Electric fish optimization: a new heuristic algorithm inspired by electrolocation

Article *in* Neural Computing and Applications · December 2019

DOI: 10.1007/s00521-019-04641-8

Electric Fish Optimization: A New Heuristic Algorithm Inspired by Electrolocation

Selim YILMAZ · **Sevil SEN**

Abstract Swarm behaviors in nature have inspired the emergence of many heuristic optimization algorithms. They have attracted much attention, particularly for complex problems, owing to their characteristics of high dimensionality, nondifferentiability, and the like. A new heuristic algorithm is proposed in this study inspired by the prey location and communication behaviors of electric fish. Nocturnal electric fish have very poor eyesight and live in muddy, murky water, where visual senses is very limited. Therefore, they rely on their species-specific ability called *electrolocation* to perceive their environment. The active and passive electrolocation capability of such fish is believed to be a good candidate for balancing local and global search, and hence it is modeled in this study. A new heuristic called *Electric Fish Optimization* (EFO) is introduced and compared with six well-known heuristics (Simulated Annealing, SA; Vortex Search, VS; Genetic Algorithm, GA; Differential Evolution, DE; Particle Swarm Optimization, PSO; and Artificial Bee Colony, ABC). In the experiments, a well-known selection of 50 basic and 30 complex mathematical functions, 13 clustering problems, and 5 real-world design problems are used as the benchmark sets. The simulation results indicate that EFO is better than or very competitive with its competitors.

Keywords Nature-inspired algorithm · Heuristics · Swarm intelligence · Real parameter optimization · Single-solution algorithms · Population-based algorithms · Real-world applications

1 Introduction

Optimization is the process of finding the best possible or at least admissible solutions among numerous solution sets for a problem with a certain number of constraints [78]. The characteristics of complex problems such as multimodality, high dimensionality, nondifferentiability, nonlinearity make it impossible to solve these problems using classical algorithms. Owing to the No Free Lunch (NFL) [74] theorem, which states that there is no sole method that provides the best solution to all problems, there has been growing interest in approximate heuristic algorithms that do not guarantee an optimal solution to the problem, but find admissible solutions within an acceptable time. Such algorithms have been applied to numerous engineering problems in the literature, such as image processing [54, 22], data mining [67], routing [18], and speech recognition [68].

There are two major search components in heuristics: *Exploration* and *exploitation*. Exploration, or global search, is the ability of a heuristic that directs it to search unknown regions of the search space by keeping the search agents as diverse as possible, whereas exploitation is the ability that focuses on searching by exploiting information of the best agents in the space. The success of a heuristic is strongly dependent on how well these components are balanced [78, 76]. It is a rule of thumb to use exploration at the beginning of the iterations to avoid trapping in local optima and to fade it out by a lapse of iterations to allow an algorithm to search its locality [60, 70].

The vast majority of metaheuristics have been derived from the behavior of biologic or physical systems in nature. There have been very popular heuristic algorithms in the literature, called *Nature-Inspired Algorithms*, that model natural phenomena. For example, pheromone, deposited by ants, is the basis of Ant Colony Optimization (ACO), and the waggle dance is the basis of ABC. In this study, a new population-based heuristic algorithm is proposed, named Electric Fish Optimization (EFO), which is inspired by the *electrolocation* and *electrocommunication* capabilities of electric fish

Selim YILMAZ
WISE Lab, Hacettepe University, Ankara, Turkey
Tel.: +90-312-7807524
E-mail: selimy@cs.hacettepe.edu.tr

Sevil SEN
WISE Lab, Hacettepe University, Ankara, Turkey
Tel.: +90-312-7807538
E-mail: ssen@cs.hacettepe.edu.tr

for solving real parameter optimization problems. Electrolocation is the ability of electric fish that enables them to perceive objects/prey around them. The electrolocation capability of electric fish is examined in two major branches: *active electrolocation* and *passive electrolocation*. These capabilities inherently fulfill the exploitation and exploration capabilities of the proposed EFO algorithm. Electrocommunication is the species-specific communication behavior of electric fish through electric signals. To investigate the performance of the proposed algorithm with respect to other heuristics, we have conducted three experiments. In the first experiment, 50 basic and 30 complex mathematical functions (also known as the *CEC'17 Benchmark Suite* [5]) are employed as a benchmark set. The Wilcoxon Signed Rank Test and multicriteria decision-making methods, namely VIKOR and TOPSIS, are applied to evaluate the statistical differences between the proposed EFO algorithm and the competitor heuristics. In the second experiment, the performance of the proposed EFO algorithm is analyzed as a clustering method, which is one of the important techniques used in data mining. Clustering methods are widely used in many research areas from image processing, geophysics, agriculture to security and crime detection [29]. In this experiment, 13 clustering datasets, which are obtained from actual data, are used as the test bed. The final experiment comprises five well-known real-world design problems: *welded-beam*, *spring design*, *pressure vessel*, *three-bar truss design*, and *speed reducer* problems. The main purpose of the third experiment is to observe the performance of EFO on constrained problems. The proposed algorithm is compared with well-known heuristics in the experiments. The results obtained from all the experiments show that EFO ensures a fine balance between global and local searches, maintaining an 'explore first, exploit later' approach, and that EFO performs much better than or competitively with its competitors, especially on complex design problems employed here, which is essential for most real-world problems. Hence EFO is introduced to the community as a new nature-inspired algorithm.

To the best of our knowledge the proposed algorithm in this study is the first model inspired by the active/passive electrolocation and electrocommunication capabilities of electric fish to solve complex problems. One recently proposed algorithm, *Fish Electrolocation Optimization (FEO)* [27], models *capacitance detection*, the ability of electric fish to discriminate animate objects from inanimate objects by analyzing their capacitive and resistive properties. Individuals in FEO consider object-related attributes such as capacitance, slope, and width of the object, which are measured through active electrolocation. Although the metaphor utilized for the creation of FEO is interesting, there are some critical deficits in the algorithm's design choices. The most noticeable is that it possesses many parameters (more than 15), which are problem-dependent. Thus, a small change in the parameter set may yield a high variation in its performance. The performance validation is also quite questionable. Because it is comparatively tested on a limited number of low-dimensional problems and only on one high-dimensional problem, a fair comparison is missing. The proposed EFO, on the contrary, models the electric discharge activity that changes as a response to the distance between the fish and the target object while electrolocating. Moreover, the communication between electric fish through electric signals is modeled in EFO as well, which is crucial for swarm-based algorithms. Contrary to FEO, the proposed EFO algorithm showed overwhelmingly better performance than FEO with fewer parameters and simpler search components, which makes EFO an easy-to-implement algorithm. The modeled behaviors of electric fish are explained in more detail in Section 3.3.

There are also a few studies that exploit the active electrolocation capability of electric fish to design robotic sensory systems. A model mimicking the active electrolocation behavior of weakly electric fish was developed in [42] for the navigation around a sphere of an underwater robot. The model comprises a number of spherical electrodes, one of which emits a signal and the remaining ones are the receivers. Perturbation created by the surrounding object helps the model estimate the position and radius of the object. In [45], the authors examined *ghost kniefish*, a weakly electric fish, to develop a prototype for the purpose of designing better next-generation autonomous underwater vehicles (AUV). Not only was the prototype inspired by the active electrolocation, but also by the maneuvering behavior of electric fish once they detect prey. Another study [46] developed a signal processing algorithm inspired by the behavior of weakly electric fish and designed an artificial sensor array that provided electrosensory capabilities to a submarine robot. In [66], the active electrolocation capability of electric fish was modeled to create a sensory robot. This robot continuously processes the electrosensory data to control the position of an electric field and to localize underwater objects. Interested readers are referred to [53] for the details of these studies. On one hand, as stated earlier, these studies are in the field of robotic sensor technology and were only inspired by the active electrolocation capability of electric fish. Moreover, because the proposed design solutions rely on a single sensory agent, swarm intelligence is not introduced in these studies [42, 45, 46, 66]. On the other hand, the proposed EFO algorithm is inspired by both the active and passive electrolocation of electric fish, and it provides swarm intelligence by using communication among electric fish.

The contributions of this study are summarized as follows:

- The proposed EFO algorithm is the first swarm-based heuristic algorithm inspired by both active and passive electrolocation behaviors of electric fish to ensure local and global search mechanisms. In addition, electric organ discharge activities (frequency and amplitude) are modeled for the first time in a swarm-based heuristic algorithm. The metaphor behind the proposed EFO algorithm supports a well-balanced local and global search as well as an 'explore first, exploit later' approach.
- The proposed EFO algorithm has only two parameters that need to be tuned, as opposed to FEO, which makes the proposed algorithm much easier to use. In addition, it is shown in the experiments that the parameters of EFO do

not have a dramatic effect on the performance of the algorithm, which makes EFO a robust algorithm. The proposed algorithm is shared with the community[1].
- The performance of the proposed EFO algorithm is extensively evaluated on various problems. EFO is shown to perform considerably better than almost all existing approaches in the literature over different bound-constrained, clustering, and real-world design problems.

The remainder of this paper is organized as follows. Related previous works in the literature have been reviewed in Section 2. Section 3 introduces the proposed EFO algorithm in detail. Section 4 introduces the experimental environment and presents the experimental results. Finally, conclusions of the study are drawn in Section 5.

2 Related Work

Heuristic algorithms can be classified in many ways in the literature, but it is the number of solutions employed that makes heuristics fundamentally distinctive in the literature [12, 69]. Within this criterion, they are classified as *single-solution-based* and *population-based* heuristics. This paper reviews the previous works according to this taxonomy.

Single-solution-based heuristics search with a single agent and continue in a sequential manner. Simulated Annealing (SA) [40] is a single-solution-based heuristic algorithm inspired by the annealing process of glass or metal materials. Contrary to most single-solution-based heuristics, it performs global search by occasionally accepting worse solutions until it becomes stable, which is determined by the temperature value. Vortex Search (VS) is another single-solution-based algorithm, which is inspired by the vortex pattern, as the name suggests. VS randomly searches by shrinking space and accepts only better solutions at every iteration.

Population-based heuristics employ multiple agents to search in parallel. They are studied under two major groups: *Evolutionary-based algorithms* and *swarm intelligence-based*, or *swarm-based*, algorithms. Evolutionary-based algorithms are inspired by the natural selection phenomenon in evolution. Genetic Algorithm (GA) [30] and Differential Evolution (DE) [15] are two popular evolutionary-based algorithms inspired by the notion of *survival of the fittest*, and they use similar generic operators: *crossover*, *mutation*, and *reproduction*. DE differs from GA in representation, which makes it convenient to handle problems with real parameters. Recently proposed Backtracking Search Algorithm (BSA) [13] is another evolutionary-based algorithm relying on two different populations. It comprises five phases: *initialization*, *selection-I*, *mutation*, *crossover*, and *selection-II*. Initialization is responsible for random generation of the population. Selection-I ensures the algorithm to have a memory by calculating the search direction and by permuting the historical population. As common in most of the Evolutionary Algorithms (EA), mutation and crossover operators are used to generate a trial population considering these two populations. Greedy selection is used in the selection-II phase to enable trial solutions to be included in the population.

Swarm-based algorithms are inspired by the collaborative intelligent behavior of swarm agents. Particle Swarm Optimization (PSO) [39] and Artificial Bee Colony (ABC) [34, 37] algorithms are well-known swarm-based heuristics. PSO is inspired by the collective behavior of a school of fish or a flock of birds. It makes use of historical best position of every solution (called as particle in the algorithm) and global best solution in order to change the velocity of every particle, and hence to determine their new positions. Its straightforward structure and use of a single search operator make PSO very popular in the literature. ABC is inspired by the collective behavior of foraging honey bees. It consists of three search phases performed by employed, onlooker, and scout bees. Employed bees are in charge of exploiting their solutions (called as food source in the algorithm). Then, onlooker bees further develop the high quality ones of these solutions. The poor food sources are abandoned and new sources are discovered by scout bees. Inferior Search Algorithm (ISA) [24] is a novel heuristic method inspired by an interior design procedure targeting a beautiful and more decorative view. Depending on a pre-defined value, algorithm divides the population into two groups (excluding the fittest one): *composition* and *mirror*. The solutions (called elements in the algorithm) in the composition group performs a random walk to explore by shrinking search space whereas the solutions in mirror group are placed between any solution and the fittest solution for exploitation. T-Cell Algorithm (TCA) [2] is another heuristic algorithm with multi-group population that is inspired by a central role of T-cells in cell-mediated immunity. Every group of T-cells (*virgin* cells, VC; *effector* cells, EC; and *memory* cells, MC) has its own goal: diversity has been ensured through the VC group, whereas the exploitation of the best solution and exploration of conflicting zones has been performed by the MC and EC groups respectively. Crow Search Algorithm (CSA) [4] is inspired by the foraging behavior of crows. The form of the flock, memorization of the position of their hiding places, thievery, and protection of their caches are the most intelligent behaviors of crows that are modeled in the algorithm. Similar to PSO, the historical best position of every solution is memorized here and search is performed based on that. CSA is easy to implement with only two operators; but there are two control parameters that should be tuned and directly affect the balance between exploration and exploitation. Another foraging behavior-inspired algorithm is Bacterial Foraging Optimization (BFO) [57, 48]. The move to regions of high nutrient levels and the communication between bacteria are the primary motivation factors of the algorithm. In order to perform search, algorithm uses three nested progresses that are applied for every solution: elimination-dispersal, reproduction, and chemotaxis. However, these progresses highly increase the computational cost

[1] https://wise.cs.hacettepe.edu.tr/projects/efo/

of the algorithm. In addition to that BFO involves a sorting algorithm to sort population at the end of every generation. Teaching-Learning-Based Optimization (TLBO) [58] is based on the mathematical model of an effect of teachers on a group of learners. There are two successive processes in every generation of TLBO: *teacher* and *learner* phases. The exploitation of the best solution and exploration of the search space are ensured in the teacher and learner phases, respectively. Two-stage State Transition Algorithm (STA) [28] is another swarm-based algorithm in which every solution is regarded as a state and the update of the current solution is treated as a state transition which is performed through *rotation*, *expansion*, and *axesion* functions corresponding to exploitation, exploration, and a single dimensional search, respectively.

In addition to these studies, modification and hybrid methods have also been proposed in the literature in an attempt to further improve the performance of heuristics on real-world problems. The stagnation and slow convergence problems arisen in PSO were handled by integrating it with DE in [44]. The particles in PSO prematurely terminate exploration and stagnate in the rest of the search process especially when they gather to the same local minimum. To avoid that, the proposed approach enables PSO to evolve new particles using only the half of the population involving the poor particles and then employs DE to generate three offspring from historical bests of these particles. Another modification that improves the search performance and that prevents the premature convergence problem of PSO is proposed in [63] by introducing mutation operator with Gaussian probability distribution. A binary-coded PSO is proposed in [16] for dealing directly with the problems of having real, integer, and discrete variables. The only modification is in the velocity and position update scheme of the standard real-coded PSO. While continuous variables are still handled by the standard real-coded PSO, discrete and integer variables are updated in the proposed operators. Therefore, real variables are represented by a single dimension, while every integer and discrete variable is represented by an array of bits that are calculated from the upper limit, which incredibly increases total number of parameters and hence the problem's complexity. DE has also been modified to achieve a better optimization performance in various studies. In [50], it has been subjected to a simple modification to generate more than one offspring per parent so that the probability of obtaining better solution is increased; however it leads the proposed algorithm to have an additional parameter that needs to be prior set and an increase in the computational cost. The exploration and exploitation capability and the poor convergence rate of DE have been enhanced in [51] through a new mutation scheme, called triangular mutation rule. In contrast to the standard DE, where only random or best vectors are included, the proposed scheme relies on a combination of a random vector and a difference vector between best, better, and worst vectors. Another modification to DE has been introduced for constrained optimization problems in [83]. This study adopts a dynamic stochastic ranking scheme in order to keep promising infeasible solutions that are often discarded during the selection procedures. Although the proposed scheme ensures a fine search balance between feasible and infeasible regions, it brings about an additional computational complexity to the algorithm.

The performance of Evolutionary Strategy (ES) [64] has been improved in [49] by introducing three simple selection mechanisms to maintain a higher diversity. Even if the proposed approach does not require an extra parameter and yields lower computational cost, its weak local search implied by the authors in the paper needs to be further improved. In [82], an evolutionary-based algorithm with a novel roulette inversion operator has been proposed to maintain a diverse population. The main goal of the roulette inversion operator is to enable the algorithm to explore every search dimension evenly. In [8, 9], GA has been hybridized with artificial immune system to help the population move into the feasible region. Chromosomes are classified as *antigens* and *antibodies* depending on the feasibility. Afterwards, they are subjected to a series of operators such as tournament selection, mutation, crossover, and the like. The higher computational complexity of the proposed hybrid method is stressed as the obvious drawback of the approach. In [33], Cuckoo Search (CS) [75] was hybridized with GA through its the generic operators in order to improve poor balance between global and local search capability of CS. Crossover in GA and the Lévy flight in CS contribute the algorithm by performing a further diversified global search in the space and reducing the premature convergence problem in CS. Mutation allows a better local search through a moderate change on individuals. The improved version of Fruit-fly Optimization Algorithm (FOA) [56], which relies on the excellent behavior of olfaction and vision of the fruit fly while foraging, has been proposed in order to address the problems of premature convergence and poor quality of solutions in standard FOA [19]. Linear diminishing and logistic chaotic mapping are employed to avoid becoming trapped in local optima and to increase stability of FOA, respectively.

To sum up, numerous algorithms and their improvements have been introduced in the literature. However, the research area is still open for further improvements. Before concluding the section, it is also worth mentioning here that deep learning methods [84, 85, 86] have recently attracted researchers' attention, owing to their excellent performance in solving various industrial problems such as security [47], big data [87], image analysis [43], and agriculture [32].

3 Electric Fish Optimization (EFO)

3.1 Electric fish

Electric fish constitute only a fraction of all fish species ($\sim$1.2%, corresponding to 350 electric fish species out of more than 30,000 total fish species). They live in muddy water where their visual sense are very restricted and most

of them are nocturnal. However, they possess a species-specific capability, *electrolocation*, which is the distinguishing sense capability that electric fish rely upon to locate objects such as prey and obstacles. Electric fish are categorized as strongly and weakly electric fish, depending on the strength of the electric field they generate. Strongly electric fish mostly use the electrolocation capability for offensive purposes and their field intensity ranges from 10 to 600 V, which is sufficient to stun their prey. In contrast, weakly electric fish only generate electric fields of intensity between a few hundred millivolts and a few volts, which is utilized to navigate, communicate, detect objects, etc. [52].

