Arabian Journal for Science and Engineering (2023) 48:10153–10172
https://doi.org/10.1007/s13369-022-07545-3

RESEARCH ARTICLE-COMPUTER ENGINEERING AND COMPUTER SCIENCE

Electrical Search Algorithm: A New Metaheuristic Algorithm for Clustering Problem

Hüseyin Demirci[1] · Nilüfer Yurtay[1] · Yüksel Yurtay[1] · Esin Ayşe Zaimoğlu[1]

Received: 7 July 2022 / Accepted: 12 December 2022 / Published online: 21 December 2022

Abstract
In this study, we proposed a new metaheuristic algorithm called Electrical Search Algorithm (ESA). The proposed algorithm is based on the movement of electricity in high-resistive areas such as wood, glass, and gases. ESA has a unique initialization scheme that only one agent initializes at the lower and upper bounds of the search space, which creates structures called poles. After that, ESA uses unique exploration and exploitation strategies to search. The search mechanism is based on electrons moving to opposite poles. ESA differs from other metaheuristics compared to its initialization scheme, pole search mechanism, and update strategy of the best solutions. ESA was tested with the "100-Digit Challenge" benchmark functions in the IEEE-CEC-2019, four well-known benchmark functions, and an np-hard clustering problem. For the clustering problem, we used four well-known datasets: Iris, Wine, Seeds, and Hepatitis C Virus. ESA was compared with seven different metaheuristic algorithms on these well-known benchmark functions, and the results of the clustering problem were compared with the K-Means algorithm. Additionally, Friedman Signed Rank and post hoc Wilcoxon Test were run to show the significance of the results. In all of the well-known benchmark functions, ESA either offered the best results or similar results to other compared algorithms. The score of the ESA on the IEEE-CEC-2019 benchmark functions shows us that even with the minor evaluation numbers, ESA can achieve similar results to the competing algorithms. Results show that ESA has a robust mechanism for not trapping in local points and moves slow but persistent rate.

Keywords Data clustering · Genetic algorithms · Metaheuristic algorithms · Optimization · Particle swarm optimization

1 Introduction

According to Britannica, optimization is a collection of mathematical principles and methods used in many disciplines to solve quantitative problems [1]. The optimization process is applied in almost every field today. While researchers solved simple optimization problems in traditional ways in the past, nowadays, increasingly complex optimization problems are solved with the help of computers.

Real-world problems in the optimization field are complex and challenging to solve. Algorithms used to obtain exact results in such complex and challenging to solve problems are usually slow in execution time. They are designed to function only in the problem they are intended to solve. It is impossible to use such algorithms for other challenging and complex problems. For this reason, algorithms that do not provide exact solutions and work faster have been developed. These algorithms are called heuristic algorithms. Heuristic algorithms are faster than other algorithms and aim to find a solution close to the best solution by evaluating all possibilities in the solution space. However, they never guarantee that they will find the best solution. Although such algorithms work fast, they are the algorithms dependent on the problem they apply since the information of the problem that the algorithms will use is used in the development phase. These algorithms are called classical heuristic algorithms.

These authors contributed equally to this work.

✉ Hüseyin Demirci
huseyind@sakarya.edu.tr

Nilüfer Yurtay
nyurtay@sakarya.edu.tr

Yüksel Yurtay
yyurtay@sakarya.edu.tr

Esin Ayşe Zaimoğlu
esinzaimoglu@sakarya.edu.tr

[1] Computer and Information Sciences, Sakarya University, 54050 Sakarya, Turkey

Apart from classical heuristic algorithms, meta-heuristic algorithms are not dependent on the problem. Meta means high level. In other words, we can evaluate meta-heuristic algorithms as high-level heuristic algorithms. These algorithms are often designed with inspiration from nature and can be applied to many problems by simply changing the objective function [2]. Although meta-heuristic algorithms do not have any information about the problem, they can be considered a kind of black box because they can give the most appropriate variable values for the most appropriate solution. Recently, some engineering problems that we know np-hard problems were solved with metaheuristic algorithms. No Free Lunch Theorem of Optimization [3] proves that optimization algorithms are not always a suitable and perfect tool for all optimization problems. That means metaheuristic algorithms perform better results on specific optimization problems and not as well on other problems. Hence, this theorem encourages researchers to develop more efficient metaheuristic algorithms. Even in the last two years, researchers solved problems such as damage identification in plate structures [4], identification of static and dynamic cracks on mechanical structures [5], and structural damage detection [6] with metaheuristic algorithms. Also, enhanced versions of these algorithms are still being researched, such as a novel version of Cuckoo search [7] in recent years.

The primary purpose of this study is to develop a new meta-heuristic algorithm. The developed algorithm is inspired by the movement of electricity in a highly resistive environment. The proposed algorithm is called Electrical Search Algorithm (ESA). In this work, we compared our proposed algorithm with some of the early developed and recently developed metaheuristic algorithms which are Genetic Algorithm (GA) [8], Particle Swarm Optimization (PSO) [9], Backtracking Search Algorithm (BSA) [10], Differential Search Algorithm (DSA) [11], Marine Predators Algorithm (MPA) [12], K-Means Clustering Optimization Algorithm (KO) [13], and Harris Hawks Optimizer Algorithm (HHO) [14].

We compared the performance of the proposed algorithm with the performance of seven existing algorithms in four benchmark test functions and the clustering performance of four UCI machine learning clustering datasets of the K-Means algorithm. Also, we tested the proposed algorithm in IEEE CEC (2019) accuracy benchmark functions.

The rest of the paper is organized as follows. Background work and literature review are presented in Sect. 2. The problem definition of the clustering problem and the proposed Electrical Search Algorithm are presented in Sect. 3. Section 4 consists of performance evaluations and an analysis of the proposed algorithm. As a final, conclusions and future works are mentioned in Sect. 5.

2 Background work

2.1 Data clustering

Data clustering defines a collection of patterns in a uniform dataset. The objective is to develop an automatic algorithm that can accurately classify an unlabeled dataset. Data clustering is a technique for organizing data into meaningful clusters based on similarity criteria to find groupings with the smallest intra-cluster and highest inter-cluster distances. Each cluster contains data that are related to each other but different from other clusters. The clustering problem is proven that an NP-hard problem. All clustering algorithms can be classified into three categories: hierarchical, partitional, and overlapping. Hierarchical algorithms work by taking each data as a cluster. Then adds, similar data into the same cluster, and this process reduces the cluster number. The algorithm runs until the predefined cluster number is achieved. Overlapping algorithms are better expressed as fuzzy clustering. In this type of clustering, all data is a member of all clusters with a membership degree. By assigning each data to the cluster with the highest degree of membership, we can assign these data to the same clusters. Partitional algorithms take data and assign them into clusters according to the similarity between data and cluster center. The partitional clustering method splits a data set into several clusters according to the fitness function. The fitness function directly influences the nature of cluster formation. The partitioning job is transformed into an optimization problem once a suitable fitness function has been chosen, for instance, clustering data based on minimization of inter-cluster distance or maximization of the intra-cluster distance of cluster centers [15].

This approach is the most popular since MacQueen developed the k-means algorithm [16]. With this approach, algorithms can cluster large datasets easily. Because of that, researchers from various fields use these types of algorithms. These fields include, but are not limited to, signal and image processing, wireless sensor network coverage, robotics, web mining, pattern recognition, consumer identification in economics, and disease identification in medical sciences [15].

