-objective routing optimization using swarm intelligence in IoT networks. In: Intelligent Computing and Innovation on Data Science: Proceedings of ICTIDS 2019. Springer Singapore, pp 603–613
40. Khapre SP, Chopra S, Khan A, Sharma P, Shankar A (2020) Optimized routing method for wireless sensor networks based on improved ant colony algorithm. In: 2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence). IEEE, pp 455–458
41. Husnain G, Anwar S, Shahzad F (2017).Performance evaluation of CLPSO and MOPSO routing algorithms for optimized clustering in vehicular Ad hoc networks. In: 2017 14th International Bhurban Conference on Applied Sciences and Technology (IBCAST). IEEE, pp 772–778
42. Batth KK, Singh R (2016) Swarm intelligence for routing in mobile Ad Hoc networks. Int J Adv Inf Sci Technol (IJAIST). https://doi.org/10.1109/SIS.2005.1501605
43. Rathi PS, Mallikarjuna Rao CH (2020) Survey paper on routing in MANETs for optimal route selection based on routing protocol with particle swarm optimization and different ant colony optimization protocol. In: Smart Intelligent Computing and Applications: Proceedings of the Third International Conference on Smart Computing and Informatics, Vol 1. Springer Singapore, pp 539–547
44. Nath S, Banik S, Seal A, Sarkar SK (2016) Optimizing MANET routing in AODV: an hybridization approach of ACO and firefly algorithm. In: 2016 Second International Conference on Research in Computational Intelligence and Communication networks (ICRCICN). IEEE, pp 122–127
45. Kumar N, Vidyarthi DP (2018) A green routing algorithm for IoT-enabled software defined wireless sensor network. IEEE Sens J 18(22):9449–9460
46. Pasricha S, Ayoub R, Kishinevsky M, Mandal SK, Ogras UY (2020) A survey on energy management for mobile and IoT devices. IEEE Des Test 37(5):7–24
47. Arora VK, Sharma V, Sachdeva M (2016) A survey on LEACH and other's routing protocols in wireless sensor network. Optik 127(16):6590–6600
48. Yousaf A, Ahmad F, Hamid S, Khan F (2019) Performance comparison of various LEACH protocols in wireless sensor networks. In: 2019 IEEE 15th International Colloquium on Signal Processing & its Applications (CSPA). IEEE, pp 108–113
49. Meraihi Y, Ramdane-Cherif A, Acheli D, Mahseur M (2020) Dragonfly algorithm: a comprehensive review and applications. Neural Comput Appl 32:16625–16646
50. Barthlott W, Neinhuis C (1997) Purity of the sacred lotus, or escape from contamination in biological surfaces. Planta 202:1–8
51. Neinhuis C, Barthlott W (1997) Characterization and distribution of water-repellent, self-cleaning plant surfaces. Ann Bot 79(6):667–677
52. Li J, Zhang Z, Xu J, Wong CP (2000) Self-cleaning materials—lotus effect surfaces. Kirk-Othmer Encyclopedia of Chemical Technology
53. Yamamoto M, Nishikawa N, Mayama H, Nonomura Y, Yokojima S, Nakamura S, Uchida K (2015) Theoretical explanation of the lotus effect: superhydrophobic property changes by removal of nanostructures from the surface of a lotus leaf. Langmuir 31(26):7355–7363
54. Barthlott W, Mail M, Neinhuis C (2016) Superhydrophobic hierarchically structured surfaces in biology: evolution, structural principles and biomimetic applications. Philos Trans R Soc A Math Phys Eng Sci 374(2073):20160191
55. Collins CM, Safiuddin M (2022) Lotus-leaf-inspired biomimetic coatings: different types, key properties, and applications in infrastructures. Infrastructures 7(4):46
56. Yang XS (2012) Flower pollination algorithm for global optimization. In: Unconventional Computation and Natural Computation: 11th International Conference, UCNC 2012, Orléan, France, September 3-7, 2012. Proceedings 11. Springer Berlin Heidelberg, pp 240–249

57. Tanabe R, Fukunaga A (2013) Success-history based parameter adaptation for differential evolution. In: 2013 IEEE Congress on Evolutionary Computation. IEEE, pp 71–78
58. Site: https://www.mathworks.com/matlabcentral/fileexchange/124810-benchmark-problems
59. Cuevas E, Cienfuegos M (2014) A new algorithm inspired in the behavior of the social-spider for constrained optimization. Expert Syst Appl 41(2):412–425
60. Karaboga D, Akay B (2009) A comparative study of artificial bee colony algorithm. Appl Math Comput 214(1):108–132
61. Yang XS (2010) Firefly algorithm, stochastic test functions and design optimisation. Int J Bio-inspir Comput 2(2):78–84
62. Arumugam MS, Rao MVC, Tan AW (2009) A novel and effective particle swarm optimization like algorithm with extrapolation technique. Appl Soft Comput 9(1):308–320
63. Abualigah L, Diabat A, Mirjalili S, Abd Elaziz M, Gandomi AH (2021) The arithmetic optimization algorithm. Comput Methods Appl Mech Eng 376:113609
64. Hashim FA, Hussain K, Houssein EH, Mabrouk MS, Al-Atabany W (2021) Archimedes optimization algorithm: a new metaheuristic algorithm for solving optimization problems. Appl Intell 51:1531–1551
65. Pan JS, Zhang LG, Wang RB, Snášel V, Chu SC (2022) Gannet optimization algorithm: a new metaheuristic algorithm for solving engineering optimization problems. Math Comput Simul 202:343–373
66. Singh P, Mittal N (2020) Efficient localisation approach for WSNs using hybrid DA–FA algorithm. IET Commun 14(12):1975–1991