Electric fish have an electric organ containing special electric disc-like cells (called *electrocytes*), which are used to generate an electric field. This organ is generally located at the tail of the body. The simultaneous excitation of *electrocytes* is called *electric organ discharge* (EOD) in the literature. EOD is characterized by its *frequency* and *amplitude*. The EOD frequency is inversely proportional to the interval between two consecutive electric signals. It is highly affected by the presence of a novelty (sharp increase) and the distance to the stimulus (regular increase). The EOD amplitude, however, is proportional to the fish's size, owing to the electrocytes that are added during growth, and it decays with the inverse cube of the distance from the body surface [52, 41].

Depending on the EOD activity, electrolocation is examined under two categories: *active* and *passive*. In active electrolocation, electric fish generate EOD and sense their near surroundings through the modulations in the electric field. The effective range of active electrolocation is very limited, but very helpful to fish living in dark and complex environments for finding and identifying their prey. In passive electrolocation, electric fish can perceive and react to electric signals of external sources through their electroreceptors. Contrary to active electrolocation, the effective range of passive electrolocation is much wider, which allows them to locate objects at distances exceeding the active electrolocation range and to communicate with other fish.

Electric fish are both electrogenic and electroreceptive, which means they have capability to perceive electric currents emanated either from their conspecifics' organs or from organisms living in the same environment, as well as to generate an electric field, which enables them to communicate with each other through electrical signals. By using this capability, called *electrocommunication*, electric fish can exchange information about *i)* their identity, gender, status in the hierarchy; *ii)* their physiological or motivational state; and *iii)* aspects about their environment, presence of food, etc. [52].

3.2 Swarm intelligence in electric fish

The term swarm is particularly used for aggregated animals such as fish schools, bird flocks, and colonies of insects such as ants, bees, and termites. Swarm intelligence, a form of artificial intelligence (AI), is the emergent collective intelligence of groups of simple swarm agents [11]. It stems from the interaction of agents with their neighbors. In swarm intelligence, there is no centralized (global level) control structure that dictates how an agent should behave, but the rules that specify the interactions among the swarm agents are executed at a purely local level. The first use of the expression 'swarm intelligence' dates back to the 1980s in the context of cellular robotic systems [11]. Since then, researchers in the field of optimization have developed several swarm-based algorithms. In the 1990s, the success of ACO and PSO in solving different types of problems has attracted researchers to the development of new swarm-based algorithms.

Self-organization is a key feature of a swarm that results in global level (macroscopic level) response by means of low-level interactions (microscopic level) [11, 36]. For example, the emerging structure of foraging (macroscopic level) in ants includes organized networks of pheromone trails (microscopic level). The following four characteristics are the basic ingredients upon which self-organization relies [11]: Positive feedback, negative feedback, fluctuation, and multiple interactions. Each characteristic is introduced and explained from the perspective of electric fish as follows:

1. *Positive feedback:* This promotes the creation of convenient structures. Examples of positive feedback include recruitment of ant or bee species to a food source depending on the trail laying or waggle dance, respectively.

   An electric fish that is close to a stimulus or prey source generates an electric field with higher EOD frequency, which means that it will search in its locality through its active electrolocation capability. As the fish begins to produce electric more often, it attracts other fish.

2. *Negative feedback:* This counterbalances the positive feedback, and it is required to avoid saturation, exhaustion, and competition.

   An electric fish far away from a stimulus or prey source decreases its EOD frequency, switches to silent mode, and thus no longer attracts its conspecific.

3. *Fluctuations:* This enables the discovery of new solutions, and in a swarm it is crucial for creativity and innovation. For example, foragers in an ant swarm follow trails with a certain error and may get lost in the colony; the lost foragers can find new and unexplored new food sources.

   A fish perceives electrical stimuli not only from its conspecific's electric organ, but also from other aquatic organisms.

4. *Multiple interactions:* These enable some agents in the swarm to receive information originating from other agents for information to spread throughout the swarm.

   Electric fish communicate with each other through active and passive electrolocation and broadcast messages to each other about their surroundings: *electrocommunication*.

3.3 Methodology

All heuristics use a common framework. However, the usage of search operators depends on the model, which is what differentiates heuristics in the literature [60, 20]. EFO is no exception and uses very simple search operators. The details of the algorithm will be presented in the subsequent sections. First, the assumptions that transform the behavior of electric fish into the heuristic algorithm, in accordance with their true nature, are introduced. The first assumption is that there is an infinite food source in the search space, where one food source is recognized as the best source. Electric fish, corresponding to the individuals in the algorithm, are located somewhere in the space and carry their location information. The quality of each individual is determined by its distance to the location of the best source, and hence the problem at hand becomes a problem of finding the best quality food source. Another assumption, which is inspired by natural evolution, is that fish possessing high-quality resources over a long period of time grow faster and thus are able to produce higher-amplitude electric signals than the others [81].

The intelligent behaviors of electric fish are modeled as follows:

– *Active electrolocation:* Electric fish produce more frequent electric signals by means of their electric organs the nearer they get to the best food source (Fig. 1-A and Fig. 1-C)).

  Active electrolocation has a very limited range, and therefore it has been used to ensure the local search through individuals with better fitness values, such that these individuals could be better able to search their vicinity, as in electric fish.

– *Passive electrolocation:* Other fish do not generate an electric field, but rely on electric signals emanated either from their conspecific's electric organ or from animate organisms (Fig. 1-B).

  Because passive electrolocation has a wider range than active electrolocation, in the algorithm, it has been used to balance active electrolocation and to ensure the global search by enabling individuals, particularly those of poor fitness, to explore distant spaces.

– *EOD frequency:* In nature, the frequency of electric field generation depends on the closeness to the source. The fish closest to the best source are expected to produce electric field more frequently than others.

  EOD frequency has been used in EFO as a key to determine the role of each individual at time t, because it is an indicator used by electric fish to find out which individuals are in the vicinity of a better food source. As in nature, individuals with higher frequency employ active electrolocation, and others use passive electrolocation.

– *EOD amplitude:* The amplitude of electric field depends on the growth (and thus the size) of the fish, and it determines the effective range of the electrical stimuli.

  In the EFO algorithm, owing to its ability to determine electric fields' domination, EOD amplitude has been used to determine the effective range in local search and the probability of individuals' being sensed in global search.

3.3.1 Population initialization

Initially, the electric fish population (N) (henceforth called individuals) are randomly spread through the search space by taking into account the boundaries of the space.

$$x_{ij} = x_{\min j} + \phi\left(x_{\max j} - x_{\min j}\right) \tag{1}$$

where x_{ij} represents the position of the ith individual in the population of size $|N|$ $(i = 1,2,...,|N|)$ in the d-dimensional search space. $x_{\min j}$ and $x_{\max j}$ are the lower and upper boundaries for dimension $j \mid j \in 1,2,...,d$, respectively. $\phi \in [0,1]$ is a random value drawn from a uniform distribution.

Individuals in the population move around the search space through their active or passive electrolocation capability, just after the initialization phase. The frequency plays a key role in the EFO algorithm to balance exploration and exploitation, and is used to determine whether an individual will perform active or passive electrolocation. It enforces better individuals (active mode individuals), which are most likely to be vicinity of promising regions, to exploit their neighborhood, and it leads the other individuals (passive mode individuals) to explore the search space so that they discover new regions, which is essential for multimodal functions.

In the EFO algorithm, as in nature, individuals with a higher frequency employ active electrolocation, and others use passive electrolocation. The frequency value of an individual ranges from a minimum value f_{min} to a maximum value

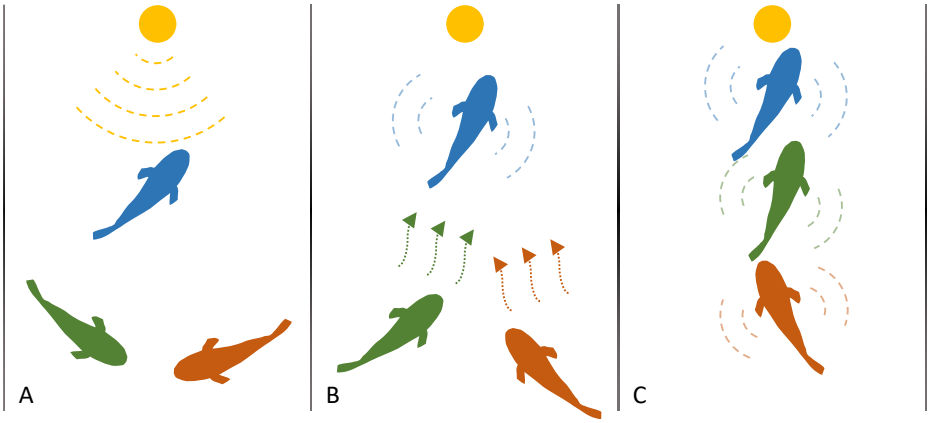

Fig. 1: Prey detection and communication in a fish swarm. A) Fish (colored blue) perceives the signals from the source. B) It switches into active mode, generating signals with a higher frequency, and broadcasts its surroundings. C) Other fish (colored green and brown) in passive mode sense the conspecific's signal and move toward the source.

f_{max}. Because the frequency value of an electric fish at time t is strictly related to its closeness to the food source, an individual's frequency value (f_i^t) is derived from its fitness value:

$$f_i^t = f_{min} + \left(\frac{fit_{worst}^t - fit_i{}^t}{fit_{worst}^t - fit_{best}^t} \right) (f_{max} - f_{min}) \tag{2}$$

where fit_{worst}^t and fit_{best}^t are respectively the worst and best fitness values obtained from individuals in the current population at iteration t, whereas fit_i^t is the fitness value of the ith individual at iteration t. In this study, as the frequency value is used for a probability calculation, f_{min} and f_{max} are set to 0 and 1, respectively.

Apart from the frequency, electric fish have also amplitude information. This determines the active range of a fish while actively electrolocating, and the probability of being perceived by other passively electrolocating fish, as the strength of electric field decays with the inverse cube of distance.

The amplitude of an individual depends on the weight of the individual's previous amplitudes (α in Eq. 3), and therefore it may not change sharply. The amplitude value of the ith individual (A_i) is calculated as follows:

$$A_i^t = \alpha A_i^{t-1} + (1-\alpha) f_i^t \tag{3}$$

where $\alpha \mid \alpha \in [0,1]$ is a constant value that determines the magnitude of the previous amplitude value. In EFO, the initial amplitude value of the ith individual is set to its own initial frequency value f_i.

The frequency and amplitude parameter values of a fish (or individual) are updated based on the proximity of the fish to the best prey source. In every iteration of the algorithm, the population is divided into two groups based on the frequency value of each individual: individuals performing active electrolocation (N_A) and those performing passive electrolocation (N_P) (i.e., $N_A \cup N_P = N$). Because the frequency of an individual is compared with a uniformly distributed random value, the higher the frequency value an individual has, the more likely it is to perform active electrolocation. The search is then performed by individuals in N_A and N_P in a parallel manner.

3.3.2 Active electrolocation

The biologically effective range of active electrolocation is limited to approximately half of the fish size, and the fish is unable to perceive prey outside this range. This means that this capability allows fish to locate any food source in their immediate vicinity. The local search or *exploitation* capability of EFO is based on these characteristics of active electrolocation.

In EFO, all individuals in active mode (performing active electrolocation) move around the search space by modifying their traits. However, only one parameter that is randomly chosen is allowed to be modified, lest the individuals move too far away from the promising region.

The movement of the ith individual may vary depending on the existence of neighbors inside its active range. If no neighbor exists (Figure 2-A), it performs a *random walk* throughout its range, otherwise; it chooses one neighbor randomly and changes its location depending on this neighbor (Figure 2-B). The optional search contributes an 'explore first, exploit later' approach in EFO. As the individuals are initially far away from each other, they are less likely to belong to each other's active range. Therefore, it is very likely for active mode individuals to initially search randomly in their neighborhood and then start to exploit one of the closest neighbors as the iteration proceeds.

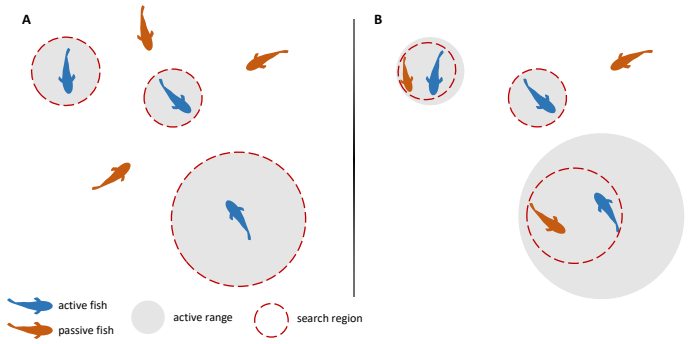

Fig. 2: Local search demonstration of EFO. Individual in active mode either moves somewhere inside its active range (A) or search region (B) depending on the existence of neighbors in its vicinity.

The active range of the ith individual (r_i) is determined by its own amplitude value (A_i), as in nature. The active range calculation in EFO is given in Eq. 4.

$$r_i = \left(x_{\max j} - x_{\min j}\right) A_i \tag{4}$$

To find neighboring individuals ($S \mid S \subset N$) in the sensing/active range, one needs to measure the distance between the ith individual and the rest of the population (i.e., $N \setminus \{i\}$). The distance between individuals i and k is determined by using the Cartesian distance calculation:

$$d_{ik} = \|x_i - x_k\| = \sqrt{\sum_{j=1}^{d} \left(x_{ij} - x_{kj}\right)^2} \tag{5}$$

In the case where at least one neighbor exists in the active sensing area, EFO uses Eq. 6; otherwise (i.e., $S = \emptyset$), Eq. 7 is employed.

$$x_{ij}^{cand} = x_{ij} + \varphi \left(x_{kj} - x_{ij}\right) \tag{6}$$

where k represents a randomly chosen individual from the neighbor set of the ith individual (i.e., $k \in S$ and $d_{ik} \leq r_i$)

$$x_{ij}^{cand} = x_{ij} + \varphi r_i \tag{7}$$

where $\varphi \in [-1, 1]$ in Eq. 6 and 7 is a random number generated from a uniform distribution and x_{ij}^{cand} represents the candidate location of the ith individual.

3.3.3 Passive electrolocation

As opposed to active electrolocation, the sensing distance does not depend on the ith individual and exceeds beyond the range of active electrolocation, as in nature. This is why the passive electrolocation capability fulfills the requirements of the global search or *exploration* mechanism of the proposed EFO algorithm.

As mentioned before, the perceiving probability of a signal is directly proportional to its own amplitude value and the distance to a target individual. Individuals in the passive mode choose other individuals in active mode that propagate electrical signals depending on a probability, and then change their locations (see Figure 3-A).

The probability of the kth individual in active mode (i.e., $k \in N_A$) being perceived by the ith individual in passive mode (i.e., $i \in N_P$) is calculated using Eq. 8.

$$p_k = \frac{A_k / d_{ik}}{\sum_{j \in N_A} A_j / d_{ij}} \tag{8}$$

As seen from the equation, as well as amplitude-based selection, passive electrolocation employs a distance-based selection mechanism as in Gravitational Search Algorithm [60] and Firefly Algorithm [77]. It is important to note here that the probabilistic neighbor selection forces passive mode individuals to perform exploration before the exploitation. Therefore, passive mode individuals choose neighbor individuals at the initial iterations, because the distance is the dominant factor, which causes the selection of poor individuals and which contributes to the discovery of new regions of the search space. Amplitude becomes the dominant factor as the iteration proceeds (because the distance between EFO individuals approaches zero), which causes the selection and local search of the best individuals.

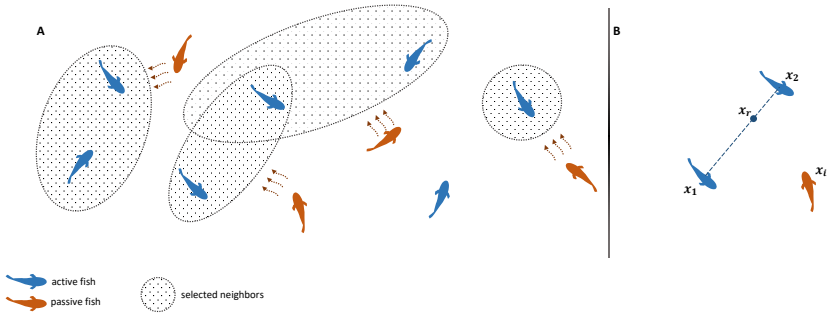

Fig. 3: Global search demonstration of EFO.

Using various strategies, such as roulette wheel selection as employed here, K individuals are chosen from N_A based on Eq. 8, and then a reference location (x_{rj}) is determined based on Eq. 9 (see Fig. 3-B). The new location is then generated through Eq. 10.

$$x_{rj} = \frac{\sum_{k=1}^{K} A_k x_{kj}}{\sum_{k=1}^{K} A_k} \tag{9}$$

$$x_{ij}^{new} = x_{ij} + \varphi\,(x_{rj} - x_{ij}) \tag{10}$$

Contrary to the search in active electrolocation, more than one parameter can be modified, such that individuals explore the search space much faster. However, albeit rarely, there might be a case in which an individual with a higher frequency performs passive electrolocation. In such a case, that individual would lose its location information completely, which is not expected owing to the promising region in which it is located. To avoid this, EFO considers Eq. 11 to determine which parameters will be modified. Within the acceptance condition, the probability for such an individual to modify its whole trait is significantly lowered.

$$x_{ij}^{cand} = \begin{cases} x_{ij}^{new} & rand_j(0,1) > f_i \\ x_{ij} & else \end{cases} \tag{11}$$

where $rand_j(0,1)$ is a uniform random number generated for the jth parameter.

The final step of passive electrolocation is to modify one parameter of the ith individual using Eq. 12 to increase the probability of a trait being changed.

$$x_{ij}^{cand} = x_{\min j} + \varphi(x_{\max j} - x_{\min j}) \qquad rand(0,1) \leq rand(0,1) \tag{12}$$

where $rand(0,1)$ is a random number generated from a uniform distribution.