2.2 Data clustering as an optimization problem

Since we can handle clustering as an optimization problem, we can cluster data with meta-heuristic algorithms. Many studies have been done on solving clustering problems with meta-heuristics. Commonly used heuristic algorithms for solving clustering problems are Genetic Algorithms (GAs) and swarm-based optimization algorithms such as the particle swarm optimization algorithm (PSO).

In [17], the genetic algorithm is used to optimize clusters created during unsupervised clustering. The aim of the [17] is to assign data to clusters, and the algorithm's performance

depends on the initial population. Sarkar et al. used evolutionary programming for clustering in [18]. The methodology and the objective functions used for solving the clustering problem are similar to our approach. In work [19], another form of the genetic algorithm is proposed for solving clustering problems. However, it differs from [17] in that they selected initial cluster centers from data that will be clustered. In [20], another genetic algorithm variation is used to solve the clustering problem. In this work, the structure of the gene string consists of a representation of assigned clusters. In [21], the chromosome structure of the genetic algorithm is represented as strings of real numbers values of the cluster centers. The Genetic K-Means Algorithm (GKA) is introduced in [22]. GKA has one step K-Means operator, which takes assigned data points from coded strings and reassigns them to the nearest cluster centers. In [23], a hybrid k-medoid algorithm (HKA) is presented. HKA consists of k-medoid and local search heuristics, and the heuristic search part of the algorithm has hybridized with GA. Fast Genetic K-Means Algorithm (FGKA) is introduced in [24]. FGKA is inspired by [22]. The difference between the FGKA and GKA is that the FGKA allows illegal strings, and GKA tries to eliminate illegal strings. The incremental Genetic K-means Algorithm (IGKA) [25] is an extension of the FGKA [24]. IGKA has better running performance than FGKA in cases when the mutation probability is small. The idea behind the IGKA is that when the mutation probability is small, calculate the objective value and cluster centroids incrementally. Both algorithms converge to the same result, but IGKA is faster than FGKA. In the same paper, the authors also introduced the Hybrid Genetic K-means Algorithm (HGKA), a hybridized version of the IGKA and FGKA that combines the best parts of both algorithms. In [26], an algorithm called COWCLUS is presented. This algorithm is a combination of GA and hill-climbing algorithms. COWCLUS genes are represented as an assigned cluster of data points. COWCLUS acts as a standard GA algorithm, but at the last iteration, the algorithm performs a local search with the hill-climbing algorithm. COWCLUS uses the Variance Ratio Criterion (VRC) as the objective function to determine clusters.

Previously mentioned papers take the clustering problem in fixed cluster numbers. However, in the real world, in most cases, cluster size cannot be determined without prior knowledge. Algorithms that do not need to specify cluster size are developed to overcome this problem.

In [27], automatic clustering is proposed. The idea of the [27] is a weight added into the inter-distance in the objective function, which consists of subtraction of the inter-distance from intra-distance that can determine the cluster size. A small weight value causes clusters to be large numbers and compact, while a higher value of the weight results in clusters with a small number and broader. In paper [28], real coded GA is used for the clustering problem where the number of the clusters is not fixed prior. The paper's fundamental idea is that each gene contains real coded clusters centers in the gene structure and can have a different length. That means that each gene can represent a different number of cluster centers. Another attempt for automatic clustering was made in [29]. This work also uses real-coded GA, and the gene structure contains real-coded cluster centers. Each gene's length is fixed on a maximum number of clusters. The critical element of the paper is that in gene coding, there is a symbol represented as "do not care," which means genes can represent an unfixed number of cluster centers.

There are also swarm-based meta-heuristic algorithms that can solve the clustering problem. One of the popular swarm-based metaheuristic algorithms is PSO.

In [30], PSO is used for solving the clustering problem. The paper shows that the standard PSO algorithm can solve the clustering problem and develop a PSO hybrid that takes the initial seed from the K-Means algorithm. A different approach for clustering with PSO is made in [31]. Despite other approaches, a particle represents only one cluster center and is affected by data points in this approach. Each particle has one part of the solution. All particles must be united and held as the final solution to solve the clustering problem. Accelerated chaotic particle swarm optimization (ACPSO) is proposed to solve the clustering problem [32]. This work uses chaotic maps in standard PSO velocity updating formulation to fast convergence.

Besides pure PSO algorithms, hybrid algorithms are used for solving the clustering problem.

A hybrid version of the PSO and K-Harmonic Means(KHM) algorithm is introduced [33]. The algorithm called PSOKHM works sequentially as PSO and KHM. The first eight iteration works as PSO, and after the eighth iteration, the algorithm works throughout four iterations as KHM. This switching process continues until the end of the iteration limit. A combination of PSO and Rough Set (RS) is presented in [34]. Unlike the other approaches, in this work, the dimension of the data point has a cluster membership. If a data point reaches a certain amount of membership in that cluster, that data point is assigned to that cluster. In [35], a hybrid algorithm called FAPSO-ACO-K is proposed that consists of the combination of fuzzy adaptive particle swarm optimization (FAPSO), ant colony optimization (ACO), and the K-means algorithm. The algorithm differs from the PSO as all particles have their global best value, which can be selected and updated via the ACO trail intensity mechanism. After the algorithm reaches the stopping criteria, the final global best value is considered the initial solution of the K-Means algorithm, and the K-Means algorithm starts clustering. At the end of the calculation, if the found solution is better than the global best value solution algorithm considers the output result of the K-Means solution. If it is imperfect, the algorithm considers the output result of the PSO-ACO part.

In conclusion, in two decades, comprehensive work has been done to solve this np-hard problem. In our work, we selected the objective function, cluster coding scheme in agents, and clustering performance parameters in light of these works.

2.3 Particle swarm optimization

Particle Swarm Optimization (PSO) is a swarm intelligence-based optimization algorithm [9]. The behavior of a flock of birds searching for a food source has inspired this algorithm. The algorithm's core principle is that birds in the swarm share information about newly discovered food resources. The birds move toward the best food resource among the newly discovered resources. While migrating to the best food source, birds examine the search space for new food resources, causing the algorithm to scan most of the search space for alternative food sources, potentially identifying an approximate best solution. In the algorithm, each bird in the flock is called a particle. The position update formula of the particle is given in Eq.(1), and the velocity update formula of the particle is given in Eq. (2). The mathematical expression of the search for food behavior can be defined as follows:

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t) \tag{1}$$

$$\begin{aligned} v_{ij}(t+1) = {} & v_{ij}(t) + c_1 r_{1j}(t)\left[y_{ij}(t) - x_{ij}(t)\right] \\ & + c_2 r_{2j}(t)\left[\Upsilon_j(t) - x_{ij}(t)\right] \end{aligned} \tag{2}$$

where variable x_{ij} and v_{ij} denote the current value and velocity, respectively, of the particle i in dimension j at iteration t. Variable y_{ij} and Υ_j are the best solutions found by particle and the best solutions of all particles, respectively. Constants c_1 and c_2 are the personal and social learning coefficient, respectively, and the variables r_{1j} and r_{2j} are the random numbers generated in the range of [0.0,1.0].

2.4 Genetic algorithm

The genetic algorithm was introduced in 1975 by Holland [8]. The genetic algorithm is an evolutionary algorithm based on the biological reproduction mechanism. In the genetic algorithm, each optimization agent is defined as a gene, and the collection of these genes is expressed as a population. There are two unique steps in this algorithm. The first is the crossover stage, which combines two genes to create a new one. The second stage is a mutation, which randomly modifies different regions of the newly formed gene. With the help of these unique steps, randomly generated populations may converge to a good solution. Over time, the genetic algorithm evolved. Crossover operators [36–38], mutation operators [39], and gene selection mechanisms [40] were all introduced in the literature [41–44].