If the jth parameter value of the ith individual exceeds the boundaries of the search space, it is relocated to the boundary of the space that it exceeds:

$$x_{ij}^{cand} = \begin{cases} x_{\min j} & x_{ij}^{cand} < x_{\min j} \\ x_{ij}^{cand} & x_{\max j} > x_{ij}^{cand} > x_{\min j} \\ x_{\max j} & x_{ij}^{cand} > x_{\max j} \end{cases} \tag{13}$$

The simplified pseudocode of the EFO algorithm and its detailed form are given in Algorithms 1 and 2, respectively. The active electrolocation phase of EFO is presented in lines 7-16, and passive electrolocation is presented in lines 18-23 in Algorithm 2. These phases determine the complexity of the proposed EFO algorithm. The complexity of the active and passive phases is mainly governed by the distance calculation of the individuals (lines 9 and 18). Hence, the time complexities of the active and passive phases are $O(|N_A| \times |N|)$ and $O(|N_P| \times |N_A|)$, respectively. In addition, the algorithm has a complexity that is proportional to $O(|N|)$ for the fitness evaluation of the population (line 26). In short, for one iteration, the time complexity of EFO is $O(|N|)$ and $O(|N|^2)$ for the best (when $|N_A| = 1$) and worst cases (when $|N_A| = |N| - 1$), respectively.

Algorithm 1: Simplified pseudocode of Electric Fish Optimization algorithm.

```
Generate initial population;
Evaluate the quality of the individuals;
repeat
    Split the population into two subpopulations (N_A, N_P) depending on the frequency values of each individual (f);
    Perform active and passive electrolocation for the individuals belonging to N_A and N_P, respectively;
    Update frequency f and amplitude A values for each individual;
until termination criterion is met;
```

Algorithm 2: Detailed pseudocode of Electric Fish Optimization algorithm.

```
Generate initial population N by Eq. 1;
Evaluate fitness value fit of the individual;
Calculate frequency f and amplitude A values of every individual by Eq. 2 and 3;
repeat
    foreach i ∈ N do
        if f_i ≥ rand then                                          // active electrolocation phase
            Randomly choose one parameter j to be modified;
            Calculate active range (r_i) of ith individual;
            Calculate distance of ith individual to other individuals;
            Examine neighbor individuals (S) in the sensing area ;
            if S ≠ ∅ then
                Randomly choose one individual k in the active space;
                Modify jth parameter by Eq. 6;
            else
                Modify jth parameter by Eq. 7;
            end
        else                                                        // passive electrolocation phase
            Considering p values (Eq. 8), probabilistically choose K individuals from actively electrolocating population N_A;
            Modify all parameters of ith individual by Eq. 9, 10 and accept only parameters to be modified considering Eq. 11;
            if rand(0,1) ≤ rand(0,1) then
                Determine one more parameter j to be modified;
                Modify jth parameter by Eq. 12;
            end
        end
        Check for boundaries and apply Eq. 13 for those exceeding search space;
        Evaluate quality of new source and accept if found better;
    end
    Update frequency and amplitude values of the population N;
until termination criterion is met;
```

3.4 Search strategy

A fine trade-off between exploration and exploitation characteristics is very important for the overall efficiency and performance of an algorithm. It is less likely, or maybe impossible, for an algorithm to converge to the optimal solution in the case of too much exploration, which leads to performance degradation. Algorithms that mostly favor exploitation, however, may get stuck in a local optimum, even if they have fast convergence [72, 79]. In addition, an 'explore first, exploit later' approach should also be taken into consideration to further improve the optimization performance [70].

The proposed EFO enables well-balanced exploration and exploitation, owing to a metaphor that relies upon the very nature of electric fish. The search strategy of EFO fundamentally relies on three key factors: *frequency*, *amplitude*, and the *distance* among EFO individuals. A trajectory plot that captures the positions of EFO individuals (10 individuals in total) during 1000 iterations on two-dimensional *Michalewicz* function (F24 in Table A.1) has been used to better explain the search strategy of EFO, as shown in Figure 4.

The frequency-based mode determination in EFO individuals forces better individuals to perform exploitation and other individuals to perform exploration. As seen from the figure, active mode individuals are somewhere closer to the promising regions (local and global optima) throughout the search space, which enables them to behave more exploitatively, and passive mode individuals to discover new regions as they are too far away from the these regions.

Amplitude and distance, however, have a cooperative role in maintaining the 'explore first, exploit later' approach. As seen in the figure, the individuals are initially far away from each other and then approach as the iteration proceeds. Therefore, active mode individuals are more likely to perform random search in their vicinity at the beginning of the search, as they are less likely to belong to each other's active range. They start to exploit one of the closest neighbors as the iteration proceeds. Passive mode individuals, however, choose ordinary individuals at the beginning of the search because of the distance that plays a key role initially, which leads EFO to discover unseen regions. As the search reaches to the final iterations, the elite individuals are more likely to be chosen as the distance approaches zero, which makes the amplitude dominant in the determination of neighbor individuals, which increases the exploitation capability of EFO.

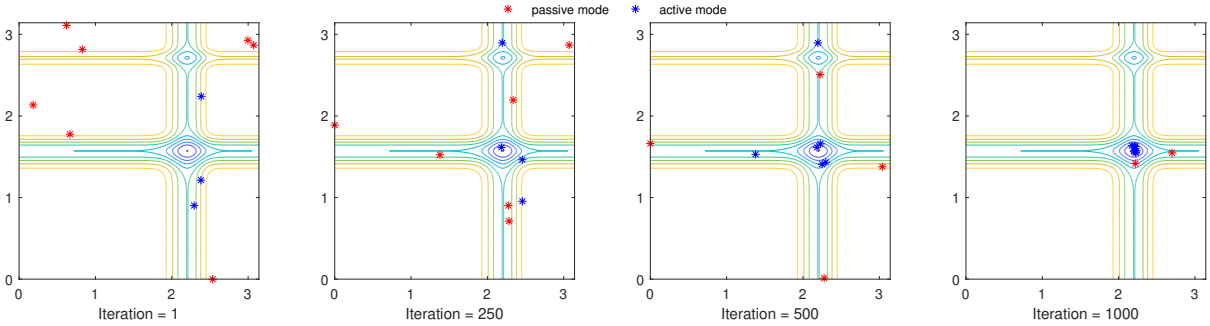

Fig. 4: Trajectory plot tracing the positions of EFO individuals on two-dimensional *Michalewicz* function.

The main differences between EFO and the competitor algorithms in terms of the search strategy are outlined as follows:

- EFO is a population-based algorithm, unlike SA and VS, and it makes use of multiple individuals to find an optimal solution in the search space, which makes EFO a better algorithm for solving complex problems.
- Evolutionary-based GA and DE rely on generic operators such as selection, crossover, and mutation, whereas EFO uses frequency, amplitude, and distance to control the diversity and intensity of the population throughout the search.
- PSO makes use of the particle's and population's best in the previous iterations; EFO, however, takes only the current population into consideration. Moreover, it does not consider only *elite individuals* to ensure better exploration capability.
- Onlooker individuals in ABC take only one neighbor individual at every iteration, whereas in EFO, passive mode individuals rely on more than one neighbors to reduce the likelihood of being trapped in local optima.
- Unlike to ABC and PSO, EFO initially has a much lower tendency to choose individuals with better fitness values, owing to the inclusion of distance in the algorithm to discover the search space efficiently, which also maintains the 'explore first, exploit later' approach.

There are also considerable differences between EFO and competitor algorithms in terms of the optimization performance. The superior search capability of EFO with respect to the competitor algorithms has been proven through the experimental findings, which are presented in the following section.

4 Experiments

To evaluate the optimization capability of the proposed algorithm, three experiments have been conducted and EFO has been compared with a selection of well-known algorithms with a wide range of characteristics (including single-solution-based, population-based, evolutionary-based, and swarm intelligence-based). The first experiment comprises basic and complex mathematical benchmark function sets and the problems in these sets are bound-constrained. The second experiment comprises a clustering problem set; these problems do not possess any type of constraints, including bound constraints. The third experiment, however, comprises well-known constrained real-world design problems, which have both bound and design constraints. The problem sets and competitor algorithms are introduced in detail in the following subsections.

4.1 Benchmark sets

4.1.1 Mathematical benchmark functions

In the first experiment, EFO was tested on two benchmark sets, including a well-known selection of 50 basic and 30 complex mathematical functions, known as the *CEC'17 Benchmark Suite* [5], which were obtained from the studies of Karaboga and Akay [35] and Awad et al. [5], respectively. The details of these problems are provided in Table A.1 and A.2.

The mathematical benchmark functions are divided into six classes depending on their characteristics: unimodal, multimodal, separable, nonseparable, hybrid, and composition. A unimodal function (F1–F17 in Table A.1 and F1–F3 in Table A.2) is a function that contains only one optimum in its search space and it is used to measure how effectively an algorithm performs local search; a multimodal function (F18–F50 in Table A.1 and F4–F10 in Table A.2) has multiple local or global optima and in order for an algorithm to achieve better performance, it must exhibit a better exploration capability. Design parameters are independent of each other in a separable function (F1–F5, F18–F26 in Table A.1),

which means the cost of a separable function is the sum of the costs independently calculated from every parameter. A nonseparable function (F6–F17, F27–F50 in Table A.1) has design parameters that are interrelated to each other, which makes them much more complex compared to separable functions. A hybrid function (F11–F20 in Table A.2) comprises a number of subcomponents that represent the basic function. As for a composition function (F21–F30 in Table A.2), it merges the characteristics of subfunctions to maintain continuity around global/local optima. Heuristics are very successful in solving problems with few parameters. However, they face another problem known as the *curse of dimensionality*, as the parameter size increases and data points in the search space become very sparse [80]. The number of design parameters to be optimized in the basic mathematical benchmark function set varies from 2 to 30, whereas it is 10 for all the complex mathematical benchmark functions. For a list of constant parameters of some functions, refer to [35] and [5].

4.1.2 Clustering problems

The performance of EFO as a clustering technique is investigated in the second experiment. Clustering is a process that aims to determine the optimal cluster centers to gather objects with great similarities into the same cluster or, in other words, to separate different objects from the same cluster as much as possible. Here, the distance measurement generally determines the similarities among objects. The formal definition of the clustering problem, also known as unsupervised learning in the literature, is as follows: given O objects, assign every object into one of C clusters such that it minimizes the Euclidean distance between every object and the center of the cluster to which it belongs:

$$J(w,z) = \sum_{i=1}^{O} \sum_{j=1}^{C} w_{ij} ||x_i - z_j||^2 \tag{14}$$

where x_i and z_j are the locations of the object i and cluster j, respectively. w_{ij} is a weight parameter that takes either 1 (if i belongs to the jth cluster) or 0 (otherwise).

The goal of an optimization algorithm is to find the optimal C cluster centers from an infinite number of candidate cluster centers by minimizing the fitness function. Every individual in EFO represents a candidate solution to the clustering problem, which is actually a vector containing C clusters. The cost function for determining the quality of the ith individual in EFO is adopted as follows:

$$f_i = \frac{1}{D_{Train}} \sum_{j=1}^{D_{Train}} d(x_j, p_i^{CL_{known}(x_j)}) \tag{15}$$

where D_{Train} is the number of instances used for training purpose and $p_i^{CL_{known}(x_j)}$ is the label of the class to which instance x_j belongs.

In this experiment, 13 datasets from a well-known database repository in literature, the UCI repository [10], are used. Each dataset contains actual data collected from different domains. These datasets and their features are listed in alphabetical order in Table 1. In this table, the total number of objects (O) and clusters (C); number of features (F); percentages of continuous, integer, Boolean values (*%num*), and nominal values (*%symb*); flag (*miss*) indicating whether a dataset contains missing attribute values; and percentage of classes (*%major*) to which the majority of instances belongs are listed.

Table 1: Datasets and their properties.

	O	*C*	*F*	*%num*	*%symb*	*miss*	*%major*
Balance	625	3	4	100	0	No	46
Cancer	569	2	30	100	0	No	63
Cancer-Int	699	2	9	100	0	Yes	66
Credit	690	2	51	40	60	Yes	56
Dermatology	366	6	34	97	3	Yes	31
Diabetes	768	2	8	100	0	No	65
E.Coli	327	5	6	88	12	No	43
Glass	214	6	9	100	0	No	36
Heart	303	2	35	100	0	Yes	54
Horse Colic	364	3	58	32	68	Yes	61
Iris	150	3	4	100	0	No	33
Thyroid	215	3	5	100	0	No	70
Wine	178	3	13	100	0	No	40

4.1.3 Real-world design problems

The main purpose of this experiment is to validate the performance of EFO on real-world problems, as most of these problems involve equality and/or inequality constraints. Different approaches have been proposed in the literature to handle design constraints. Because of its simplicity and ease of implementation, the common approach to handle constrained problems is the *penalty method*. The penalty method maps the constrained search space to the unconstrained search space through a penalty function that introduces all the equality and inequality constraints into the fitness function of the problem, such that it penalizes all the constraint violations (see Eq. 16). In this experiment, the penalty method has been employed as a constraint handling method.

$$\min \Pi(X) = f(X) + \sum_{j=1}^{n} \mu_j g_j^2(X) + \sum_{k=1}^{p} \upsilon_k q_k^2(X) \tag{16}$$

In this equation, g and q represent the costs of inequality and equality constraints, respectively, and $\mu(>0)$ and $\upsilon(>0)$ are the weights employed for the inclusion of constraints in the fitness function. These values should be fine-tuned, as an excessively large value considerably increases the complexity of the search space and feasible solutions may not be found throughout the run; an excessively small value, however, leads to a constraint violation problem as is does not yield a considerable cost to the problem fitness. It is empirically observed that EFO is able to yield feasible solutions when μ and υ are taken as 10^4. Therefore, they are set to 10^4 in the experiments. The penalty function is equal to the fitness function, i.e., $\Pi(X) = f(X)$, when a solution found is feasible.

In this experiment, the five most popular real-world engineering benchmark problems have been chosen: *welded-beam*, *spring design*, *pressure vessel*, *three-bar truss design*, and *speed reducer*. These problems are introduced in the following paragraphs.

Welded-beam design problem: The main objective of the welded-beam problem is to manufacture a beam that is made of low-carbon steel (C-1010) and welded to a rigid member within a minimum fabrication cost. This problem was introduced to the literature as a benchmark structural engineering problem by Rao [59]. Figure 5 presents the beam and the member.

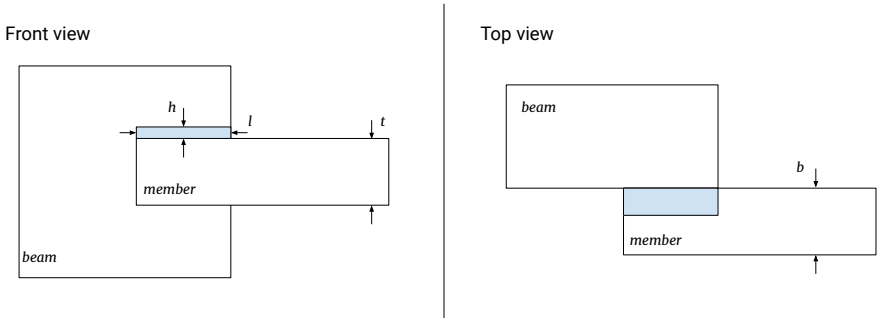

Fig. 5: Welded-beam design.

The problem has four design parameters and five inequality constraints. The parameters to be optimized are h (x_1), the thickness of the weld; l (x_2), the length of the welded joint; t (x_3), the width of the member; and b (x_4), the thickness of the member. The first two variables are discrete and are multiples of 0.0065. EFO is adopted by rounding the values of these parameters to the nearest integer. The bound constraints of these parameter variables are: $0.125 \leq x_1 \leq 5$ and $0.1 \leq x_2, x_3, x_4 \leq 10$.

Spring design problem: The purpose of this design problem is to obtain a minimum-weight spring to achieve its optimal parameter values [3]. The problem with the variables to be optimized is demonstrated in Figure 6.

There are three design variables and four inequality constraints in this problem. The wire diameter d (x_1), the mean diameter D (x_2), and the number of active coils N (x_3) are the design variables of this problem. The bound constraints of these parameters are $0.05 \leq x_1 \leq 1$, $0.25 \leq x_2 \leq 1.3$, and $2 \leq x_3 \leq 15$.

Pressure vessel problem: The pressure vessel problem, where the objective is to minimize the cost value including welding, material, and forming costs, has four design variables and four inequality constraints. The problem was proposed by Sandgren [62]. Figure 7 depicts the schematic of the problem with its design parameters. The thickness of the shell T_s (x_1), the thickness of the head T_h (x_2), the inner radius R (x_3), and the length of the cylindrical section of the vessel L (x_4) are the design variables.

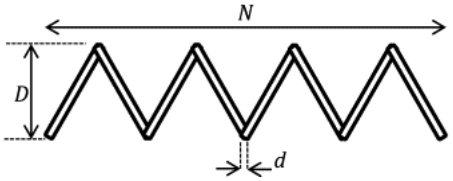

Fig. 6: Spring design problem.

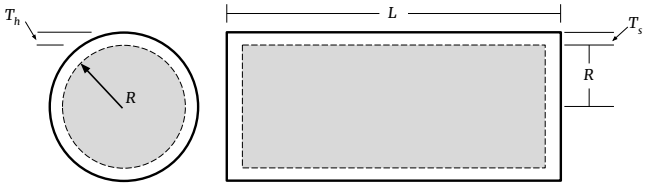

Fig. 7: Pressure vessel design problem.

The bound constraints of these design variables are: $1\times0.0625\leq x_1,x_2\leq99\times0.0625$ and $10\leq x_3,x_4\leq200$. The first two variables (the thicknesses) are discrete and are multiples of 0.0625. As in the welded problem, EFO is adopted here by rounding the values of these parameters to the nearest integer to handle discrete variables.

Three-bar truss design: The three-bar truss design problem is one of the well-known constrained mechanical design problem in the literature, where the objective is to design a three-bar truss through minimizing the volume. The schematic of three-bar truss design problem is given in Figure 8.

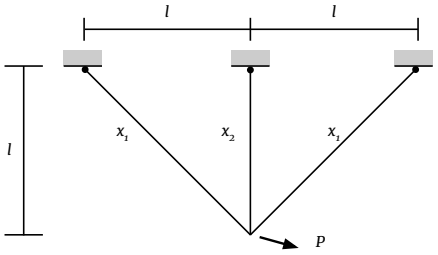

Fig. 8: Three-bar truss design problem.

As shown in the schematic (Fig. 8), there are two parameters to be optimized in this problem, which represent the cross-sectional areas (x_1 and x_2). The bound constraints of these parameters are $0\leq x_1,x_2\leq1$. The problem also has three inequality design constraints.

Speed reducer: The speed reducer problem, proposed by Golisnki [26], represents the design of a gearbox. The objective of this problem is to minimize the total weight of the speed reducer. The schematic of the speed reducer problem is given in Figure 9.

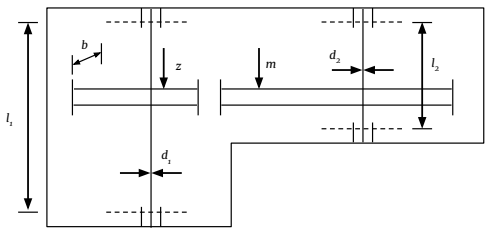

Fig. 9: Speed reducer design problem.

As seen from Fig. 9, there are seven design variables (three of these, from x_2 to x_4, lie at the boundaries of the feasible search space) and eleven design constraints, four of which are linear and seven are nonlinear, which makes this problem a much more challenging benchmark [25]. The face width d (x_1), module of teeth m (x_2), number of teeth in the pinion z (x_3), length of the first shaft between bearings l_1 (x_4), length of the second shaft between bearings l_2 (x_5), diameter of the first shaft d_1 (x_6), and diameter of the second shaft d_2 (x_7) are the design parameters to be optimized. The bound constraints are: $2.6 \leq x_1 \leq 3.6$, $0.7 \leq x_2 \leq 0.8$, $17 \leq x_3 \leq 28$, $7.3 \leq x_4 \leq 8.3$, $7.8 \leq x_5 \leq 8.3$, $2.9 \leq x_6 \leq 3.9$, and $5 \leq x_7 \leq 5.5$.

4.2 Competitor heuristics

In addition to the benchmark set used to prove the optimization success of the algorithm, the choice of competitor algorithms is also very important. Several heuristics have been proposed in the literature. To verify the effectiveness of our proposed contribution, EFO has been compared with well-known heuristics from the literature.

In the first two experiments, EFO was compared with two single-solution-based (SA and VS) and four population-based heuristics (GA, DE, PSO, and ABC); the general characteristics of these heuristics are outlined in Table 2. The pseudocode for these algorithms is provided[1]. It is important to note here that we have intentionally excluded FEO as a competitor heuristic, because its performance is too weak. Speaking concretely, EFO has outperformed FEO in all of the 12 benchmark functions employed in [27].

Table 2: Outline of the heuristics.

	Class		Solution Type	Motivation	Parameter
SA	Single solution-based	Swarm-based	Real-valued	Thermal equilibrium	Initial temperature, cooling factor
VS				Vortex pattern	N.A.
GA	Population-based	Evolution-based	Binary-coded	Survival of the fittest	Crossover and mutation rates
DE			Real-valued		Crossover and scaling factor
PSO		Swarm-based		Cognitive and social experiences	c_1, c_2, and ω
ABC				Scout, employer, and onlooker bees	*limit*
EFO				Active and passive electrolocation	α and K

The final experiment considers only the studies that were recently proposed for solving constrained design problems to determine the success rate of EFO: *i)* Novel approaches such as ABC [1], ISA [24], CSA [4], STA [28], BFO [48], BSA [71], and TLBO [58]; *ii)* modified versions of some heuristics such as TCA [2], PSO [16, 63], FOA [19], DE [50, 51, 83], EA [49, 82]; and *iii)* hybrid approaches such as Genetic Algorithm-Artificial Immune System (GA-AIS) [8, 9], Particle Swarm Optimization-Differential Evolution (PSO-DE) [44], and Cuckoo Search-Genetic Algorithm (CSGA) [33] have been used as competitor heuristics for the problems in this experiment.

4.3 Design of experiments

There are several factors that directly affect the performance of an optimization algorithm, such as the initial condition, problem type, and complexity. The parameter setting of an algorithm is also a crucial factor, and determining the optimal or approximate values of parameters introduces a separate problem (known as a *parameter tuning problem* in the literature) [61]. The parameter settings of each algorithm should optimally be tuned to achieve a fair comparison, which is necessary to ensure the reliability of the experiments. Because finding the optimal parameter values requires very costly large-scale experimentation [65], the design of experiments (DoE) methodology is applied to tune the parameter values, including the population size of each algorithm in this study. The DoE methodology uses a second-order linear model to identify approximations to the optimal parameters.

$$y = \beta_0 + \sum_i \beta_i x_i + \sum_i \sum_{j>i} \beta_{ij} x_i x_j \sum_i \beta_{ii} x_i^2 + \varepsilon \tag{17}$$

where y is a response variable representing the fitness value obtained from the parameter setting x. After the coefficient β is estimated, approximations to the optimal parameter values are obtained by applying quadratic programming.

Because it would be exhaustive to obtain the optimal values for each possible parameter setting, three-level fractional design is employed, where each parameter is set to three levels (referred to as high, intermediate, and low). Function evaluation number (FEN) is employed as the termination criterion of a single run. Every algorithm is run twice with 500,000 and 20,000 FENs for the first and second experiments at each parameter setting, respectively. Therefore, ABC is executed $3^1 \times 2$ times, SA is executed $3^2 \times 2$ times, GA, DE, and EFO are executed $3^3 \times 2$ times, and PSO is executed

Table 3: Approximate optimal parameter values for EFO and competitor algorithms.

Algorithms & Parameters		Value Range	Optimal Parameter Value	
			Experiment 1	Experiment 2
SA				
	Initial temperature	[10, 100]	79.575	73.601
	Cooling factor	[0.01, 0.99]	0.476	0.235
GA				
	Crossover rate	[0.01, 0.99]	0.772	0.842
	Mutation rate	[0.01, 0.20]	0.059	0.011
	Population size	[20, 100]	64	44
DE				
	Crossover rate	[0.01, 0.99]	0.318	0.839
	Scaling factor	[0.01, 0.99]	0.705	0.613
	Population size	[20, 100]	61	43
PSO				
	Magnitude for cognitive best (c_1)	[0.01, 2.00]	1.594	1.291
	Magnitude for social best (c_2)	[0.01, 2.00]	1.526	1.307
	Inertia for old velocity (ω)	[0.01, 0.99]	0.502	0.923
	Population size	[20, 100]	65	67
ABC				
	Population size	[20, 100]	62	40
EFO				
	Magnitude for old amplitude (α)	[0.01, 0.99]	0.416	0.501
	Max. number of neighbor inds. (K)	[1, $\lceil \lvert N \rvert / 2 \rceil$]	20	8
	Population size	[20, 100]	65	20

$3^4 \times 2$ times on each problem. As VS is a single-solution-based and parameter-free algorithm it has been discarded from the parameter tuning. The parameters with their range as well as their optimal values obtained by using the DoE methodology for the first and second experiments are outlined in Table 3. FEN is set to 500,000 and 20,000 for the first and second experiment, respectively. These parameter values of EFO are set according to each problem in the third experiment and they are given in Section 4.4.3. It is important to note here that while the DoE methodology tries to find the optimal values of parameters, it also explores their interactions with other parameters. However such interaction effects have not been explicitly investigated in both the proposed EFO algorithm and its competitor algorithms in this study. For a heuristic algorithm to be regarded as robust, its performance should be less sensitive to differences in the problem characteristic and tuning parameters [7]. The empirical findings from 50 basic benchmark functions reveal that EFO is not dependent on parameter settings and shows similar performance on problems with different characteristics. These results prove the robustness of the proposed EFO algorithm. Therefore, even though the DoE methodology was applied for finding the optimal parameters in this study, it is believed that any parameter settings (other than the optimal parameters) could be employed to yield sufficiently feasible solutions for the problem of interest.

4.4 Results

4.4.1 Experiment 1 (on bound-constrained mathematical problems)

In this experiment, each algorithm has been run 30 times and the best fitness value obtained from each run has been considered when calculating the mean, deviation, and best values. The *mean* values provide insight into the overall performance of the algorithm. Owing to their stochastic nature, heuristics can produce different results each time, and thus the *standard deviation* is a good indicator to reveal the extend to which an algorithm can produce similar results. The *best* value is obtained from the mean values of several runs, indicating the best performance obtainable. All algorithms have been run in parallel using a server machine (24 core Intel® Xeon CPU, 48 GB RAM).

Some functions, such as Easom (F7 in Table A.1), have a valley-type global minimum which is very small compared to their entire search space. Heuristics seek the search space considering the fitness quality of the population. However, such functions do not provide fitness information, because the population is initially far away from the narrow global minimum. This case has also been observed in our experiments: the standard deviation of the fitness values of the population is very close to 0. To avoid this, the frequency of each individual is set to a random value (i.e., $f = rand(0,1)$ when $\sigma(fit(N)) < \varepsilon$). The ε value is set to 10^{-5} in this study.

The mean, standard deviation, and best values obtained from both the basic and complex mathematical benchmark functions are provided in detail[1]. Although these results shed light on the performance of the algorithms, a pairwise statistical test has also been employed to further investigate any significant differences between the algorithms. Pairwise tests are divided into two categories, *problem-based* and *multiproblem-based*. Problem-based testing considers the best results obtained from several runs of an algorithm against a particular problem, whereas multiproblem-based testing considers the average of the best results obtained from several runs. Problem-based pairwise comparisons have been widely used to compare the performance of two given algorithms [14, 17]. The problem-based Wilcoxon Signed Rank

Test at a statistical significance level of 95% is employed in this experiment. The problem-based statistical results for the basic and complex mathematical benchmark functions are provided in Table 4 and 5, respectively.

In this study, the researcher's null hypothesis is that there is no statistically significant difference between the median of the results produced by EFO and its competitor algorithm (C); i.e., median (EFO) = median (C). To speculate that EFO performs much better on a given problem, the null hypothesis should be rejected or an alternative hypothesis should be accepted. T+ and T- values in Table 4 and 5 are the sum of the ranks for the problem in which the competitor algorithm outperforms EFO and is inferior to EFO, respectively. These values are examined to reject or to accept [17]. The (+/-/=) marks in column W are used to show statistical findings. A '+' mark indicates a case where the null hypothesis is rejected and EFO exhibited superior performance, whereas a '-' mark indicates a case where the null hypothesis is accepted and EFO exhibited inferior performance; the '=' mark indicates no statistical finding to reject or to accept the null hypothesis and EFO performs similar performance with its competitor. The last row of the table shows the total count (+/-/=) of the cases.

Table 6 shows problem-type-classified statistical results of the first experiment. As in most modern development tools, values below 10^{-16} are accepted as 0 during the statistical pairwise comparison in this study, because an arithmetic precision that is higher than necessary makes it difficult to compare the local search abilities of the algorithms [14]. From the problem-type statistical results for the basic benchmark functions in Table 6, it is seen that EFO outperforms all its competitors without exception, which is also the case for the complex benchmark functions, with the only exception that DE outperforms EFO on the eight problems overall. It is also seen that EFO obtains overwhelmingly better results on the *multimodal*, *hybrid* (except DE), and *composition* (except ABC) functions, and it shows similar or better performance on *unimodal* functions. The experiment results support that EFO does not face a *curse of dimensionality* problem due to its better performance on problems with high dimensions (15 out of 50 basic mathematical problems). Moreover, EFO has shown a better or comparative performance on hard problems such as *Perm* and *Power Sum* problems, where the global optimal solutions lie at the boundaries of the search space.

In addition to the Wilcoxon's sign test, the VIKOR (Vise Kri-terijumska Optimizacija I Kompromisno Resenje) [55] and TOPSIS (Technique for Order Preference by Similarity to an Ideal Solution) [31] ranking methods are applied for the performance evaluation of EFO and its competitor heuristics on basic and complex mathematical benchmark problems. VIKOR and TOPSIS are the multicriteria decision-making (MCDM) methods considering the ranking of alternatives. They make use of a decision matrix, weight vector, and real number (v) to calculate rankings. In this study, a $7\times TPS$ decision matrix is given, in which rows represent the heuristic methods (alternatives) and columns represent the mean performance obtained through a heuristic on the problem (criteria). In addition, a $1\times TPS$ weight vector is given. Because each problem is given equal weight, each value in this vector is equal to $1/TPS$. Finally, v is set to 0.5 in the ranking calculation. Here, TPS represents total problem size, which is equal to 50 and 30 for the basic and complex problem sets, respectively. Table 7 shows the rankings of EFO and its competitor heuristics evaluated separately on the basic and complex benchmark functions. The VIKOR and TOPSIS ranking values that are evaluated on the basic benchmark set and the VIKOR ranking values that are evaluated on the complex benchmark sets support the superior performance of EFO over its competitors as well. However, according to the TOPSIS ranking values, EFO showed inferior performance to DE but much better than the other competitors on the complex benchmark function set.

Table 4: Wilcoxon Sign Test comparison of the algorithms for basic mathematical benchmark functions.

No	EFO vs. SA				EFO vs. VS				EFO vs. GA				EFO vs. DE				EFO vs. PSO				EFO vs. ABC			
	p-value	T+	T-	W	p-value	T+	T-	W	p-value	T+	T-	W	p-value	T+	T-	W	p-value	T+	T-	W	p-value	T+	T-	W
F1	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=
F2	0	0	0	=	9.330e-06	0	325	+	1.729e-06	0	465	+	0	0	0	=	0	0	0	=	0	0	0	=
F3	1.734e-06	0	465	+	0	0	0	=	1.734e-06	0	465	+	0	0	0	=	0	0	0	=	1.734e-06	0	465	+
F4	1.734e-06	0	465	+	0.0003	0	153	+	1.734e-06	0	465	+	0	0	0	=	0	0	0	=	1.734e-06	0	465	+
F5	1.238e-05	445	20	-	0.0006	400	65	-	1.734e-06	0	465	+	2.127e-06	2	463	+	1.734e-06	465	0	-	1.734e-06	0	465	+
F6	1.734e-06	0	465	+	0	0	0	=	0	0	0	=	0	0	0	=	0.0833	0	6	=	1.734e-06	0	465	+
F7	2.517e-06	0	276	+	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=
F8	1.734e-06	0	465	+	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=	0.0015	0	91	+
F9	1.734e-06	0	465	+	1.734e-06	465	0	-	2.353e-06	3	462	+	1.734e-06	465	0	-	1.734e-06	465	0	-	1.734e-06	0	465	+
F10	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=
F11	0	0	0	=	0	0	0	=	1.734e-06	0	465	+	0	0	0	=	0	0	0	=	0	0	0	=
F12	1.734e-06	0	465	+	0	0	0	=	1.734e-06	0	465	+	0	0	0	=	0	0	0	=	1.734e-06	0	465	+
F13	1.734e-06	0	465	+	1.734e-06	0	465	+	1.734e-06	0	465	+	1.734e-06	0	465	+	1.734e-06	465	0	-	0.0117	355	110	-
F14	1.734e-06	0	465	+	1.734e-06	0	465	+	1.734e-06	0	465	+	0	0	0	=	0	0	0	=	1.734e-06	0	465	+
F15	1.734e-06	465	0	-	1.734e-06	465	0	-	1.734e-06	0	465	+	1.734e-06	0	465	+	1.734e-06	465	0	-	2.353e-06	3	462	+
F16	3.182e-06	6	459	+	1.734e-06	0	465	+	1.734e-06	0	465	+	1.734e-06	0	465	+	1.921e-06	1	464	+	0.0656	143	322	=
F17	3.515e-06	7	458	+	1.734e-06	0	465	+	1.734e-06	0	465	+	1.734e-06	0	465	+	1.734e-06	0	465	+	1.734e-06	465	0	-
F18	4.320e-08	0	465	+	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=	0.3173	0	1	=
F19	0.3173	7	2	=	0.6547	9	6	=	0.7055	12	16	=	1.0000	10	10	=	0.3173	7	2	=	1.0000	5	5	=
F20	1.734e-06	0	465	+	0	0	0	=	4.320e-08	0	465	+	0	0	0	=	0	0	0	=	0	0	0	=
F21	1.734e-06	0	465	+	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=
F22	1.734e-06	0	465	+	1.728e-06	0	465	+	1.734e-06	0	465	+	1.734e-06	0	465	+	1.717e-06	0	465	+	0	0	0	=
F23	1.734e-06	0	465	+	1.725e-06	0	465	+	1.734e-06	0	465	+	0.0003	0	153	+	1.733e-06	0	465	+	0	0	0	=
F24	1.526e-05	300	0	-	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=
F25	3.773e-06	406	0	-	5.664e-05	0	231	+	0	0	0	=	0.3173	0	1	=	1.543e-05	0	300	+	0	0	0	=
F26	1.733e-06	465	0	-	1.734e-06	0	465	+	1.734e-06	0	465	+	0.1797	0	3	=	1.734e-06	0	465	+	0	0	0	=
F27	1.656e-06	0	465	+	0	0	0	=	1.013e-07	0	465	+	0.1573	0	3	=	0.1573	0	3	=	0	0	0	=
F28	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=
F29	1.207e-06	0	465	+	0	0	0	=	4.320e-08	0	465	+	0	0	0	=	0	0	0	=	0	0	0	=
F30	1.734e-06	0	465	+	0	0	0	=	7.752e-07	0	465	+	0	0	0	=	0	0	0	=	1.335e-05	0	300	+
F31	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=
F32	0.0005	0	120	+	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=
F33	1.734e-06	0	465	+	1.734e-06	465	0	-	0.0495	328	137	-	3.112e-05	435	30	-	3.112e-05	435	30	-	0.0017	80	385	+
F34	2.399e-06	0	435	+	0.5930	45	60	=	0.0021	5	100	+	0.1967	40	80	=	2.006e-05	1	298	+	0.2482	26	52	=
F35	4.671e-06	6	429	+	0.3458	66	104	=	0.0904	63	147	=	0.7815	42	49	=	0.0002	20	280	+	0.4795	22	13	=
F36	2.503e-06	0	435	+	0.7963	64	56	=	0.1177	20	58	=	0.0339	31	4	-	0.0005	9	181	+	0.5271	33	22	=
F37	1.734e-06	0	465	+	0.0039	373	92	-	0.1156	156	309	=	1.127e-05	446	19	-	0.3185	281	184	=	4.286e-06	9	456	+
F38	1.921e-06	1	464	+	1.734e-06	465	0	-	0.0034	375	90	-	0.1414	304	161	=	2.879e-06	460	5	-	9.316e-06	17	448	+
F39	1.652e-05	1	230	+	1.0000	10	10	=	0.3173	7	2	=	0.3173	7	2	=	0.6547	9	6	=	0.6547	9	6	=
F40	3.258e-07	0	465	+	0.0041	6	85	+	0.0019	12	141	+	0.2488	25	53	=	0.0004	9	181	+	0.2568	20	8	=
F41	1.734e-06	0	465	+	1.884e-06	1	464	+	1.734e-06	0	465	+	0.1088	6	0	=	0.0018	5	115	+	0.0396	25	95	+
F42	1.734e-06	0	465	+	3.962e-05	432	33	-	1.734e-06	0	465	+	1.734e-06	0	465	+	2.308e-06	434	1	-	6.306e-06	451	14	-
F43	1.734e-06	0	465	+	5.896e-05	0	231	+	1.734e-06	0	465	+	0	0	0	=	0.0231	0	21	+	1.734e-06	0	465	+
F44	1.734e-06	0	465	+	0	0	0	=	1.734e-06	0	465	+	0	0	0	=	0	0	0	=	1.734e-06	0	465	+
F45	0.0009	0	105	+	0.5637	2	4	=	0.4142	7	14	=	1.0000	5	5	=	0.0005	8	163	+	0.1797	3	12	=
F46	9.712e-07	0	435	+	0.3173	0	1	=	0.3173	0	1	=	0	0	0	=	0.0001	0	190	+	0	0	0	=
F47	4.175e-07	0	465	+	0.1609	101	199	=	2.443e-06	0	435	+	0.0455	220	80	-	0.0009	34	266	+	0.3738	15	30	=
F48	1.734e-06	0	465	+	0	0	0	=	6.907e-07	0	465	+	0	0	0	=	0	0	0	=	0	0	0	=
F49	1.734e-06	0	465	+	0.0023	381	84	-	1.921e-06	1	464	+	0.1650	300	165	=	0.0036	91	374	+	0.0008	70	395	+
F50	1.360e-05	21	444	+	0.2712	286	179	=	1.921e-06	1	464	+	4.729e-06	10	455	+	6.339e-06	13	452	+	0.5577	204	261	=
+/=/-	38/7/5				13/29/8				29/19/2				9/36/5				17/26/7				17/30/3			

Table 5: Wilcoxon Sign Test comparison of the algorithms for complex mathematical benchmark functions.