2.5 Differential search algorithm

The seasonal migration of various animals searching for a productive livelihood is the theory behind the development of the DSA. All organisms join together to form a superorganism and begin searching for efficient habitats. The individuals of the superorganism examine whether they fit the provisional criteria of randomly chosen places during their travels. Persons of the superorganism announced the stopover quickly nestle and continue their voyage from that area if any site is suited for their temporary layover during the trip. During a stopover, the superorganism uses a random process related to the Brownian-like random walk to explore the sites left between the organisms. Then, donors are created by reorganizing all of the superorganisms' individuals. These donors are ideal for locating the stopover location. Randomly chosen persons of the superorganism migrate towards the donor's destination to effectively find the stopover area. This change in the site allows the superorganism to keep moving toward the global minimum. The DSA is convenient for multimodal optimization problems because it does not prefer to select the best possible solution for a presented problem correctly. The DSA contains two fine-tuning control variables based on the task at hand. Detailed information on the DSA can be found in [11].

2.6 Backtracking search algorithm

The BSA [10], which is based on Evolutionary Algorithms (EAs), is designed to address common issues in EAs, such as high sensitivity to control parameters and premature convergence. The BSA follows the same five steps as the traditional EA: initialization, selection-I, mutation, crossover, and selection II. During selection-I, The BSA computes the historical population as a pointer of the search path. At the beginning of each iteration, the algorithm can redefine the historical people. A person from a prior generation, chosen randomly, behaves like a memory until it is modified. The BSA has a different mutation and crossover strategy than the EA and its improved forms. In the mutation phase, merely one parameter is employed to govern the expanse of the search direction matrix while producing trial populations. Crossover, on the other hand, is a difficult concept to grasp. The final trial population is created using two ways. The first method controls the number of individuals who will mutate in a trial using a mix rate.

In contrast, the second method enables merely one randomly selected person to mutate in each attempt. The population is updated using greedy selection in the second stage of the BSA, in which persons with only high fitness

values in the attempted population are employed. Despite its simplistic formation, the algorithm's usage of the dual population approach may make it time and memory-consuming to compute.

2.7 Marine predator algorithm

The idea behind the Marine Predator Algorithm (MPA) is the movements of ocean predators while hunting for their prey. MPA search agents use both Lévy flight and Brownian motion. This algorithm has three main stages. Phase one is when the prey is moving more swiftly than the predator. Phase two is when both predator and prey are moving at nearly the same rate, and phase three is when the predator moves faster than the prey. In phase one, exploration stands out, and Brownian motion is used. Later, in phase two, both Lévy flight and Brownian motion are used for exploration and exploitation. After that, in phase three, only Lévy flight is used for exploitation. The algorithm uses Brownian motion for giant leaps, and for small steps, it uses the Lévy flight strategy [12].

2.8 K-means clustering optimization algorithm

The main idea of the K-Means Clustering Optimization Algorithm (KO) is to find better potential search spaces by clustering the search space into three clusters. Unlike the other optimization algorithms, KO focuses on improving the search space, not the search agent itself. KO updates the initial population only by the assigned search agent to that specific population. Besides, the population in KO shrinks from the initial population number N to four by eliminating the worst solutions. Search agents search majorly in their cluster areas within the perimeter, which is calculated and shrank through the iterations. Also, agents can search outside their defined search space within the limit of the formulation. In each iteration, cluster centers are recalculated, and cluster centers are getting closer to the global best position at the late stages of the algorithm. Search agents select their strategy in the next iteration according to a formulation and a threshold value. More detailed information on the KO can be found in [13].

2.9 Harris Hawks optimizer algorithm

The Harris Hawks Optimizer algorithm (HHO) is inspired by the predator bird Harris' Hawks and their behavior of foraging with their flock and family members. In nature, other predator birds search and hunt down the prey alone. In contrast, Harris' Hawks have a unique cooperative foraging behavior. HHO has two exploration and two exploitation strategies. Exploration strategies are based on random location perch or other family members' location perch, and they have the same chances of selection. The transition from exploration to the exploitation stage is based on escaping energy of the prey, which decreases through iterations. While the prey's escaping energy is high in the early stages of the algorithm, the search strategy is selected as exploration. At the later stages of the algorithm, the search strategy changes from exploration the exploitation because of the decreased prey's escaping energy. For the exploitation stage, HHO has four possibilities: soft besiege, hard besiege, soft besiege with progressive rapid dives, and hard besiege with progressive rapid dives. More detailed information on the HHO can be found in [14].

3 Material and methods

3.1 Definition of clustering problem

Clustering is the operation of dividing a given set of n points in an N-dimensional Euclidean space by a predefined number of groups (or clusters), such as K, based on the measure of similarity/dissimilarity. For example, let us say a group of n points $x_1, x_2, \ldots, x_n$ is represented by S, and a cluster K by $C_1, C_2, \ldots, C_K$. In this case, the mathematical expression of the clustering operation can be defined as follows:

$$\begin{gathered} C_i \neq \varnothing \quad i = 1, \ldots, K \\ C_{ij} = \varnothing, \quad i = 1, \ldots, K, \quad j = 1, \ldots, K, \\ i \neq j \quad \bigcup_{i=1}^{K} C_i = S \end{gathered}$$

The variables that we will optimize in the problem are the cluster centers. In general, this problem is a minimization problem. The problem aims to minimize the sum of the Euclidean distance difference between the points belonging to each cluster center. For the points x and y given by coordinates in n-dimensional Euclidean space, the distance formula is given in Eq. (3).

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{3}$$

Since each point in the dataset is assigned to a cluster, the objective function can be defined as in Eq. (4).

$$f = \sum_{i=1}^{n} d(x_i - z_j) \tag{4}$$

where, x_i is the member of the cluster k_j with the cluster center of z_j, n is the number of points in the cluster k_j and j is defined between 1 to K, which is the number of the predefined

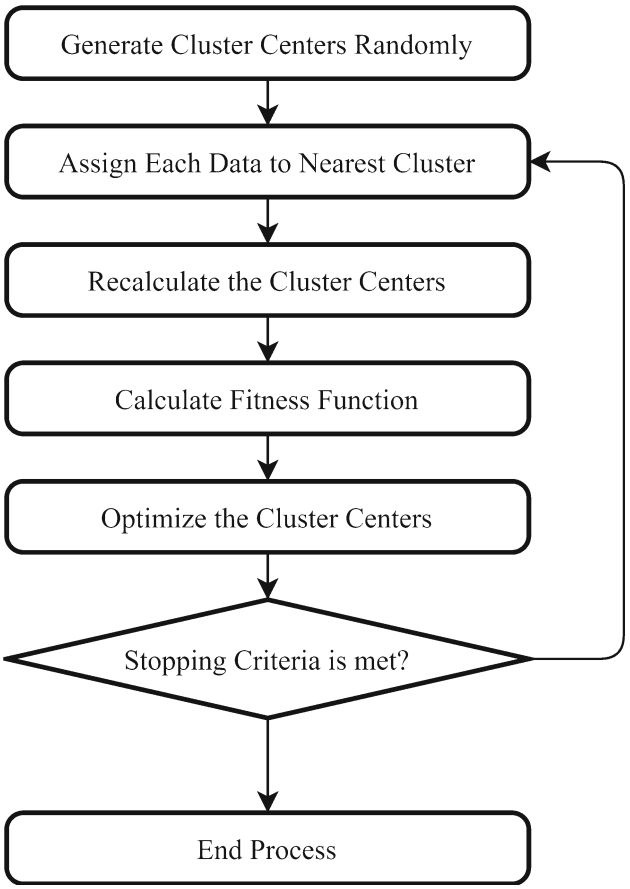


Fig. 1 Flowchart of the clustering problem algorithm

cluster. Thus, our objective in solving the clustering problem is the minimize the function f.