No	EFO vs. SA				EFO vs. VS				EFO vs. GA				EFO vs. DE				EFO vs. PSO				EFO vs. ABC			
	p-value	T+	T-	W	p-value	T+	T-	W	p-value	T+	T-	W	p-value	T+	T-	W	p-value	T+	T-	W	p-value	T+	T-	W
F1	1.921e-06	1	464	+	1.921e-06	1	464	+	1.734e-06	0	465	+	1.732e-06	465	0	-	2.353e-06	3	462	+	3.112e-05	30	435	+
F2	0	0	0	=	0	0	0	=	0	0	0	=	0	0	0	=	0.3173	0	1	=	0	0	0	=
F3	0.0041	55	0	-	0.0041	55	0	-	1.733e-06	0	465	+	0.0041	55	0	-	0.0041	55	0	-	1.734e-06	0	465	+
F4	0.0502	137	327	=	0.1413	150	285	=	2.561e-06	0	435	+	1.909e-06	1	464	+	0.1324	137	269	=	0.9038	184	194	=
F5	1.733e-06	0	465	+	5.276e-05	36	429	+	8.457e-06	16	449	+	0.1589	301	164	=	1.733e-06	0	465	+	0.0086	96	339	+
F6	1.734e-06	0	465	+	0.0006	0	120	+	1.711e-06	0	465	+	0	0	0	=	8.298e-06	0	351	+	0	0	0	=
F7	1.734e-06	0	465	+	2.603e-06	4	461	+	3.182e-06	6	459	+	4.860e-05	35	430	+	1.734e-06	0	465	+	0.0719	145	320	=
F8	1.729e-06	0	465	+	1.722e-06	0	465	+	2.351e-06	3	462	+	0.0786	147	318	=	1.720e-06	0	465	+	1.668e-05	18	416	+
F9	1.734e-06	0	465	+	0	0	0	=	1.702e-06	0	465	+	0	0	0	=	0.0033	0	66	+	0.0339	0	15	+
F10	1.734e-06	0	465	+	1.127e-05	19	446	+	0.0117	110	355	+	4.860e-05	430	35	-	1.734e-06	0	465	+	0.7655	218	247	=
F11	1.734e-06	0	465	+	2.124e-06	2	463	+	2.592e-06	4	461	+	2.529e-06	435	0	-	1.920e-06	1	464	+	0.2384	163	272	=
F12	0.8451	223	242	=	0.0495	137	328	+	0.1306	159	306	=	1.734e-06	465	0	-	0.3389	279	186	=	0.0041	93	372	+
F13	1.734e-06	0	465	+	1.734e-06	0	465	+	2.843e-05	29	436	+	1.734e-06	465	0	-	1.127e-05	19	446	+	0.0207	345	120	-
F14	1.734e-06	0	465	+	0.0028	378	87	-	5.787e-05	428	37	-	1.734e-06	465	0	-	0.2369	290	175	=	0.3600	188	277	=
F15	1.734e-06	0	465	+	2.369e-05	27	438	+	0.0285	126	339	+	1.731e-06	465	0	-	0.1650	165	300	=	0.4165	193	272	=
F16	1.734e-06	0	465	+	1.731e-06	0	465	+	3.670e-06	7	457	+	2.035e-06	463	2	-	1.733e-06	0	465	+	0.0286	107	299	+
F17	1.734e-06	0	465	+	1.732e-06	0	465	+	1.921e-06	1	464	+	2.266e-05	365	13	-	1.734e-06	0	465	+	0.0197	92	286	+
F18	1.734e-06	0	465	+	1.734e-06	0	465	+	2.879e-06	5	460	+	1.734e-06	465	0	-	0.0196	119	346	+	0.1414	161	304	=
F19	2.353e-06	3	462	+	0.1650	300	165	=	0.0015	387	78	-	1.734e-06	465	0	-	0.0012	390	74	-	0.0256	341	124	-
F20	1.734e-06	0	465	+	1.732e-06	0	465	+	2.152e-06	2	462	+	0.8907	7	8	=	1.734e-06	0	465	+	0.8539	4	5	=
F21	2.127e-06	2	463	+	2.863e-06	460	5	-	0.0001	46	418	+	3.880e-06	8	457	+	1.972e-05	25	440	+	0.0817	137	298	=
F22	1.734e-06	0	465	+	2.127e-06	463	2	-	1.733e-06	0	465	+	7.230e-06	10	425	+	2.163e-05	26	439	+	0.1846	297	168	=
F23	1.734e-06	0	465	+	0.0350	130	335	+	0.0002	54	411	+	0.0243	342	123	-	1.734e-06	0	465	+	0.1414	161	304	=
F24	7.033e-06	2	376	+	0.8658	13	15	=	1.734e-06	0	465	+	1.730e-06	0	465	+	1.390e-05	1	324	+	0.0844	61	17	=
F25	0.0002	54	411	+	0.0002	34	344	+	2.554e-06	0	435	+	0.0022	61	316	+	4.256e-06	9	456	+	4.281e-06	456	9	-
F26	1.734e-06	0	465	+	1.649e-06	0	465	+	1.733e-06	0	465	+	1.671e-06	0	465	+	6.161e-06	1	377	+	0.2584	120	204	=
F27	1.921e-06	1	464	+	0.4528	269	196	=	0.1731	154	280	=	1.729e-06	465	0	-	1.732e-06	0	465	+	0.5038	265	200	=
F28	3.878e-06	0	378	+	0.0117	0	36	+	1.731e-06	0	465	+	3.631e-06	0	406	+	5.723e-05	0	231	+	0.9176	66	70	=
F29	1.734e-06	0	465	+	0.8612	224	241	=	0.0207	120	345	+	0.0017	385	80	-	1.921e-06	1	464	+	0.0175	348	117	-
F30	0.8612	241	224	=	0.1359	305	160	=	0.0545	139	326	=	2.370e-05	438	27	-	0.1470	303	162	=	0.0018	384	81	-
+/=/-	25/4/1				18/8/4				24/4/2				8/6/16				22/6/2				8/17/5			

Table 6: Problem-type-based statistical comparison of EFO for mathematical benchmark functions.

Problem Type	EFO vs. SA	EFO vs. VS	EFO vs. GA	EFO vs. DE	EFO vs. PSO	EFO vs. ABC
Basic benchmark functions						
US	2/2/1	2/2/1	4/1/0	1/4/0	0/4/1	3/2/0
UN	9/2/1	4/6/2	8/4/0	4/7/1	2/7/3	6/4/2
MS	4/1/3	4/4/0	4/4/0	2/6/0	4/4/0	0/8/0
MN	21/2/0	3/15/5	12/9/2	2/17/4	11/9/3	8/14/1
Total (+/=/-)	38/7/5	13/29/8	29/19/2	9/36/5	17/26/7	17/30/3
Complex benchmark functions						
U	1/1/1	1/1/1	2/1/0	0/1/2	1/1/1	2/1/0
M	6/1/0	5/2/0	7/0/0	2/4/1	6/1/0	3/4/0
H	9/1/0	8/1/1	7/1/2	0/1/9	6/3/1	3/5/2
C	9/1/0	4/4/2	8/2/0	6/0/4	9/1/0	0/7/3
Total (+/=/-)	25/4/1	18/8/4	24/4/2	8/6/16	22/6/2	8/17/5

Table 7: Ranking scores evaluated using VIKOR and TOPSIS methods for basic and complex mathematical benchmark functions

Methods	**Basic benchmark functions**				**Complex benchmark functions**			
	VIKOR		**TOPSIS**		**VIKOR**		**TOPSIS**	
	Q_j	**Rank**	R_j	**Rank**	Q_j	**Rank**	R_j	**Rank**
SA	0.5000	6	0.4946	7	0.9355	7	0.4187	7
VS	0.0781	4	0.8455	3	0.3760	2	0.8163	3
GA	0.3468	5	0.5364	6	0.5399	5	0.6604	6
DE	0.0351	3	0.8327	4	0.4694	4	0.9749	1
PSO	0.6598	7	0.7680	5	0.6308	6	0.6920	5
ABC	0.0154	2	0.9338	2	0.4521	3	0.7328	4
EFO	0.0000	1	0.9879	1	0.0000	1	0.9113	2

The convergence rate indicates how long it takes for an algorithm to find a solution. The problem known as *premature convergence* occurs when an algorithm converges to a local optimum and is no longer able to explore other parts of the search space [73]. To reveal the comparative convergence capability of EFO, the change in fitness value obtained from basic and complex mathematical benchmark functions with respect to the iteration is comparatively shown in Figures B.1 to B.3. Because SA is not an iteration-based heuristic, it is excluded from all these figures. The convergence behaviors demonstrate that EFO showed better or at least highly competitive performance on the problem set with respect to the convergence speed and also that VS, which showed a much more compelling performance than most of the other competitors, exhibited much slower convergence behavior on almost all the problems.

4.4.2 Experiment 2 (on unconstrained clustering problems)

Contrary to the first experiment, where the benchmark itself directly measures the performance of an algorithm, this experiment comprises two phases: *training* and *testing*. We have used the first 75% of every dataset for the training purpose and the last 25% for the testing purpose.

The first phase of this experiment, training, is crucially important for evaluating the learning capability of the algorithm. It should be noted that, as in [21] and [38], the purpose of the clustering in this study is to learn the optimal clustering centers with a priori knowledge of the number of clusters. However, there are also studies [6, 23] in the literature that aim to optimize also the number of clusters. Optimal cluster centers are learned through the training phase according to Eq. 15.

Testing reveals how well an algorithm classifies instances taken as a subset of the whole dataset. In the testing phase, every instance in the test dataset is assigned to a class depending the Euclidean distances to the cluster centers learned during the training phase. To evaluate the clustering performance of EFO, we have used the classification error percentage (CEP), the percentage of misplaced instances of the whole test dataset:

$$CEP = \frac{\#\ of\ misplaced\ instances}{total\ number\ of\ test\ instances} \tag{18}$$

The comparative CEP and ranking values obtained from the testing phase are shown in Table 8. The results indicate that EFO performs best on 11 out of 13 problems. The comparative findings of the first two experiments strongly support the NFL [74] theorem. GA that performs worse in the first experiment shows very compelling performance on the clustering problems, whereas VS and PSO that are very competitive in the first experiment are inferior in this problem. To reveal the overall performance of the algorithms, the average CEP values calculated from all the problems

and general ranking based on the average values are provided in Table 9. Upon close inspection of the average CEP values and the rankings, it can be seen that EFO has the best performance overall.

Table 8: Classification error percentages and rankings (within parenthesis) of the algorithms.

Problem	SA	VS	GA	DE	PSO	ABC	EFO
Balance	25.64 (2)	57.05 (3)	25 (1)	25 (1)	25.64 (2)	25 (1)	25 (1)
Cancer	6.34 (3)	31.69 (5)	4.93 (1)	5.63 (2)	7.04 (4)	4.93 (1)	4.93 (1)
Cancer-int	2.86 (1)	6.86 (2)	2.86 (1)	2.86 (1)	2.86 (1)	2.86 (1)	2.86 (1)
Credit	21.51 (1)	49.42 (5)	25 (2)	26.16 (3)	26.16 (3)	40.7 (4)	26.16 (3)
Dermatology	6.52 (2)	65.22 (7)	23.91 (6)	7.61 (3)	8.7 (4)	18.48 (5)	5.43 (1)
Diabetes	22.92 (2)	63.02 (3)	22.92 (2)	22.4 (1)	22.92 (2)	22.4 (1)	22.4 (1)
E-coli	14.63 (1)	93.9 (2)	14.63 (1)	14.63 (1)	15.85 (1)	14.63 (1)	14.63 (1)
Glass	39.62 (2)	88.68 (5)	41.51 (3)	37.74 (1)	52.83 (4)	52.83 (4)	37.74 (1)
Heart	17.11 (1)	38.16 (3)	18.42 (2)	17.11 (1)	18.42 (2)	17.11 (1)	17.11 (1)
Horse	43.96 (5)	46.15 (6)	40.66 (3)	42.86 (4)	32.97 (1)	36.26 (2)	42.86 (4)
Iris	2.63 (1)	84.21 (2)	2.63 (1)	2.63 (1)	2.63 (1)	2.63 (1)	2.63 (1)
Thyroid	5.66 (1)	26.42 (2)	5.66 (1)	5.66 (1)	5.66 (1)	5.66 (1)	5.66 (1)
Wine	4.44 (1)	51.11 (3)	4.44 (1)	4.44 (1)	6.67 (2)	4.44 (1)	4.44 (1)
# of being 1st ranking	7	0	7	9	5	9	11

Table 9: Average classification error percentages and ranking of the algorithms.

	SA	VS	GA	DE	PSO	ABC	EFO
Average CEP	16.45	53.99	17.89	16.52	17.57	19.07	16.30
Rank	2	7	5	3	4	6	1

4.4.3 Experiment 3 (on both bound and design constrained real-world problems)

The performance of EFO has been tested through 30 separate runs for every problem introduced in Section 4.1.3, as in the first experiment. A cost function in which fitness function of the problem as well as all its constraints are included through the penalty method has been targeted for optimization in this experiment. The obtained best, average, and worst performances as well as the average search time, population size, and function evaluation numbers are given in Table 10.

Table 10: General performance of EFO measured on the design problems.

						Optimal parameter values			
problem	best	mean	worst	std. dev.	avg. time (s)	α	K	pop. size	FEN
welded-beam	1.8555	2.2719	2.4984	0.1626	17.714	0.664	9	15	30,000
spring design	0.012650	0.012703	0.012789	4.23e-05	10.508	0.990	12	15	30,000
pressure vessel	6,059.60	6,102.87	6,138.60	20.998	12.038	0.989	13	25	50,000
truss design	263.895409	263.8954	263.8955	3.62e-05	5.0707	0.588	5	10	30,000
speed reducer	2,996.243	2,996.243	2,996.243	5.19e-08	1.923	0.989	6	10	10,000

From the statistical findings in Table 10, it can be concluded that EFO showed a consistent performance on the spring design, three-bar truss design, and speed reducer problems, whereas it produced results with much higher deviation on the pressure vessel problem. In spite of the much higher evaluation number used in the pressure vessel problem than that used in the other problems, it is seen that the average time taken for EFO to complete the search process is less than that in the welded-beam problem. This is due to the computational complexity of the constraints of the welded-beam problem.

Detailed comparative best performances of EFO and the approaches already proposed in the literature are presented in Tables 12 to 16 for the welded-beam, spring design, pressure vessel, three-bar truss, and speed reducer problems. The minimum cost obtained, optimal design parameter values to ensure a minimum cost, as well as the FENs to find this cost are provided in these tables. The constraint values of the design problems obtained by the EFO algorithm are given in Table 11.

Table 11: Constraint values of the best solution of design problems obtained by EFO.

	welded-beam	**spring design**	**pressure vessel**	**speed reducer**	**truss design**
g(1)	-22.3465	-0.0003	0.0000	0.0000	-0.0739
g(2)	-959.2553	0.0000	-0.0359	-1.4642	-0.1980
g(3)	-0.0023	-4.0382	-1.5256	-0.5358	-0.4992
g(4)	-3.3271	-0.7326	-63.3700	N.A.	-0.9015
g(5)	-0.1090	N.A.	N.A.	N.A.	0.0000
g(6)	-0.2352	N.A.	N.A.	N.A.	0.0000
g(7)	-2770.9099	N.A.	N.A.	N.A.	-0.7025
g(8)	N.A.	N.A.	N.A.	N.A.	0.0000

Table 11: Constraint values of the best solution of design problems obtained by EFO.

	welded-beam	**spring design**	**pressure vessel**	**speed reducer**	**truss design**
g(9)	N.A.	N.A.	N.A.	N.A.	-0.5833
g(10)	N.A.	N.A.	N.A.	N.A.	-0.0513
g(11)	N.A.	N.A.	N.A.	N.A.	-0.0109

Table 12: Comparative best performance on the *welded-beam design* problem.

Researchers	Method	FEN	$x_1(h)$	$x_2(l)$	$x_3(t)$	$x_4(b)$	Cost
Akay and Karaboga [1]	ABC	30,000	0.2057	3.4704	9.0366	0.2057	1.7249
Aragon et al. [2]	TCA	320,000	0.2444	6.2186	8.2915	0.2444	2.3811
Bernardino et al. [9]	GA-AIS	320,000	0.2444	6.2183	8.2915	0.2444	2.3812
Bernardino et al. [8]	GA-AIS	320,000	0.2443	6.2202	8.2912	0.2444	2.3812
Datta and Figueira [16]	PSO	N.A.	0.1875	1.7821	8.2500	0.2500	1.9553
Gandomi et al. [24]	ISA	30,000	0.2443	6.2199	8.2915	0.2443	2.3812
Gandomi et al. [24]	CSGA	30,000	0.2443	6.2199	8.2915	0.2443	2.3812
Han et al. [28]	STA	60,000	0.2053	3.2603	9.0366	0.2057	1.6956
Kanagaraj et al. [33]	CSGA	25,000	0.2443	6.2175	8.2915	0.2444	2.3809
Montes and Ocana [48]	BFO	48,000	0.2057	3.4711	9.0367	0.2057	2.3868
Wang et al. [71]	BSA	60,000	0.2057	3.4704	9.0366	0.2057	1.7249
Zhang et al. [83]	DE	24,000	0.2444	6.2175	8.2915	0.2444	2.3810
Zhang et al. [82]	EA	28,897	0.2443	6.2201	8.2940	0.2444	2.3816
Present Study	EFO	30,000	0.2340	3.1135	8.5707	0.2362	1.8555

From the comparative cost values in Table 12, it can be seen that EFO showed better performance than TCA [2], GA-AIS [8, 9], PSO [16], ISA [24], and BFO [48], with a much lower FEN. DE [83], EA [82], and CSGA [33] are also inferior to EFO, but with a lower FEN than EFO. ABC [1], STA [28], BSA [71] are better methods for this problem than EFO; however, they all considered the first two parameters as continuous, when in fact they should be discrete and integer multiples of 0.0065 [25].