Since we have an objective function, we can solve the problem with metaheuristics algorithms. In our approach, cluster centers are the variables that need to be optimized. Firstly, cluster centers for K clusters are randomly generated. After that, all data is assigned to the nearest cluster with the nearest cluster center. Thus, each data can have one cluster. After assigning each point to the nearest cluster, a new cluster center is found by simply averaging the points assigned to the cluster to ease the algorithm's work. After that, the objective function f is calculated with new cluster centers. The optimization process continues with new cluster centers until predefined stopping criteria are met. A flowchart of the process is given in Fig. 1.

3.2 Proposed algorithm

3.2.1 Movement of the electricity

Electricity is a phenomenon that occurs in nature. In nature, electricity can be found in many forms, such as lightning, the human nerve system, and some animals' defense system. The

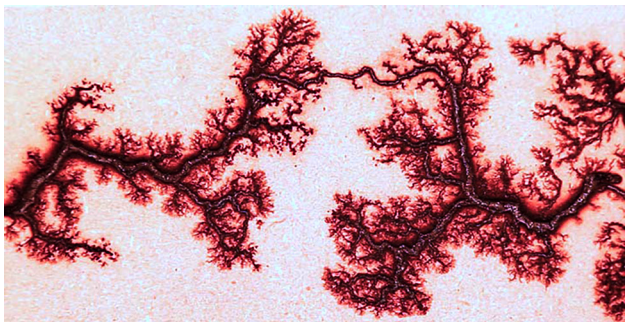


Fig. 2 Lichtenberg figure created on wood with high voltage electricity [46]

movement of the electricity is called electric current and is defined as simply the motion of the atom's valance electron. Suppose an existing potential energy source with a conductor attached to the poles of the energy source has an energy that is large enough to attract or repel an electron. In that case, the electrons in this source can move through the negative pole of the energy source to the positive pole with the help of the valance electrons in the conductor's atom. By its nature, electricity likes to move in low-resistive areas. Since areas of less resistance require less energy than areas of higher resistance, it seeks areas of minimal resistance to moving. While searching the less resistive areas, electricity moves like Brownian motion and leaves behind Lichtenberg figures' pattern. The mathematical structure of the search model of the proposed algorithm is based on the generation of Lichtenberg figures generated by electricity in high resistance fields [45]. In Fig. 2, we can see the pattern of the Lichtenberg figure. Although the shapes have random patterns, they are similar to trees that branch. Because of the tree structure, these shapes are also called Brownian Trees.

The mathematical expression of how the valence electron moves in a high resistive area is given in Eq. (5) [45].

$$p(x,t) = \frac{N}{\sqrt{4\pi Dt}} \times \exp\left(\frac{-x^2}{4Dt}\right) \tag{5}$$

where, N represents the total particle number, D the diffusion coefficient of the environment, t time, and x position.

The electricity moves to the opposite pole to complete the loop via low-resistance areas. The main inspiration of the work is the movement of the electricity while trying to complete the loop and its search for low resistive paths, not the shortest paths.

If we examine the Lichtenberg figures, we can see that electricity diverts to low resistive areas even if it has a chance to complete the loop via the shortest path. It moves around the high-resistive areas and branches to the low-resistive regions. The start point of the poles is the edge of the wood surface and

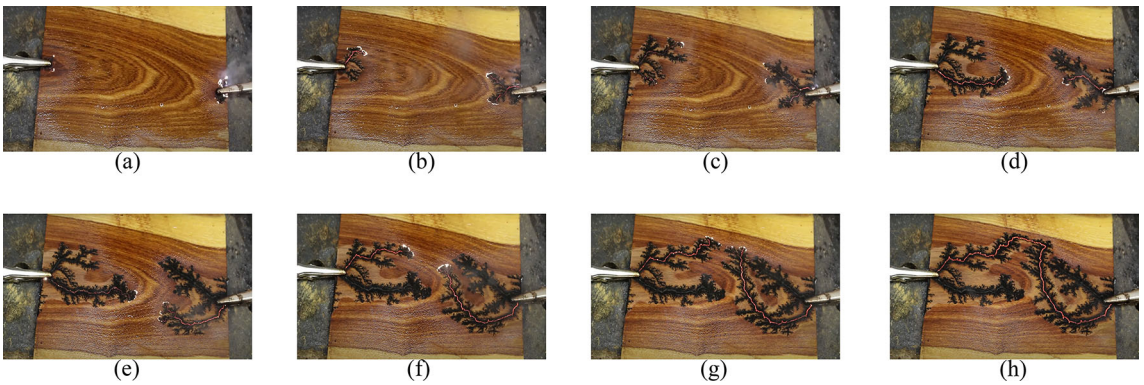


Fig. 3 Burn marks created on wood on high voltage electricity

searches the low resistive areas to the center. If we imagine the wood surface as a search space, the initialization scheme of the proposed algorithm cannot be random. It has to start at two different and far points, which is different from most metaheuristic algorithms. The electricity searches the entire wood surface slowly but stably. This persistence makes our algorithm tackle the early convergence problem. The electricity movement was done in the poles, which is a different mechanic for a search algorithm. In our proposed algorithm first half of the population searches the search space towards the opposite pole.

Meanwhile, the best location of the pole and the current best location of the entire search space is updated. After that other half of the population starts to search with updated values of the best locations. The search starts with only one agent in each pole, and new search agents are created with local search. While creating the agents, the best locations are also updated. This mechanism gives our algorithm a dynamic structure.

Moving steps of the alternating current high voltage electricity on wood given in Fig. 3. If we examine Fig. 3, we can see in 3a that the electricity starts to move with random branches. Because the alternating current, negative and positive poles of the electric source are shifting, it causes the movement of the electric start on both poles, respectively. Regular electric grid alternates at 50 or 60 Hz, depending on the country. It means polar shift happens every 0.02 or 0.0167 seconds. Because the shifting is too fast, the human eye cannot see the shifting motion, and it may seem that electricity moves from both poles simultaneously. However, it moves respectively. If we look at 3b right side of the pole is branched with two main branches, and the left side branched three separate ways but only continues from one. Later 3c, we can see that the left side of the pole continues from one way while the right side selects the main way to move forward. After that, in 3d, we can see that left side of the pole, the main way is abandoned, and a new main way is generated while the right side of the pole adjusts itself to changes in the left pole. When we look at the 3e and 3f, it may seem electricity almost completes the loop, but because of the high resistive areas main way on the left side of the pole is abandoned. A new way starts from the earlier main way, which is generated in 3c, and the right side of the pole adjusts itself and curves upward to reach the left side of the pole. In images 3g and 3h, we can see both poles search the wood for low resistive areas and move towards each other. If we look at the starting point of both poles, the shortest way is the linear line, but electricity chooses to consume less energy to move and complete the loop. This structure of the search mechanism creates our algorithm's search pattern. Our proposed algorithm mimics this natural phenomenon and tries to search the entire search space with branches while abandoning high resistive areas and transferring energy to more valuable areas.

3.2.2 Algorithm steps

The proposed algorithm mimics the event of the movement of electricity on a high resistive area or surface. Like electricity, our algorithm has two poles which are negative and positive. The algorithm starts in these poles. The search agent in the negative pole starts a search by referencing the minimum values of the search space dimensions. The search agent starts a search with maximum values of the search space dimensions for the positive pole. The search begins with an agent at both poles sequentially and increases the number of agents while agents of opposite poles attract each other. The maximum number of agents for each pole is equal and predefined at the beginning of the algorithm. Agents are attracted to both the best of the opposite pole agent and the global best agent, which can be in the opposite pole or the same pole as the agent. As shown in Fig. 2, many branches encountered with very high resistance are terminated prematurely and are not continued moving to the opposite pole. We can say those branches failed for search or are trapped in local extremum points. To mimic this mechanism, we added a failure variable

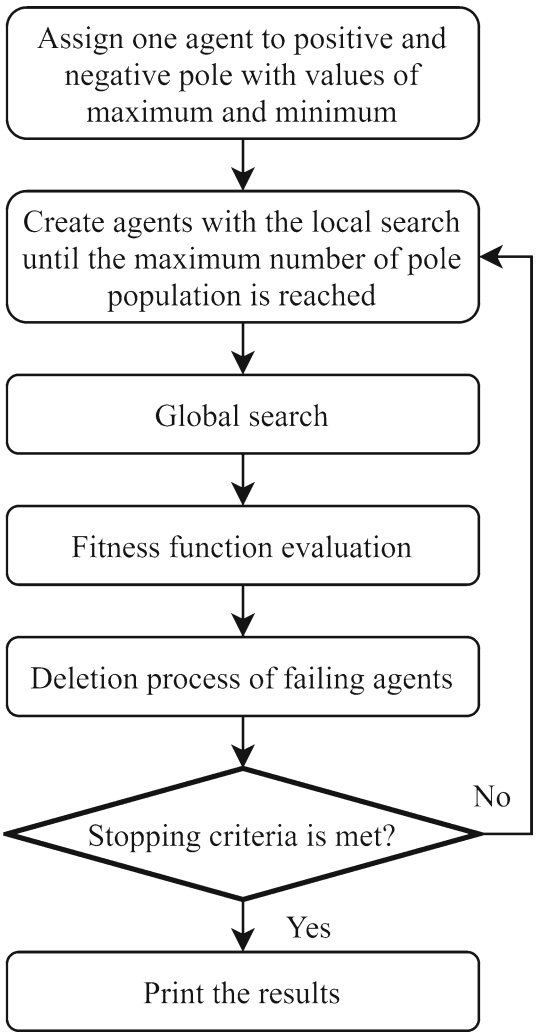


Fig. 4 Flowchart of proposed ESA

for the agents. This variable will be increased whenever an agent fails to provide a better solution for the problem. That means our agents have a memory feature that can remember the best solution of their own.

After an agent passes the limit of the failure variable, the proposed algorithm will delete the agent, and a new agent will be created in deleted agent's assigned pole. This mechanism provides the algorithm solution for the problem of being stuck up in the local extremum point. As a result of that, the population gets better diversity. The flowchart of the proposed algorithm is given in Fig. 4. The proposed algorithm uses Eq. (6) for global search and Eq. (7) for local search. Besides that, the local search formula is an adapted version of Eq. (5).

$$x_i(t+1) = 0.7 \times \left(r_1 \times x_i + (1-r_1) \times x_{\text{best}_i}\right) + 0.3 \times \left(r_2 \times x_i + (1-r_2) \times x_{\text{pole}_i}\right) \quad (6)$$

$$x_i(t+1) = x_i + \frac{0.3 \times \Delta x_{\text{best}} + 0.7 \times \Delta x_{\text{pole}}}{\sqrt{4\pi\, Dt}} exp\left(\frac{-1}{4Dt}\right) \quad (7)$$

In Eq. (6), variable x_i denotes the current value of the agent i. Variables r_1 and r_2 are the random numbers generated in the range of [0.0,1.0]. Finally, variables x_{best_i} and x_{pole_i} denote the values of the best agents in the search space and opposite pole in that iteration, respectively. Eq. (6) mimics the attraction of the opposite poles. The values x_{best_i} and x_{pole_i} change in every iteration, and these values are specific for the running iteration. The change in values mimics the adjustment when the pole shifting occurs and a new main way for the pole is generated. The fixed values 0.7 and 0.3 are attraction coefficients selected by numerous experiments while developing the algorithm. Because we want exploration to occur in small and big steps, we limit the opposite pole attraction to 0.3 and the best location attraction to 0.7. Attraction steps have two possibilities: the best location occurs in the opposite pole, and the best location occurs in the same pole. When the best location occurs in the opposite pole search agent moves with big steps to the opposite pole and if the best location occurs in the same pole search agent moves with small steps to the opposite pole. Either way, the agents always attract to opposite poles and tend to move towards opposite poles.

In Eq. (7), same as Eq. (6), variable x_i denotes the current value of the agent i. Variables D and T represent the diffusion coefficient and iteration number, respectively. Finally, variables Δx_{best} and Δx_{pole} indicate the distance between the current agent's position and the best agents of the search space and opposite pole, respectively. With the help of Eq. (6), the proposed algorithm uses a big step to reach the global best solution, and with Eq. (7), the algorithm uses a small step to approach the opposite pole's best solution. All properties and phases of the ESA are visualized in Fig. 5. In section (a) of Fig. 5, the initialization of the ESA is shown. In sections (b) and (c), global and local search is initiated. Finally, in part (d) search process is completed. Subdivision (e) of Fig. 5 shows us the failing branches of the algorithm. In subdivision (f), we can see the creation of random new twigs and boughs resulting from the deletion of failing branches through the iterations. Eq. (7) mimics the branching structure of electricity. The phrase $0.3 \times \Delta x_{\text{best}} + 0.7 \times \Delta x_{\text{pole}}$ determines the direction and the phrase $\frac{1}{\sqrt{4\pi\, Dt}} \times exp\left(\frac{-1}{4Dt}\right)$ determines the magnitude of the step. The algorithm's exploitation phase consists of the agent's branching movement after the exploration phase. The fixed values 0.7 and 0.3 are selected for the same purpose as in Eq. (6) to limit the branching with big and small steps. Also, this exploitation stage has two possibilities: When the best location occurs in the opposite pole, the search agent branches with big steps to the opposite pole, and if the best location occurs in the same pole, the search agent branches with small steps to the opposite pole.