Table 13: Comparative best performance on the *spring design* problem.

Researchers	Method	FEN	$x_1(d)$	$x_2(D)$	$x_3(N)$	Cost
Akay and Karaboga [1]	ABC	30,000	0.051749	0.358179	11.203763	0.012665
Aragon et al. [2]	TCA	36,000	0.051622	0.355105	11.384534	0.012665
Askarzadeh [4]	CSA	50,000	0.0516890284	0.3567169544	11.2890117993	0.0126652328
Bernardino et al. [9]	GA-AIS	36,000	0.051660806	0.35603234	11.329555	0.012666
Bernardino et al. [8]	GA-AIS	36,000	0.0514305	0.3505298	11.6611924	0.012666
Dos Santos Coelho [63]	Q-PSO	2,000	0.051515	0.352529	11.538862	0.012665
Du et al. [19]	FOA	25,000	0.05206590	0.36570924	10.78621813	0.01267607
Gandomi et al. [24]	ISA	8,000	N.A.	N.A.	N.A.	0.012665
Han et al. [28]	STA	60,000	0.0516800	0.3565001	11.3018335	0.01266534
Mohammed [51]	NDE	24,000	0.051689058	0.35671768	11.28896875	0.01266523
Montes and Ocana [48]	BFO	48,000	0.051825	0.359935	11.107103	0.012671
Wang et al. [71]	BSA	60,000	0.051743	0.358017	11.213187	0.012665
Zhang et al. [83]	DE	24,000	0.05169	0.35672	11.289	0.012665233
Present Study	EFO	30,000	0.051496	0.35248	11.523	0.012650

The results obtained from the spring design problem (see Table 13) reveal that EFO showed superior performance on the spring design problem in comparison to its all competitors. It can also be seen that Q-PSO [63] and ISA [24] yield their results with the lowest FENs, which are 2,000 and 8,000, respectively. NDE [51] and DE [83] used fewer function evaluations than EFO to complete their search processes.

Table 14: Comparative best performance on the *pressure vessel design* problem.

Researchers	Method	FEN	$x_1(T_s)$	$x_2(T_h)$	$x_3(R)$	$x_4(L)$	Cost
Akay and Karaboga [1]	ABC	30,000	0.8125	0.4375	42.098446	176.636596	6,059.7147
Aragon et al. [2]	TCA	80,000	0.8125	0.4375	42.098429	190.787695	6,390.554
Askarzadeh [4]	CSA	250,000	0.8125	0.4375	42.09844539	176.63659855	6,059.7144
Bernardino et al. [9]	GA-AIS	80,000	0.8125	0.4375	42.0973	176.6509	6,059.8546
Bernardino et al. [8]	GA-AIS	36,000	0.8125	0.4375	42.094967	176.67972	6,060.138
Dos Santos Coelho [63]	Q-PSO	8,000	0.8125	0.4375	42.0984	176.6372	6,059.7208
Du et al. [19]	FOA	25,000	0.7804	0.3849	40.3888	199.1172	5,894.5981
Gandomi [24]	ISA	5,000	0.8125	0.4375	42.09845	176.6366	6,059.714
Han et al. [28]	STA	60,000	0.7785	0.3848	40.3389	199.7753	5,886.45436
Mohammed [51]	NDE	20,000	0.8125	0.4375	42.0984	176.63659	6,059.71433

Table 14: Comparative best performance on the *pressure vessel design* problem.

Researchers	Method	FEN	$x_1(T_s)$	$x_2(T_h)$	$x_3(R)$	$x_4(L)$	Cost
Montes and Ocana [48]	BFO	48,000	0.8125	0.4375	42.096394	176.683231	6,060.460
Wang et al. [71]	BSA	40,000	0.8125	0.4375	42.098497	176.635967	6,059.708274
Present Study	EFO	50,000	0.8125	0.4375	42.099	176.6300	6,059.60

From the best cost values obtained from the pressure vessel design problem, as shown in Table 14, it can be concluded that EFO was able to produce the lowest welding, material, and forming costs compared to the other methods, except for FOA [19] and STA [28]. However, both methods considered the first two parameters as continuous. ISA [24] and Q-PSO [63] are the two methods that complete the search with the lowest FENs to produce their costs for this problem.

Table 15: Comparative best performance on the *three-bar truss design* problem.

Researchers	Method	FEN	x_1	x_2	Cost
Askarzadeh [4]	CSA	25,000	0.7886751284	0.4082483080	263.895843
Liu et al. [44]	PSO-DE	17,600	0.7886751347	0.4082482900	263.895843
Mohammed [51]	NDE	4,000	0.7886753196	0.4082477671	263.895843
Wang et al. [71]	BSA	7,500	0.788675	0.408248	263.895843
Zhang et al. [83]	DE	15,000	0.7886751359	0.4082482868	263.895843
Present Study	EFO	30,000	0.7886970484	0.4081771662	263.895409

The results in Table 15 obtained from the three-bar truss design problem suggest that EFO showed better optimization performance than all the competitor methods, but with more function evaluations. Among the competitor methods, BSA [51] showed the fewest function evaluations.

Table 16: Comparative best performance on the *speed reducer design* problem.

Researchers	Method	FEN	$x_1(d)$	$x_2(m)$	$x_3(z)$	$x_4(l_1)$	$x_5(l_2)$	$x_6(d_1)$	$x_7(d_2)$	Cost
Akay and Karaboga [1]	ABC	30,000	3.49	0.7	17	7.3	7.8	3.3502	5.2878	2,997.0584
Bernardino et al. [9]	GA-AIS	36,000	3.5	0.7	17	7.3	7.8	3.3502	5.2866	2,996.3483
Bernardino et al. [8]	GA-AIS	36,000	3.5	0.7	17	7.3	7.7153	3.3502	5.2866	2,994.4712
Kanagaraj et al. [33]	CSGA	25,000	3.5	0.7	17	7.6050	7.8181	3.35	5.2687	2,996.3482
Liu et al. [44]	PSO-DE	54,350	3.5	0.7	17	7.3	7.8	3.3502	5.2866	2,996.3481
Mohammed [51]	NDE	18,000	3.5	0.7	17	7.3	7.7153	3.3502	5.2866	2,994.4710
Montes et al. [49]	EA	36,000	3.5061	0.7	17	7.4601	7.9621	3.3629	5.3089	2,996.3566
Montes et al. [50]	DE	24,000	3.5	0.7	17	7.3	7.8	3.3502	5.2866	2,996.3566
Rao and Vakharia. [58]	TLBO	10,000	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	2,996.3481
Wang et al. [71]	BSA	7,500	3.49	0.7	17	7.3	7.7153	3.3502	5.2866	2,994.4683
Present Study	EFO	10,000	3.5	0.7	17	7.3	7.8	3.3502	5.2865	2,996.2430

As seen from the results in Table 16, EFO showed overwhelmingly better performance than ABC [1], GA-AIS [9], CSGA [33], PSO-DE [44], EA [49], DE [50], and TLBO [58] on the *speed reducer* problem with fewer function evaluations. NDE [51], GA-AIS [8], and BSA [71], however, were shown to be the better methods to solve this problem. It is worth mentioning here that although EFO has become inferior to BSA [71] it showed better performance overall as the mean value yielded by BSA is much higher than that of EFO, at 2,998.0101. Given that the *speed reducer* problem differs from the other design problems in the best feasible solution where two of the parameters lie at the boundaries, it can be concluded that EFO can still competitively handle the problems with such a complex search space.

5 Conclusion

In this study, active and passive electrolocation, a species-specific object/prey detection capability, and electrocommunication capabilities of electric fish are employed for the creation of a novel swarm-based heuristic algorithm, EFO, to solve bound-constrained real parameter optimization problems. To the best knowledge of the authors, this is the first study to propose a heuristic inspired by the electrolocation and electrocommunication capabilities of electric fish, and it comprises very comprehensive experimentation. The proposed EFO is quite simple, with its easy-to-understand structure, few parameters, and two main search frameworks (active and passive search phases).

The authors believe that the optimization and convergence performance of EFO is largely due to the modeled active and passive electrolocation capabilities, which fulfill the "explore first, exploit later" approach that is strongly suggested for heuristics in the literature [60, 70, 14]. As mentioned, the individuals in EFO perform global search in the premier iterations and local search as the iterations lapse, owing to the distance among the population that gradually decreases through the iterations. In active electrolocation, individuals in active mode perform a random search inside their active

search range at the beginning of the iterations, as it is less likely to belong to another's active area, and start to search around their neighborhood toward the end of the iterations. In passive electrolocation, however, the selection of neighbor individuals is highly dependent on the distance between them (see Eq. 8). At the initial phase of the iterations, individuals in passive mode select their neighbors considering the distance rather than their amplitudes. The distance-based selection results in a better global search owing to the higher selection probability of the "poor" individuals. As the iteration proceeds, the amplitude becomes the dominant factor that determines the individuals being selected, which contributes the local search of EFO.

To measure the performance of EFO, well-known single-solution-based (SA and VS) and population-based (GA, DE, PSO, and ABC) heuristics as well as approaches that were recently proposed for constrained problems are employed for three experiments with different characteristics. From the overall optimization performance of EFO, it performs much better than almost all of the other heuristics, with a highly compelling convergence capability. The optimization performance measured on different problem set domains (i.e., bound-constrained mathematical benchmark problems, unconstrained clustering problems, and bound- and design-constrained real-world design problems) has proven the robustness of the proposed EFO algorithm. In addition, it can be understood from the results that EFO has better global search capability and it outperforms all the competitor algorithms on the highly complex problems examined in this study, which is regarded as very promising when considering the complexity of most of the real-world problems. The limitations of the proposed algorithm are outlined as follows, which can be considered for the future directions of this study:

- While the distance-based selection leads EFO to have superior performance on multimodal and complex problems, the complexity of the algorithm is also mainly governed by this distance calculation of individuals.
- Two parameters (magnitude for old amplitude α, maximum number of neighbor individuals K) of the EFO algorithm are need to be set, whereas some algorithms do not even have a single parameter such as VS. Fortunately, the experiments show that EFO is less sensitive to the difference in parameter settings.
- EFO may require some additional efforts to further enhance its search capability for better performance, particularly on the unimodal, hybrid, and composed functions in the first experiment.

In this study, the proposed EFO has been applied to bound-constrained mathematical benchmark functions, unconstrained clustering problems, and real-world design problems comprising bound and complex design constraints. In the future, the performance of EFO on multiobjective problems shall be investigated, as most design optimization problems in engineering possess multiple mutually conflicting design objectives.

Acknowledgment

This work was supported by the National MSc and PhD Scholarship Programme for Senior Undergraduate Students (2228) of the Scientific and Technological Research Council of Turkey (or TUBITAK). The authors appreciate this support. In addition, the authors sincerely acknowledge and thanks to Hacettepe Teknokent Technology Transfer Center for advanced editing service to this article. Finally, the authors would also like to give special thanks to Dr. Bahriye AKAY for her valuable comments on our study.

Appendix A

Table A.1: Mathematical benchmark function set used in the first experiment. D: Parameter size, U: Unimodal, M: Multimodal, S: Separable, N: Nonseparable.

No	Function	D	Type	Min	Formulation
F1	Stepint	5	US	0	$f(x) = 25 + \sum_{i=1}^{5} \lfloor x_i \rfloor$
F2	Step	30	US	0	$f(x) = \sum_{i=1}^{n} \left(\lfloor x_i + 0.5 \rfloor \right)^2$
F3	Sphere	30	US	0	$f(x) = \sum_{i=1}^{n} x_i^2$
F4	Sum Squares	30	US	0	$f(x) = \sum_{i=1}^{n} i x_i^2$
F5	Quartic	30	US	0	$f(x) = \sum_{i=1}^{n} i x_i^4 + random[0,1)$
F6	Beale	5	UN	0	$f(x) = (1.5 - x_1 + x_1 x_2)^2 + \left(2.25 - x_1 + x_1 x_2^2\right)^2 + \left(2.625 - x_1 + x_1 x_2^3\right)^2$
F7	Easom	2	UN	-1	$f(x) = -\cos(x_1)\cos(x_2)\exp\left(-(x_1-\pi)^2 - (x_2-\pi)^2\right)$
F8	Matyas	2	UN	0	$f(x) = 0.26\left(x_1^2 + x_2^2\right) - 0.48 x_1 x_2$
F9	Colville	4	UN	0	$f(x) = 100\left(x_1^2 - x_2\right)^2 + (x_1-1)^2 + (x_3-1)^2 + 90\left(x_3^2 - x_4\right)^2 + 10.1\left((x_2-1)^2 + (x_4-1)^2\right) + 19.8(x_2-1)(x_4-1)$
F10	Trid 6	6	UN	-50	$f(x) = \sum_{i=1}^{n} (x_i - 1)^2 - \sum_{i=2}^{n} x_i x_{i-1}$
F11	Trid 10	10	UN	-210	$f(x) = \sum_{i=1}^{n} (x_i - 1)^2 - \sum_{i=2}^{n} x_i x_{i-1}$
F12	Zakharov	10	UN	0	$f(x) = \sum_{i=1}^{n} x_i^2 + \left(\sum_{i=1}^{n} 0.5 i x_i\right)^2 + \left(\sum_{i=1}^{n} 0.5 i x_i\right)^4$
F13	Powell	24	UN	0	$f(x) = \sum_{i=1}^{n/k} (x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} + x_{4i})^2 + (x_{4i-2} + x_{4i-1})^4 + 10(x_{4i-3} + 10x_{4i})^4$
F14	Schwefel 2.22	30	UN	0	$f(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$
F15	Schwefel1.2	30	UN	0	$f(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2$
F16	Rosenbrock	30	UN	0	$f(x) = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i)^2 + (x_i - 1)^2\right]$
F17	Dixon-Price	30	UN	0	$f(x) = (x_i - 1)^2 + \sum_{i=1}^{n} i\left(2x_i^2 - x_{i-1}\right)^2$
F18	Fox-holes	2	MS	0.998	$f(x) = \left[\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}\left(x_i - a_{ij}\right)^6}\right]^{-1}$
F19	Branin	2	MS	0.398	$f(x) = \left(x_2 - \frac{5.1}{4\pi^2} x_1^2 + \frac{5}{\pi} x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$
F20	Bohachevsky 1	2	MS	0	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$
F21	Booth	2	MS	0	$f(x) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$
F22	Rastrigin	30	MS	0	$f(x) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$
F23	Schwefel	30	MS	-12569.5	$f(x) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{x_i}\right)$
F24	Michalewicz 2	2	MS	-1.8013	$f(x) = -\sum_{i=1}^{n} \sin(x_i)\left(\sin\left(\frac{i x_i^2}{\pi}\right)\right)^{2m}$
F25	Michalewicz 5	5	MS	-4.6877	$f(x) = -\sum_{i=1}^{n} \sin(x_i)\left(\sin\left(\frac{i x_i^2}{\pi}\right)\right)^{2m}$
F26	Michalewicz 10	10	MS	-9.6602	$f(x) = -\sum_{i=1}^{n} \sin(x_i)\left(\sin\left(\frac{i x_i^2}{\pi}\right)\right)^{2m}$
F27	Schaffer	2	MN	0	$f(x) = 0.5 + \frac{\sin^2\left(\sqrt{x_1^2 + x_2^2}\right) - 0.5}{\left(1 + 0.001\left(x_1^2 + x_2^2\right)\right)^2}$
F28	SHCB	2	MN	-1.03163	$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1 x_2 - 4x_2^2 + 4x_2^4$
F29	Bohachevsky2	2	MN	0	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1)(4\pi x_2) + 0.3$
F30	Bohachevsky 3	2	MN	0	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1 + 4\pi x_2) + 0.3$
F31	Shubert	2	MN	-186.73	$f(x) = \left(\sum_{i=1}^{5} i\cos\left((i+1)x_1 + i\right)\right)\left(\sum_{i=1}^{5} i\cos\left((i+1)x_2 + i\right)\right)$
F32	Goldstein-Price	2	MN	3	$f(x) = \left[1 + (x_1 + x_2 + 1)^2 X_1\right]\left[30 + (2x_1 - 3x_2)^2 X_2\right]$ $X_1 = \left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1 x_2 + 3x_2^2\right)$ $X_2 = \left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1 x_2 + 27x_2^2\right)$
F33	Kowalik	4	MN	0.00031	$f(x) = \sum_{i=1}^{11}\left(a_i - \frac{x_1\left(b_i^2 + b_i x_2\right)^2}{b_i^2 + b_i x_3 + x_4}\right)$
F34	Shekel 5	4	MN	-10.15	$f(x) = -\sum_{i=1}^{5}\left[(x - a_i)(x - a_i)^T + c_i\right]^{-1}$
F35	Shekel7	4	MN	-10.4	$f(x) = -\sum_{i=1}^{7}\left[(x - a_i)(x - a_i)^T + c_i\right]^{-1}$
F36	Shekel 10	4	MN	-10.53	$f(x) = -\sum_{i=1}^{10}\left[(x - a_i)(x - a_i)^T + c_i\right]^{-1}$
F37	Perm	4	MN	0	$f(x) = \sum_{k=1}^{n}\left[\sum_{i=1}^{n}\left(i^k + \beta\right)\left(\left(\frac{x_i}{i}\right)^k - 1\right)\right]^2$
F38	Power Sum	4	MN	0	$f(x) = \sum_{k=1}^{n}\left[\left(\sum_{i=1}^{n} x_i^k\right) - b_k\right]^2$
F39	Hartman3	3	MN	-3.86	$f(x) = -\sum_{i=1}^{4} c_i exp\left[-\sum_{j=1}^{3} a_{ij}\left(x_j - p_{ij}\right)^2\right]$
F40	Hartman 6	6	MN	-3.32	$f(x) = -\sum_{i=1}^{4} c_i exp\left[-\sum_{j=1}^{6} a_{ij}\left(x_j - p_{ij}\right)^2\right]$
F41	Griewank	30	MN	0	$f(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$
F42	Ackley	30	MN	0	$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$
F43	Penalized1	30	MN	0	$f(x) = \frac{\pi}{n}\left\{\begin{array}{c}10\sin^2(\pi y_1) + \\ \sum_{i=1}^{n-1}(y_i - 1)^2\left[1 + 10\sin^2(\pi y_{i+1})\right] + \\ (y_n - 1)^2\end{array}\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{1}{4}(x_i + 1)$ $u(x_i, a, k, m) = \left\{\begin{array}{ll} k(x_i - a)^m & x_i > a \\ 0 & -a \le x_i \le a \\ k(-x_i - a)^m & x_i < -a \end{array}\right.$
F44	Penalized 2	30	MN	0	$f(x) = 0.1\left\{\begin{array}{c}\sin^2(\pi x_1) + \\ \sum_{i=1}^{n-1}(x_i - 1)^2\left[11 + \sin^2(3\pi x_{i+1})\right] + \\ (x_n - 1)^2\left[1 + \sin^2(2\pi x_n)\right]\end{array}\right\} + \sum_{i=1}^{n} u(5, 100, 4)$

(continued on next page)

Table A.1 (continued.)