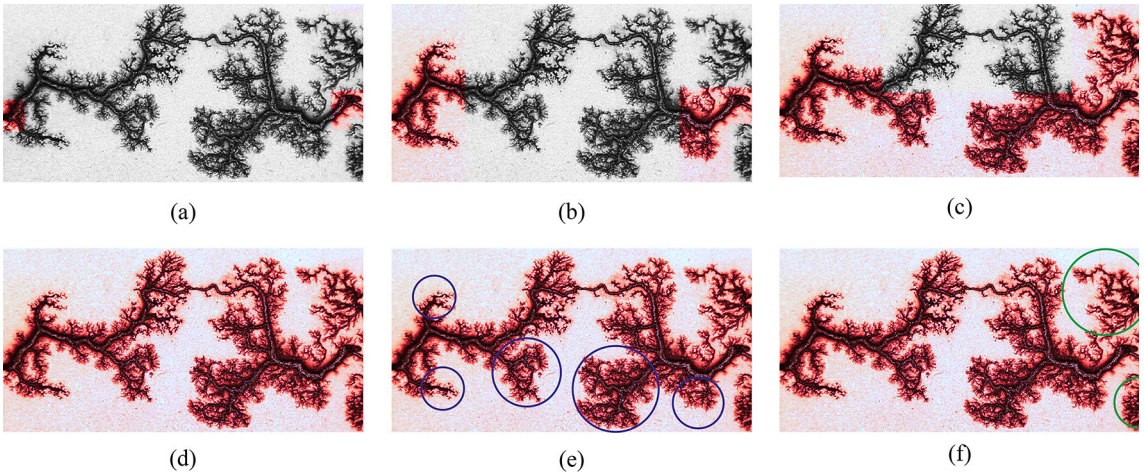


Fig. 5 **a** Initialization of ESA, **b** Global and local search in early stages of ESA, **c** Global and local search in late stages of ESA, **d** Completion of search process of ESA, **e** Failing Branches, **f** Randomly created branches

The fixed values 0.7 and 0.3 can be adjusted for the problem, but we choose not to temper these coefficients. Because tuning these coefficients to best is itself an optimization problem.

The failure limit mimics the abandonment of the main ways and transferring energy to a different area in phenomena. This structure helps the algorithm not to trap in local points. The failure limit is determined with the ‰3 of the maximum iteration number. Also, the failure limit can be tuned. The failure limit determines whether the agents spend time on exploration or exploitation. For more exploration failure limit can be decreased, and for more exploitation failure limit can be increased.

Graphical abstract of the proposed algorithm is given in Fig. 6 and pseudo code of the algorithm is given in Algorithm 1

Algorithm 1 Psuedo Code of The ESA

```
Initialize the parameters:
    pop_size
    max_iteration
    Diffusion_Coefficient
    Kill_limit
Initialize one spark to positive and negative pole with values of maximum and minimum of the search space
while t < max_iteration do
    Create sparks with the local search until the maximum number of pole population is reached
    Global Search
    Calculate Fitness value for each spark
    Update:
        failure of the sparks
        best_spark_of_population
        best_spark_of_each_poles
    Delete sparks which has failure ≥ Kill_limit
    t = t + 1
end while
Return:
    best_spark_fitness
    best_spark_variables
```

3.3 Test environment setup

We tested the proposed algorithm with four frequently used benchmark functions. These functions have multi-modal, unimodal, separable, and non-separable properties. Multi-modal functions have more than two local extremum points and are hard to solve compared to unimodal functions because of the probability of trapping in local extremum points. Solving a non-separable function is more challenging than solving a separable function because function's each variable depends on the other variables. The comparison functions used to test the algorithm are given in Eq.(8), Eq. (9), Eq. (10), and Eq. (11) and named Rosenbrock, Ackley, Griewank, and Rastrigin functions, respectively. All benchmark problems are minimization problems; therefore, minimum results are better.

$$f(x) = \sum_{i=1}^{d-1} \left[100 \left(x_{i+1} - x_i^2 \right)^2 + (x_i - 1)^2 \right] \tag{8}$$

$$f(x) = -20 \times \exp\left(-0.2 \times \sqrt{\frac{1}{d} \sum_{i=1}^{d} x_i^2} \right) - \exp\left(\frac{1}{d} \sum_{i=1}^{d} \cos\left(2\pi x_i\right) \right) + 20 + \exp(1) \tag{9}$$

$$f(x) = \sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}} \right) + 1 \tag{10}$$

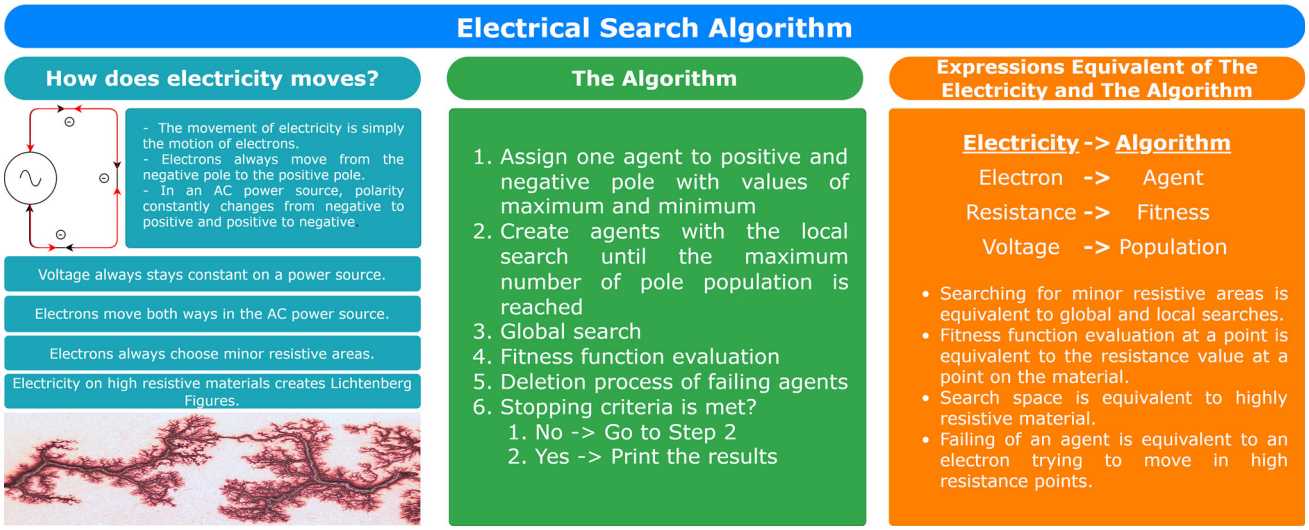


Fig. 6 Graphical abstract of the proposed ESA algorithm

Table 1 Control parameters of algorithms

Algorithm	Control parameters
ESA	D=1.49 (Diffusion coefficient of wood), Failure Limit=15
GA	Crossover rate=0.75, Mutation rate=0.1
PSO	C_1=1.49, C_2=1.49
BSA	All Parameters Set to Default
DS	Method=Surjective DSA (S-DSA)

$$f(x) = 10d + \sum_{i=1}^{d} \left[x_i^2 - 10\cos(2\pi x_i) \right] \tag{11}$$

Rastrigin, Ackley, and Griewank equations have a multimodal property, and the Rosenbrock equation has unimodal property. Also, Ackley, Griewank, and Rosenbrock equations have a non-separable property, and the Rastrigin equation has separable property. We compared the proposed algorithm with GA, PSO, BSA, and DS algorithms on these four benchmark functions with three different dimension values, which are 10, 20, and 30, respectively. The control parameters of these algorithms are given in Table 1. In all test cases, the population limit was set to 100. Also, for fair evaluation, the iteration limit was set to 500, 1000, and 1500 for dimensions 10, 20, and 30, respectively.

The primary purpose of selecting the CEC 2019's 100-Digit Challenge is that if we expand the time and iteration limit, our algorithm can perform better because its search mechanism structure tends to update the best locations. Besides that, the failure structure of the agents makes the agents born at different locations in the respective pole, which gives the algorithm better exploration within this expanded time and iteration limit. Because of the rounded and shifted versions of the problems, we can claim that our proposed algorithm does not tend to memorize the problems and searches randomly according to our formulations.