No	Function	D	Type	Min	Formulation
F45	Langerman 2	2	MN	-1.08	$f(x) = -\sum_{i=1}^{m} c_i A_i B_i$ $A_i = \left(\exp\left(-\frac{1}{\pi}\sum_{j=1}^{n}\left(x_j - a_{ij}\right)^2\right)\right)$ $B_i = \cos\left(\pi\sum_{j=1}^{n}\left(x_j - a_{ij}\right)^2\right)$
F46	Langerman 5	5	MN	-1.5	$f(x) = -\sum_{i=1}^{m} c_i A_i B_i$ $A_i = \left(\exp\left(-\frac{1}{\pi}\sum_{j=1}^{n}\left(x_j - a_{ij}\right)^2\right)\right)$ $B_i = \cos\left(\pi\sum_{j=1}^{n}\left(x_j - a_{ij}\right)^2\right)$
F47	Langerman 10	10	MN	N.A.	$f(x) = -\sum_{i=1}^{m} c_i A_i B_i$ $A_i = \left(\exp\left(-\frac{1}{\pi}\sum_{j=1}^{n}\left(x_j - a_{ij}\right)^2\right)\right)$ $B_i = \cos\left(\pi\sum_{j=1}^{n}\left(x_j - a_{ij}\right)^2\right)$
F48	Fletcher Powell 2	2	MN	0	$f(x) = \sum_{i=1}^{n}\left(A_i - B_i\right)^2$ $A_i = \sum_{j=1}^{n}\left(a_{ij}\sin\alpha_j + b_{ij}\cos\alpha_j\right)$ $B_i = \sum_{j=1}^{n}\left(a_{ij}\sin x_j + b_{ij}\cos x_j\right)$
F49	Fletcher Powell 5	5	MN	0	$f(x) = \sum_{i=1}^{n}\left(A_i - B_i\right)^2$ $A_i = \sum_{j=1}^{n}\left(a_{ij}\sin\alpha_j + b_{ij}\cos\alpha_j\right)$ $B_i = \sum_{j=1}^{n}\left(a_{ij}\sin x_j + b_{ij}\cos x_j\right)$
F50	Fletcher Powell 10	10	MN	0	$f(x) = \sum_{i=1}^{n}\left(A_i - B_i\right)^2$ $A_i = \sum_{j=1}^{n}\left(a_{ij}\sin\alpha_j + b_{ij}\cos\alpha_j\right)$ $B_i = \sum_{j=1}^{n}\left(a_{ij}\sin x_j + b_{ij}\cos x_j\right)$

Table A.2: Complex mathematical benchmark set. D: Parameter size, T: Type, Low and Up: Lower and upper limit, U: Unimodal, M: Multimodal, H: Hybrid, C: Composition

No	Function	D	T	Low	Up
F1	Shifted and Rotated *Bent Cigar*	10	U	-100	100
F2	Shifted and Rotated *Sum of Different Power*	10	U	-100	100
F3	Shifted and Rotated *Zakharov*	10	U	-100	100
F4	Shifted and Rotated *Rosenbrock*	10	M	-100	100
F5	Shifted and Rotated *Rastrigin*	10	M	-100	100
F6	Shifted and Rotated *Expanded Scaffer*	10	M	-100	100
F7	Shifted and Rotated *Lunacek Bi-Rastrigin*	10	M	-100	100
F8	Shifted and Rotated *Non-Continuous Rastrigin*	10	M	-100	100
F9	Shifted and Rotated *Levy*	10	M	-100	100
F10	Shifted and Rotated *Schwefel*	10	M	-100	100
F11	Hybrid of *Zakharov*, *Rosenbrock*, and *Rastrigin*	10	H	-100	100
F12	Hybrid of *High Conditioned Elliptic*, *Modified Schwefel* , and *Bent Cigar*	10	H	-100	100
F13	Hybrid of *Bent Cigar*, *Rosenbrock*, and *Lunacek Bi-Rastrigin*	10	H	-100	100
F14	Hybrid of *High Conditioned Elliptic*, *Ackley*, *Schaffer*, and *Rastrigin*	10	H	-100	100
F15	Hybrid of *Bent Cigar*, *HGBat*, *Rastrigin*, and *Rosenbrock*	10	H	-100	100
F16	Hybrid of *Expanded Schaffer*, *HGBat*, *Rosenbrock*, and *Modified Schwefel*	10	H	-100	100
F17	Hybrid of *Katsuura*, *Ackley*, *Expanded Griewank & Rosenbrock*, *Modified Schwefel*, and *Rastrigin*	10	H	-100	100
F18	Hybrid of *High Conditioned Elliptic*, *Ackley*, *Rastrigin*, *HGBat*, and *Discus*	10	H	-100	100
F19	Hybrid of *Bent Cigar*, *Rastrigin*, *Expanded Griewank & Rosenbrock*, *Weierstrass*, and *Expanded Schaffer*	10	H	-100	100
F20	Hybrid of *HappyCat*, *Katsuura*, *Ackley*, *Rastrigin*, *Modified Schwefel*, and *Schaffer*	10	H	-100	100
F21	Composition of *Rosenbrock*, *High Conditioned Elliptic*, and *Rastrigin*	10	C	-100	100
F22	Composition of *Rastrigin*, *Griewank*, and *Modified Schwefel*	10	C	-100	100
F23	Composition of *Rosenbrock*, *Ackley*, *Modified Schwefel*, and *Rastrigin*	10	C	-100	100
F24	Composition of *Ackley*, *High Conditioned Elliptic*, *Girewank*, and *Rastrigin*	10	C	-100	100
F25	Composition of *Rastrigin*, *HappyCat*, *Ackley*, *Discus*, and *Rosenbrock*	10	C	-100	100
F26	Composition of *Expanded Scaffer*, *Modified Schwefel*, *Griewank*, *Rosenbrock*, and *Rastrigin*	10	C	-100	100
F27	Composition of *HGBat*, *Rastrigin*, *Modified Schwefel*, *Bent-Cigar*, *High Conditioned Elliptic*, and *Expanded Scaffer*	10	C	-100	100
F28	Composition of *Ackeley*, *Griewank*, *Discus*, *Rosenbrock*, *HappyCat*, and *Expanded Scaffer*	10	C	-100	100
F29	Composition of *three hybrid functions* (F15, F16, and F17)	10	C	-100	100
F30	Composition Function 10 (*with three functions*)	10	C	-100	100

Appendix B

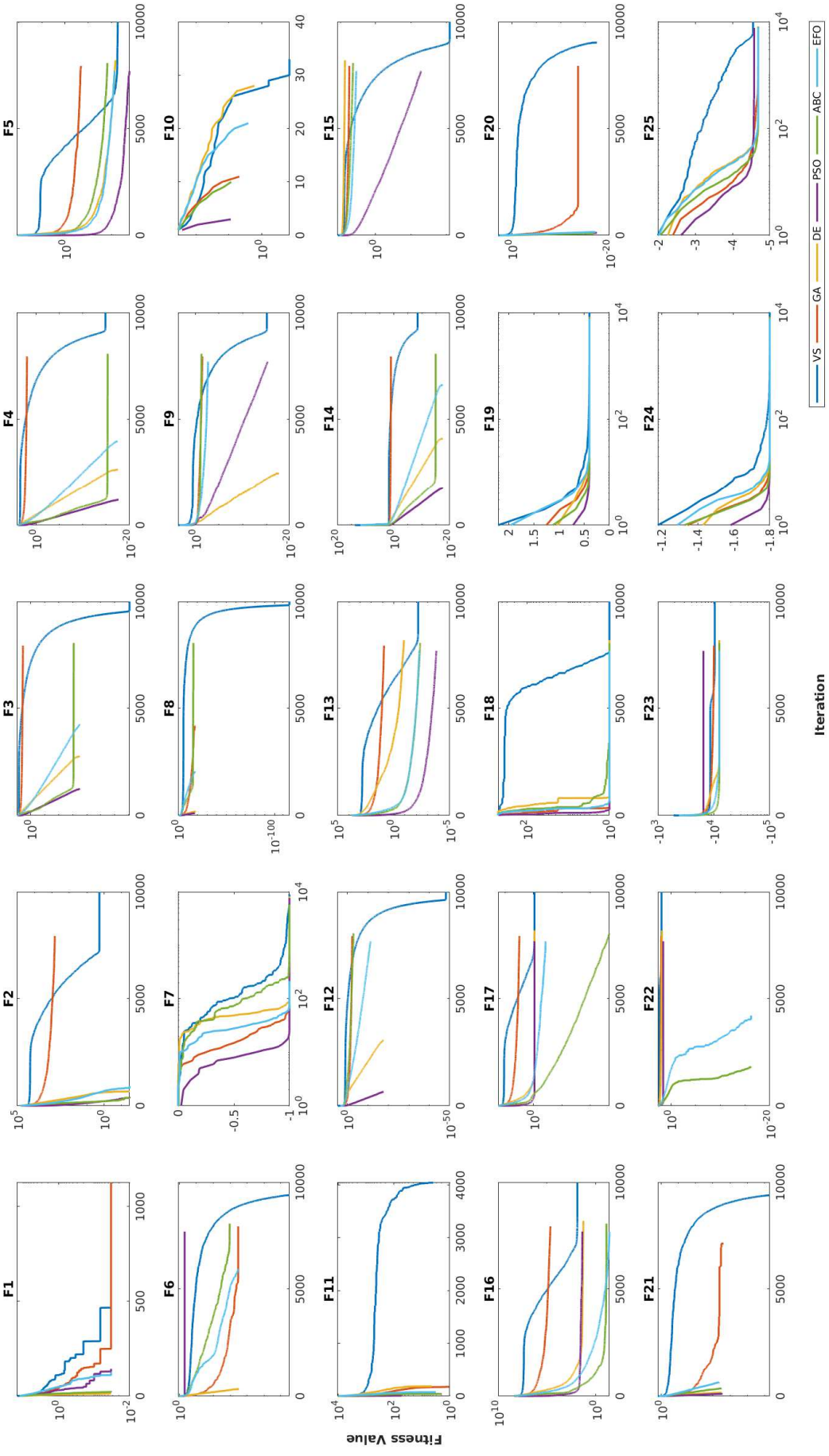

Fig. B.1: Comparative convergence characteristics of EFO for basic mathematical benchmark functions 1 to 25.

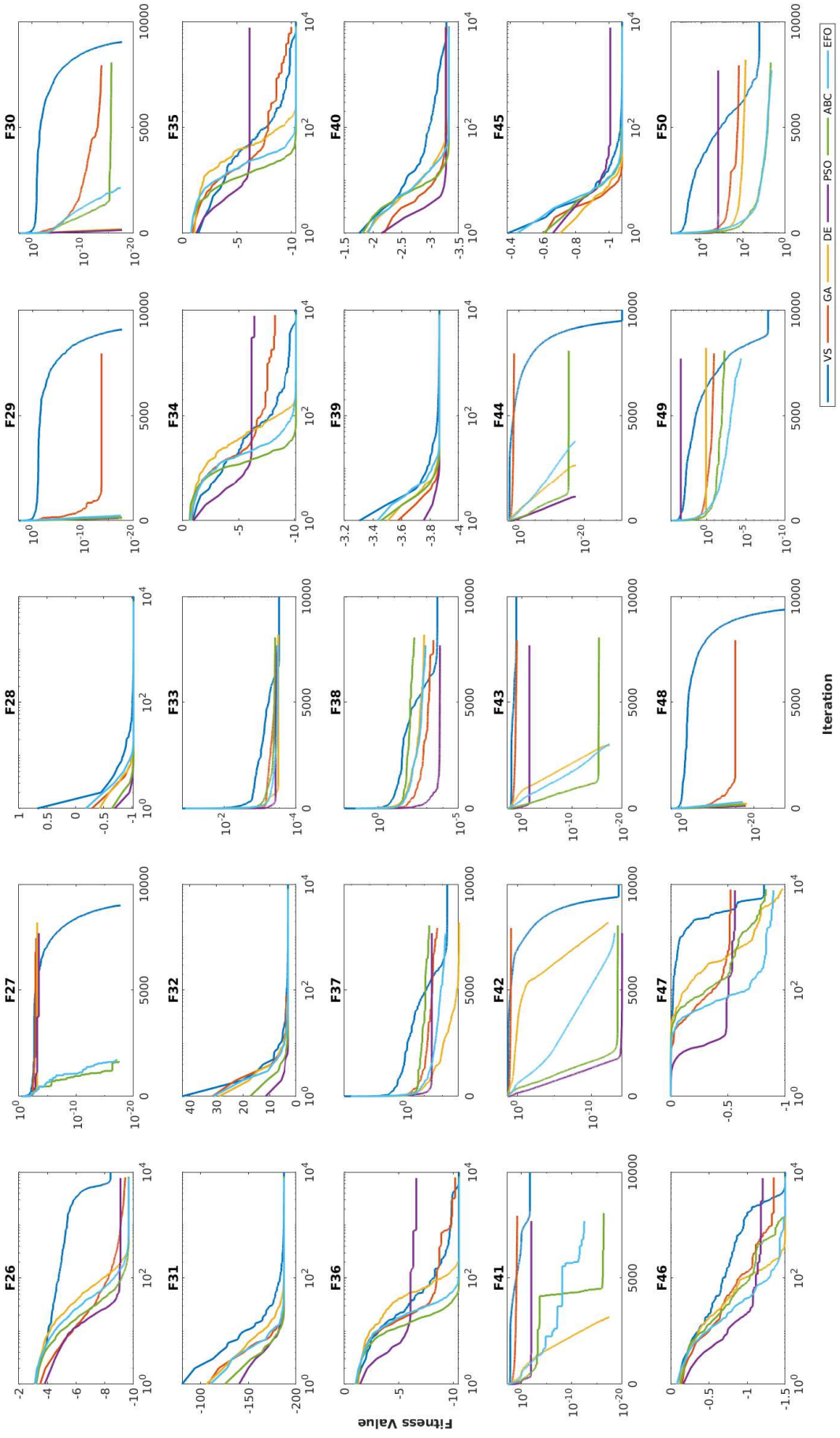

Fig. B.2: Comparative convergence characteristics of EFO for basic mathematical benchmark functions 26 to 50.

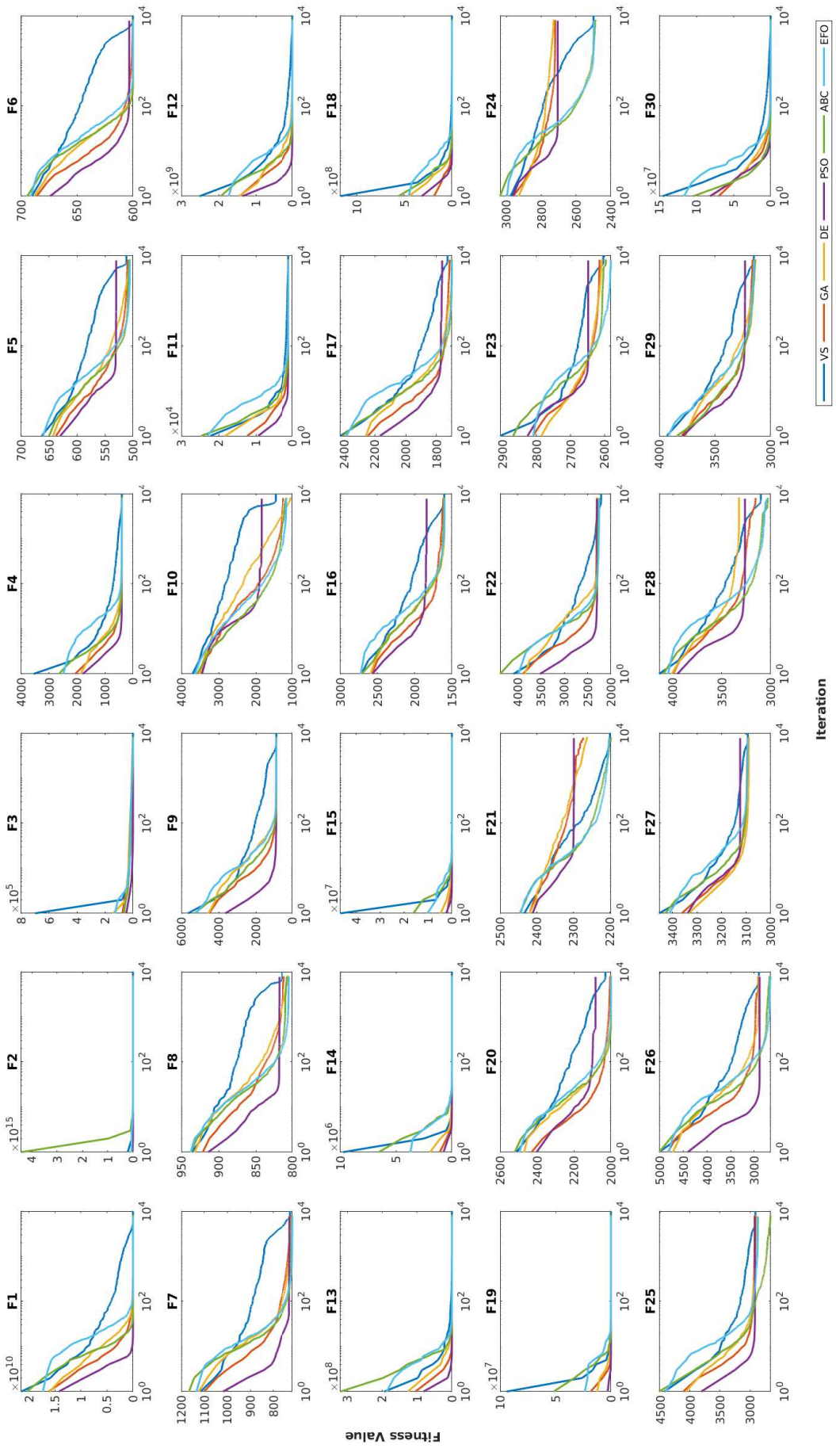

Fig. B.3: Comparative convergence characteristics of EFO for complex mathematical benchmark functions.