The CEC 2019's 100-Digit Challenge problem can be found in [47]. The challenge consists of ten problems; seven of them are shifted and rotated. The other three problems are new and challenging. The aim is to calculate the accuracy of the function up to 10 digits without a time limit. Boundaries and dimension values of the IEEE CEC (2019) functions are given in Table 2. Control parameters of the proposed algorithm are set to population=30 Diffusion Coefficient=1.49 Iteration Number=1E+06.

The proposed algorithm and the popular clustering algorithm K-Means are compared on performance in solving the clustering problem. For comparison, we used four real-life datasets provided by the UCI Machine Learning Repository, which are Iris [48], Wine [49], Seeds [50], and HCV [51] datasets. Both for the proposed algorithm and K-Means algorithm, the maximum iteration is set to 150, and the highest cluster number is set to 10. Besides, the proposed algorithm's agent number is limited to 200, and the failure limit is set to 5. The properties of these data sets are given in Table 3. The Davies–Bouldin index (DB-Index) [52] was used to evaluate the clustering problem performance. DB-Index measures the compactness and separateness of the clusters. The smallest DB-Index indicates that clustering performance is better for the found cluster centers.

3.3.1 Statistical tests

The comparability of the algorithms has frequently been inferred through hypothesis testing [53]. However, to conclude, it is necessary to determine the null hypothesis H0

Table 2 The basic parameters of the 100-digit challenge

No.	Functions	$F_i^* = F_i(x^*)$	Dimension	Search range
1	Storn's Chebyshev plynomial fting poblem	1	9	(−8192,8192)
2	Inverse Hilbert mtrix poblem	1	16	(−16.384,16.384)
3	Lennard-Jones mnimum eergy custer	1	18	(−4,4)
4	Rastrigin's fnction	1	10	(−100,100)
5	Griewangk's fnction	1	10	(−100,100)
6	Weierstrass fnction	1	10	(−100,100)
7	Modified Schwefel's fnction	1	10	(−100,100)
8	Expanded Schaffer's F6 fnction	1	10	(−100,100)
9	Happy Cat fnction	1	10	(−100,100)
10	Ackley function	1	10	(−100,100)

Table 3 Dataset properties

Dataset	Number of instances	Number of attributes
Iris	150	4
Wine	178	13
Seeds	210	7
HCV	615	14

and the alternative hypothesis H1. The null hypothesis is a statement that often denotes no differences between the algorithms being compared. The alternate hypothesis, on the other hand, represents the differences. For our purposes:

H0: There is no difference between compared algorithms

H1: There is a difference between compared algorithms

The statistical test probability value (α) also determines whether the hypothesis should be disproved. The acceptable level for our test is 0.05.

A non-parametric statistical test, the Friedman test, was first presented by Friedman [54,55]. It has been used to specify differences in how specific algorithms behave. The Friedman test results for compared algorithms are displayed in columns, and test cases are represented in rows. The test begins by ranking each row according to the values of the columns in the row, after which it calculates the overall rank values for each column. The X^2 (Chi-square) distribution with k-1 degrees of freedom (df), where k is the number of compared methods, is used to calculate the test's significance.

We can find the appropriate X^2 values for df in [56]. Expected $X^2 = 9.49$ from the table in [56] because (df)=4 and $\alpha = 0.05$. The null hypothesis must be rejected if the actual X^2 value is higher than anticipated.

Another non-parametric statistical test used to identify differences between two samples or algorithms is the Wilcoxon signed-rank test [57]. For creating the difference vector, the test typically starts with discrepancies between the two algorithms' outputs, which have sizes of $N * 1$, where N is the total number of tests. Then, it assigns a rank of 1 to each row in the vector, starting with the minimum value. The difference vector's R_- and R_+ values are then calculated. The T value of the test, $\min(R_-, R_+)$, is calculated based on the data. As a result, using the T value, the test's probability value p is calculated.

4 Results and discussions

4.1 Comparison of the results of benchmark functions

All test cases are run ten times with relevant settings. Rosenbrock, Ackley, Griewank, and Rastrigin functions' evaluation results for 10, 20, and 30 dimensions are given in Table 4. All values in Table 4 are the mean of the executed ten runs with corresponding algorithms.

We can see in Table 4 that our algorithm, ESA, has given almost similar results with MPA, KO, and HHO in each test case. Our algorithm found an exact solution in all ten runs in six of the twelve cases. Both HHO and KO have also found an exact solution in all ten runs in six of the twelve cases. MPA only found five exact solutions.

We can see in Figs. 7a, 8a, 9a, and 10a that ESA has performed a tremendous step-down pattern that is a product of our failure deletion procedure. Those failing agents are deleted whenever ESA is trapped in a local minimum point, and brand-new agents are created at corresponding poles. The Rastrigin function is the easiest of all three benchmark functions besides the Ackley and Griewank functions, which are at a similar difficulty.

ESA found better solutions than GA, PSO, BSA, and DS but only in Ackley with 20 and 30 dimensions; ESA found better results than MPA, KO, and HHO. Other than ESA performs similar results compared to MPA, KO, and HHO.

Table 4 Evaluation results of the benchmark functions

Benchmark function	Dimension	GA	PSO	BSA	DSA	MPA	KO	HHO	ESA	Figures
Rosenbrock	10	8,92E+00	1,40E+01	1,07E+01	6,24E+00	3,99E−01	6,23E+00	**5,14E−04**	1,07E+00	7
		(7,72E−02)	(2,30E+01)	(2,83E+00)	(1,99E+00)	(1,95E−01)	(3,01E−01)	(5,25E−04)	(1,74E+00)	
	20	1,70E+01	1,59E+01	3,11E+01	2,56E+01	8,95E+00	1,62E+01	**1,16E−04**	9,07E−01	
		(5,65E+00)	(1,57E+00)	(1,88E+01)	(2,01E+01)	(4,14E−01)	(2,86−E01)	(1,42E−04)	(1,62E+00)	
	30	2,60E+01	2,54E+01	4,26E+01	6,31E+01	1,84E+01	2,64E+01	**1,68E−04**	1,02E−01	
		(8,65E+00)	(2,49E+00)	(2,59E+01)	(2,63E+01)	(7,96E−01)	(3,57E−01)	(1,66E−04)	(1,90E−01)	
Ackley	10	1,14E−10	7,99E−01	1,72E−02	1,16E−04	4,44E−15	**8,88E−16**	**8,88E−16**	1,34E−04	8
		(3,19E−10)	(9,90E−01)	(9,85E−03)	(6,63E−05)	(0,00E+00)	(0,00E+00)	(0,00E+00)	(2,43E−04)	
	20	**0,00E+00**	2,95E−01	4,49E−03	5,21E−05	3,38E−15	8,88E−16	8,88E−16	**0,00E+00**	
		(0,00E+00)	(5,65E−01)	(5,43E−03)	(3,97E−05)	(1,63E−15)	(0,00E+00)	(0,00E+00)	(0,00E+00)	
	30	**0,00E+00**	1,20E−01	3,07E−04	3,42E−05	4,09E−15	8,88E−16	8,88E−16	**0,00E+00**	
		(0,00E+00)	(3,46E−01)	(1,63E−04)	(2,73E−05)	(1,75E−15)	(0,00E+00)	(0,00E+00)	(0,00E+00)	
Griewank	10	1,22E−05	2,42E−01	6,17E−02	3,27E−03	2,37E−09	**0,00E+00**	**0,00E+00**	2,96E−07	9
		(3,66E−05)	(2,18E−01)	(2,70E−02)	(4,11E−03)	(7,10E−09)	(0,00E+00)	(0,00E+00)	(5,19E−07)	
	20	**0,00E+00**	3,69E−02	9,67E−04	2,59E−06	**0,00E+00**	**0,00E+00**	**0,00E+00**	**0,00E+00**	
		(0,00E+00)	(1,51E−02)	(1,15E−03)	(4,73E−06)	(0,00E+00)	(0,00E+00)	(0,00E+00)	(0,00E+00)	
	30	**0,00E+00**	1,22E−01	7,07E−06	2,90E−08	**0,00E+00**	**0,00E+00**	**0,00E+00**	**0,00E+00**	
		(0,00E+00)	(2,45E−01)	(7,18E−06)	(5,00E−08)	(0,00E+00)	(0,00E+00)	(0,00E+00)	(0,00E+00)	
Rastrigin	10	5,69E−01	6,48E+00	2,61E+00	1,32E−03	**0,00E+00**	**0,00E+00**	**0,00E+00**	**0,00E+00**	10
		(9,94E−01)	(1,21E+01)	(6,71E−01)	(1,85E−03)	(0,00E+00)	(0,00E+00)	(0,00E+00)	(0,00E+00)	
	20	2,69E+00	2,11E+01	7,92E+00	1,45E−01	**0,00E+00**	**0,00E+00**	**0,00E+00**	9,73E−07	
		(5,49E+00)	(1,17E+01)	(1,55E+00)	(3,87E−01)	(0,00E+00)	(0,00E+00)	(0,00E+00)	(2,38E−06)	
	30	**0,00E+00**	3,99E+01	1,52E+01	1,48E−01	**0,00E+00**	**0,00E+00**	**0,00E+00**	**0,00E+00**	
		(0,00E+00)	(1,95E+01)	(3,39E+00)	(3,58E−01)	(0,00E+00)	(0,00E+00)	(0,00E+00)	(0,00E+00)	