References

1. Akay B, Karaboga D (2012) Artificial bee colony algorithm for large-scale problems and engineering design optimization. Journal of Intelligent Manufacturing 23(4):1001–1014, DOI 10.1007/s10845-010-0393-4, URL https://doi.org/10.1007/s10845-010-0393-4
2. Aragon V, C ES, Coello CCA (2010) A modified version of a t cell algorithm for constrained optimization problems. International Journal for Numerical Methods in Engineering 84(3):351–378, DOI 10.1002/nme.2904
3. Arora JS (1967) Introduction to optimum design, 1989. McGraw-Mill Book Company
4. Askarzadeh A (2016) A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm. Computers & Structures 169:1 – 12, DOI https://doi.org/10.1016/j.compstruc.2016.03.001
5. Awad N, Ali M, Liang J, Qu B, Suganthan P (2016) Problem definitions and evaluation criteria for the cec 2017 special session and competition on single objective real-parameter numerical optimization. Tech Rep
6. Bandyopadhyay S, Maulik U (2002) Genetic clustering for automatic evolution of clusters and application to image classification. Pattern Recognition 35(6):1197 – 1208
7. Barr RS, Golden BL, Kelly JP, Resende MGC, Stewart WR (1995) Designing and reporting on computational experiments with heuristic methods. Journal of Heuristics 1(1):9–32, DOI 10.1007/BF02430363, URL https://doi.org/10.1007/BF02430363
8. Bernardino HS, Barbosa HJC, Lemonge ACC (2007) A hybrid genetic algorithm for constrained optimization problems in mechanical engineering. In: 2007 IEEE Congress on Evolutionary Computation, pp 646–653, DOI 10.1109/CEC.2007.4424532
9. Bernardino HS, Barbosa HJC, Lemonge ACC, Fonseca LG (2008) A new hybrid ais-ga for constrained optimization problems in mechanical engineering. In: 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), pp 1455–1462, DOI 10.1109/CEC.2008.4630985
10. Blake C, Merz C (1998) University of california at irvine repository of machine learning databases. Department of Information and Computer Science
11. Bonabeau E, Dorigo M, Theraulaz G (1999) Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press, Inc., New York, NY, USA
12. Boussaïd I, Lepagnot J, Siarry P (2013) A survey on optimization metaheuristics. Information Sciences 237:82 – 117, DOI https://doi.org/10.1016/j.ins.2013.02.041, prediction, Control and Diagnosis using Advanced Neural Computations
13. Civicioglu P (2013) Backtracking search optimization algorithm for numerical optimization problems. Applied Mathematics and Computation 219(15):8121 – 8144, DOI https://doi.org/10.1016/j.amc.2013.02.017
14. Civicioglu P (2013) Backtracking search optimization algorithm for numerical optimization problems. Applied Mathematics and Computation 219(15):8121 – 8144, DOI https://doi.org/10.1016/j.amc.2013.02.017
15. Corne D, Dorigo M, Glover F, Dasgupta D, Moscato P, Poli R, Price KV (eds) (1999) New Ideas in Optimization. McGraw-Hill Ltd., UK, Maidenhead, UK, England
16. Datta D, Figueira JR (2011) A real-integer-discrete-coded particle swarm optimization for design problems. Applied Soft Computing 11(4):3625 – 3633, DOI https://doi.org/10.1016/j.asoc.2011.01.034
17. Derrac J, García S, Molina D, Herrera F (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Computation 1(1):3 – 18, DOI https://doi.org/10.1016/j.swevo.2011.02.002
18. Di Caro G, Ducatelle F, Gambardella LM (2005) Anthocnet: an adaptive nature-inspired algorithm for routing in mobile ad hoc networks. European Transactions on Telecommunications 16(5):443–455, DOI 10.1002/ett.1062
19. Du TS, Ke XT, Liao JG, Shen YJ (2018) Dslc-foa : Improved fruit fly optimization algorithm for application to structural engineering design optimization problems. Applied Mathematical Modelling 55:314 – 339, DOI https://doi.org/10.1016/j.apm.2017.08.013
20. Eiben AE, Smith JE (2003) Introduction to Evolutionary Computing. Springer Berlin Heidelberg, DOI 10.1007/978-3-662-05094-1
21. Falco ID, Cioppa AD, Tarantino E (2007) Facing classification problems with particle swarm optimization. Applied Soft Computing 7(3):652 – 658
22. Falco ID, Cioppa AD, Maisto D, Tarantino E (2008) Differential evolution as a viable tool for satellite image registration. Applied Soft Computing 8(4):1453 – 1462, DOI https://doi.org/10.1016/j.asoc.2007.10.013, soft Computing for Dynamic Data Mining
23. Fan S, Ding S, Xue Y (2018) Self-adaptive kernel k-means algorithm based on the shuffled frog leaping algorithm. Soft Computing 22(3):861–872, DOI 10.1007/s00500-016-2389-2
24. Gandomi AH (2014) Interior search algorithm (isa): A novel approach for global optimization. ISA Transactions 53(4):1168 – 1183, DOI https://doi.org/10.1016/j.isatra.2014.03.018, disturbance Estimation and Mitigation
25. Gandomi AH, Yang XS (2011) Benchmark Problems in Structural Optimization, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 259–281
26. Golinski J (1970) Optimal synthesis problems solved by means of nonlinear programming and random methods. Journal of Mechanisms 5(3):287 – 309, DOI https://doi.org/10.1016/0022-2569(70)90064-9, URL

http://www.sciencedirect.com/science/article/pii/0022256970900649
27. Haldar V, Chakraborty N (2016) A novel evolutionary technique based on electrolocation principle of elephant nose fish and shark: fish electrolocation optimization. Soft Computing pp 1–22, DOI 10.1007/s00500-016-2033-1
28. Han J, Yang C, Zhou X, Gui W (2018) A two-stage state transition algorithm for constrained engineering optimization problems. International Journal of Control, Automation and Systems 16(2):522–534, DOI 10.1007/s12555-016-0338-6, URL https://doi.org/10.1007/s12555-016-0338-6
29. Hatamlou A (2013) Black hole: A new heuristic optimization approach for data clustering. Information Sciences 222(Supplement C):175 – 184, DOI https://doi.org/10.1016/j.ins.2012.08.023, including Special Section on New Trends in Ambient Intelligence and Bio-inspired Systems
30. Holland JH (1992) Adaptation in Natural and Artificial Systems. MIT Press, Cambridge, MA, USA
31. Hwang CL, Yoon K (2012) Multiple attribute decision making: methods and applications a state-of-the-art survey, vol 186. Springer Science & Business Media
32. Kamilaris A, Prenafeta-Boldú FX (2018) Deep learning in agriculture: A survey. Computers and Electronics in Agriculture 147:70 – 90, DOI https://doi.org/10.1016/j.compag.2018.02.016, URL http://www.sciencedirect.com/science/article/pii/S0168169917308803
33. Kanagaraj G, Ponnambalam S, Jawahar N, Nilakantan JM (2014) An effective hybrid cuckoo search and genetic algorithm for constrained engineering design optimization. Engineering Optimization 46(10):1331–1351, DOI 10.1080/0305215X.2013.836640
34. Karaboga D (2005) An idea based on honey bee swarm for numerical optimization. Tech. rep., Technical report-tr06, Erciyes university, engineering faculty, computer engineering department
35. Karaboga D, Akay B (2009) A comparative study of artificial bee colony algorithm. Applied Mathematics and Computation 214(1):108 – 132, DOI https://doi.org/10.1016/j.amc.2009.03.090
36. Karaboga D, Akay B (2009) A survey: algorithms simulating bee swarm intelligence. Artificial Intelligence Review 31(1):61, DOI 10.1007/s10462-009-9127-4
37. Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. Journal of Global Optimization 39(3):459–471, DOI 10.1007/s10898-007-9149-x
38. Karaboga D, Ozturk C (2011) A novel clustering approach: Artificial bee colony (abc) algorithm. Applied Soft Computing 11(1):652 – 657
39. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Neural Networks, 1995. Proceedings., IEEE International Conference on, vol 4, pp 1942–1948 vol.4
40. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680, DOI 10.1126/science.220.4598.671, http://science.sciencemag.org/content/220/4598/671.full.pdf
41. Kramer B (1996) Electroreception and communication in fishes, vol 42. Gustav Fischer
42. Lebastard V, Chevallereau C, Amrouche A, Jawad B, Girin A, Boyer F, Gossiaux PB (2010) Underwater robot navigation around a sphere using electrolocation sense and kalman filter. In: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, DOI 10.1109/iros.2010.5648929
43. Litjens G, Kooi T, Bejnordi BE, Setio AAA, Ciompi F, Ghafoorian M, van der Laak JA, van Ginneken B, Sánchez CI (2017) A survey on deep learning in medical image analysis. Medical Image Analysis 42:60 – 88, DOI https://doi.org/10.1016/j.media.2017.07.005, URL http://www.sciencedirect.com/science/article/pii/S1361841517301135
44. Liu H, Cai Z, Wang Y (2010) Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. Applied Soft Computing 10(2):629 – 640, DOI https://doi.org/10.1016/j.asoc.2009.08.031, URL http://www.sciencedirect.com/science/article/pii/S1568494609001550
45. MacIver M, Fontaine E, Burdick J (2004) Designing future underwater vehicles: Principles and mechanisms of the weakly electric fish. IEEE Journal of Oceanic Engineering 29(3):651–659, DOI 10.1109/joe.2004.833210
46. Maciver MA, Nelson ME (2001) Towards a biorobotic electrosensory systemtowards a biorobotic electrosensory system. Autonomous Robots 11(3):263–266, DOI 10.1023/a:1012443124333
47. Mahdavifar S, Ghorbani AA (2019) Application of deep learning to cybersecurity: A survey. Neurocomputing 347:149 – 176, DOI https://doi.org/10.1016/j.neucom.2019.02.056, URL http://www.sciencedirect.com/science/article/pii/S0925231219302954
48. Mezura-Montes E, Hernandez-Ocana B (2008) Bacterial foraging for engineering design problems: preliminary results. In: Memorias del 4o Congreso Nacional de Computacion Evolutiva (COMCEV 2008)
49. Mezura-Montes E, Coello CC, Landa-Becerra R (2003) Engineering optimization using simple evolutionary algorithm. In: Proceedings. 15th IEEE International Conference on Tools with Artificial Intelligence, IEEE, pp 149–156
50. Mezura-Montes E, Coello CC, Velázquez-Reyes J (2006) Increasing successful offspring and diversity in differential evolution for engineering design. In: Proceedings of the seventh international conference on adaptive computing in design and manufacture (ACDM 2006), pp 131–139
51. Mohamed AW (2018) A novel differential evolution algorithm for solving constrained engineering optimization problems. Journal of Intelligent Manufacturing 29(3):659–692, DOI 10.1007/s10845-017-1294-6, URL https://doi.org/10.1007/s10845-017-1294-6

52. Moller P (1995) Electric fishes: history and behavior. Chapman and Hall fish and fisheries series, Chapman & Hall
53. Neveln ID, Bai Y, Snyder JB, Solberg JR, Curet OM, Lynch KM, MacIver MA (2013) Biomimetic and bio-inspired robotics in electric fish research. Journal of Experimental Biology 216(13):2501–2514, DOI 10.1242/jeb.082743
54. Nezamabadi-pour H, Saryazdi S, Rashedi E (2006) Edge detection using ant algorithms. Soft Computing 10(7):623–628, DOI 10.1007/s00500-005-0511-y
55. Opricovic S (1998) Multicriteria optimization of civil engineering systems. Faculty of Civil Engineering, Belgrade 2(1):5–21
56. Pan WT (2012) A new fruit fly optimization algorithm: Taking the financial distress model as an example. Knowledge-Based Systems 26:69 – 74, DOI https://doi.org/10.1016/j.knosys.2011.07.001
57. Passino KM (2002) Biomimicry of bacterial foraging for distributed optimization and control. IEEE Control Systems Magazine 22(3):52–67, DOI 10.1109/MCS.2002.1004010
58. Rao R, Savsani V, Vakharia D (2011) Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems. Computer-Aided Design 43(3):303 – 315, DOI https://doi.org/10.1016/j.cad.2010.12.015, URL http://www.sciencedirect.com/science/article/pii/S0010448510002484
59. Rao SS (1996) Engineering Optimization: Theory and Practice, 3rd Edition. Wiley-Interscience
60. Rashedi E, Nezamabadi-pour H, Saryazdi S (2009) GSA: A gravitational search algorithm. Information Sciences 179(13):2232–2248, DOI 10.1016/j.ins.2009.03.004
61. Ridge E, Kudenko D (2007) Screening the parameters affecting heuristic performance. In: In Proceedings of the Genetic and Evolutionary Computation Conference, ACM
62. Sandgren E (1990) Nonlinear integer and discrete programming in mechanical design optimization. Journal of Mechanical Design 112(2):223, DOI 10.1115/1.2912596
63. dos Santos Coelho L (2010) Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems. Expert Systems with Applications 37(2):1676 – 1683, DOI https://doi.org/10.1016/j.eswa.2009.06.044
64. Schwefel H (1965) Kybernetische evolution als strategie der experimentellen forschung in der stromungstechnik. Master's thesis, Technical University of Berlin, Germany
65. Sen S (2010) Evolutionary computation techniques for intrusion detection in mobile ad hoc networks. PhD thesis, University of York
66. Solberg JR, Lynch KM, MacIver MA (2007) Robotic electrolocation: Active underwater target localization with electric fields. In: Proceedings 2007 IEEE International Conference on Robotics and Automation, pp 4879–4886, DOI 10.1109/ROBOT.2007.364231
67. Sousa T, Silva A, Neves A (2004) Particle swarm based data mining algorithms for classification tasks. Parallel Computing 30(5–6):767 – 783, DOI https://doi.org/10.1016/j.parco.2003.12.015, parallel and nature-inspired computational paradigms and applications
68. Sun F, Hu G (1998) Speech recognition based on genetic algorithm for training HMM. Electronics Letters 34(16):1563, DOI 10.1049/el:19980096
69. Talbi EG (2009) Metaheuristics: from design to implementation, vol 74. John Wiley & Sons
70. Tan K, Chiam S, Mamun A, Goh C (2009) Balancing exploration and exploitation with adaptive variation for evolutionary multi-objective optimization. European Journal of Operational Research 197(2):701–713, DOI 10.1016/j.ejor.2008.07.025
71. Wang H, Hu Z, Sun Y, Su Q, Xia X (2018) A novel modified bsa inspired by species evolution rule and simulated annealing principle for constrained engineering optimization problems. Neural Computing and Applications DOI 10.1007/s00521-017-3329-5, URL https://doi.org/10.1007/s00521-017-3329-5
72. Weise T (2008) Global Optimization Algorithms – Theory and Application, 2008th edn. Thomas Weise
73. Weise T (2008) Global Optimization Algorithms – Theory and Application, 2008th edn. Thomas Weise, online available at http://www.it-weise.de/
74. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE transactions on evolutionary computation 1(1):67–82
75. Yang X, Suash Deb (2009) Cuckoo search via lévy flights. In: 2009 World Congress on Nature Biologically Inspired Computing (NaBIC), pp 210–214, DOI 10.1109/NABIC.2009.5393690
76. Yang X, Gandomi A, Talatahari S, Alavi A (2012) Metaheuristics in Water, Geotechnical and Transport Engineering. Elsevier insights, Elsevier Science
77. Yang XS (2010) Firefly algorithm, stochastic test functions and design optimisation. International Journal of Bio-Inspired Computation 2(2):78–84
78. Yang XS (2010) Nature-Inspired Metaheuristic Algorithms: Second Edition. Luniver Press
79. Yang XS (2013) 1 - optimization and metaheuristic algorithms in engineering. In: Yang XS, Gandomi AH, Talatahari S, Alavi AH (eds) Metaheuristics in Water, Geotechnical and Transport Engineering, Elsevier, Oxford, pp 1 – 23
80. Yang XS, Cui Z, Xiao R, Gandomi AH, Karamanoglu M (2013) Swarm Intelligence and Bio-Inspired Computation: Theory and Applications, 1st edn. Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands

81. Zahadat P, Schmickl T (2014) Wolfpack-inspired evolutionary algorithm and a reaction-diffusion-based controller are used for pattern formation. In: Proceedings of the 2014 conference on Genetic and evolutionary computation - GECCO 14, ACM Press, DOI 10.1145/2576768.2598262
82. Zhang J, Liang C, Huang Y, Wu J, Yang S (2009) An effective multiagent evolutionary algorithm integrating a novel roulette inversion operator for engineering optimization. Applied Mathematics and Computation 211(2):392 – 416, DOI https://doi.org/10.1016/j.amc.2009.01.048
83. Zhang M, Luo W, Wang X (2008) Differential evolution with dynamic stochastic selection for constrained optimization. Information Sciences 178(15):3043 – 3074, DOI https://doi.org/10.1016/j.ins.2008.02.014, nature Inspired Problem-Solving
84. Zhang N, Ding S (2017) Unsupervised and semi-supervised extreme learning machine with wavelet kernel for high dimensional data. Memetic Computing 9(2):129–139, DOI 10.1007/s12293-016-0198-x
85. Zhang N, Ding S, Zhang J, Xue Y (2017) Research on point-wise gated deep networks. Applied Soft Computing 52:1210 – 1221, DOI https://doi.org/10.1016/j.asoc.2016.08.056
86. Zhang N, Ding S, Zhang J, Xue Y (2018) An overview on restricted boltzmann machines. Neurocomputing 275:1186 – 1199, DOI https://doi.org/10.1016/j.neucom.2017.09.065
87. Zhang Q, Yang LT, Chen Z, Li P (2018) A survey on deep learning for big data. Information Fusion 42:146 – 157, DOI https://doi.org/10.1016/j.inffus.2017.10.006, URL http://www.sciencedirect.com/science/article/pii/S1566253517305328