Bold values are the smallest values of the means of the ten runs for each corresponding test case. Values in parentheses are standard deviations of the corresponding results

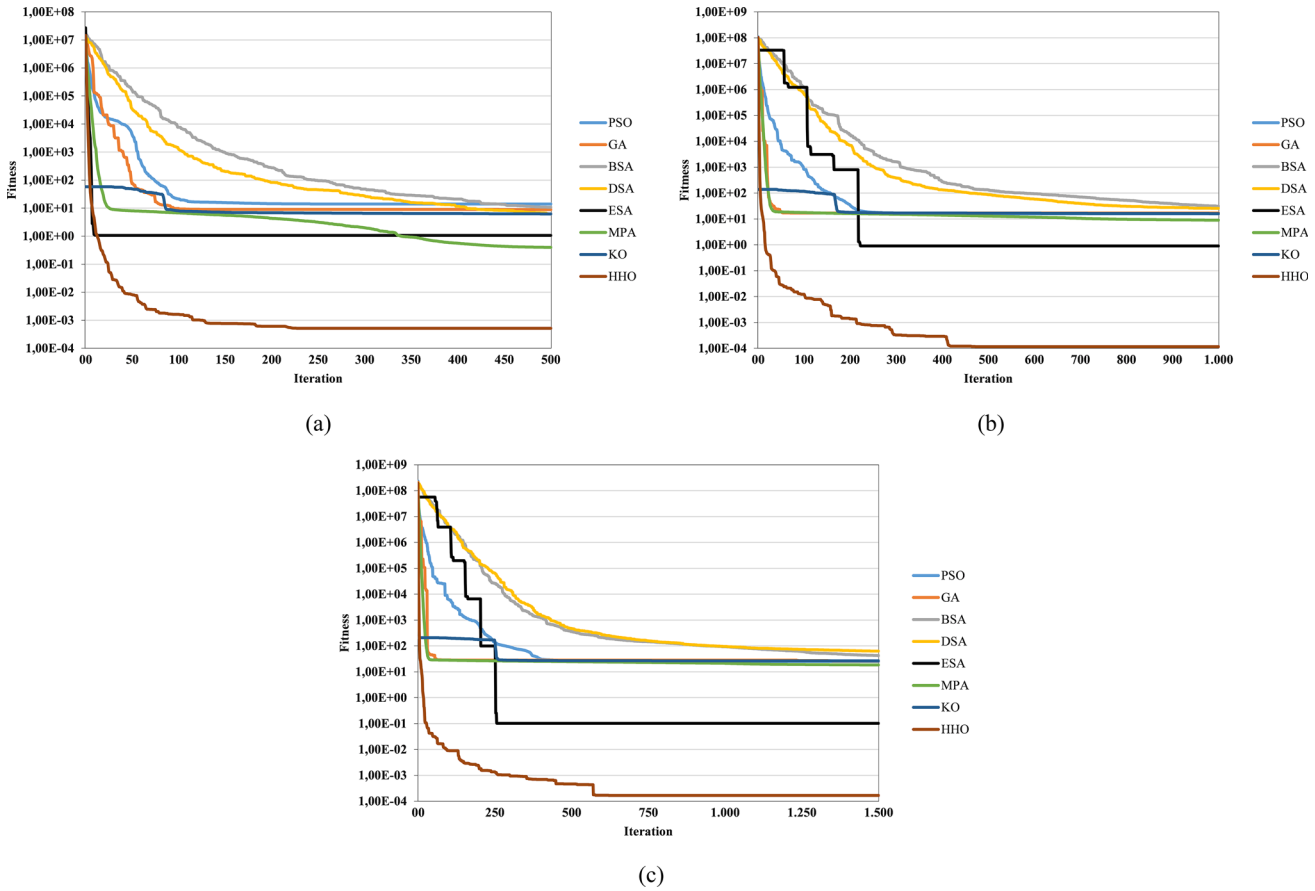


Fig. 7 Rosenbrock function in **a** 10 Dimension, **b** 20 Dimension and **c** 30 Dimension

Our proposed algorithm is designed to handle the problem with a wide range of iterations and time limits. If we double the iterations, our algorithm performs the same as the MPA, KO, and HHO and always finds the exact solutions for all test cases.

4.2 Statistical analysis

4.2.1 Friedman test

We can see in Table 5 which algorithm is more successful. The algorithm that has a minimum rank value has a better performance compared to other algorithms. As shown in Table 5, ESA performs second best in all test functions.

Besides that probability value is found $1,0245E-10$ is way less than $\alpha = .005$, X^2 value is $60,842672$, which is higher than expected value of 14.07. This means null hypothesis $H0$, "There is no difference between compared algorithms" must be rejected. It means that the $H1$ hypothesis is true: "There is difference between compared algorithms". Results of the Friedman test were given in Table 6.

4.2.2 Wilcoxon signed rank test

The major differences between the compared methods are described by p values. For the null hypothesis to be rejected, these numbers need to be lower than 0.05. If not, it must accept it, implying that the comparing methods' performance is equivalent.

We can see in Table 7 that ESA has yielded a different performance than the most of the compared algorithms. It means our proposed algorithm behaves differently from the compared algorithms. However, in the case of MPA and KO, our algorithm performed similar results.

4.3 Comparison of the results of the IEEE CEC (2019) 100 Digit Challenge Problems

100 Digit Challenge is inspired by Æsop's fabled race between Tortoise and Hare. The main idea of the fable is that relying too much on speed as the success criterion is not suitable for always. Because of that primary goal of the challenge is to determine whether the algorithms that search persistently or aggressively are better.

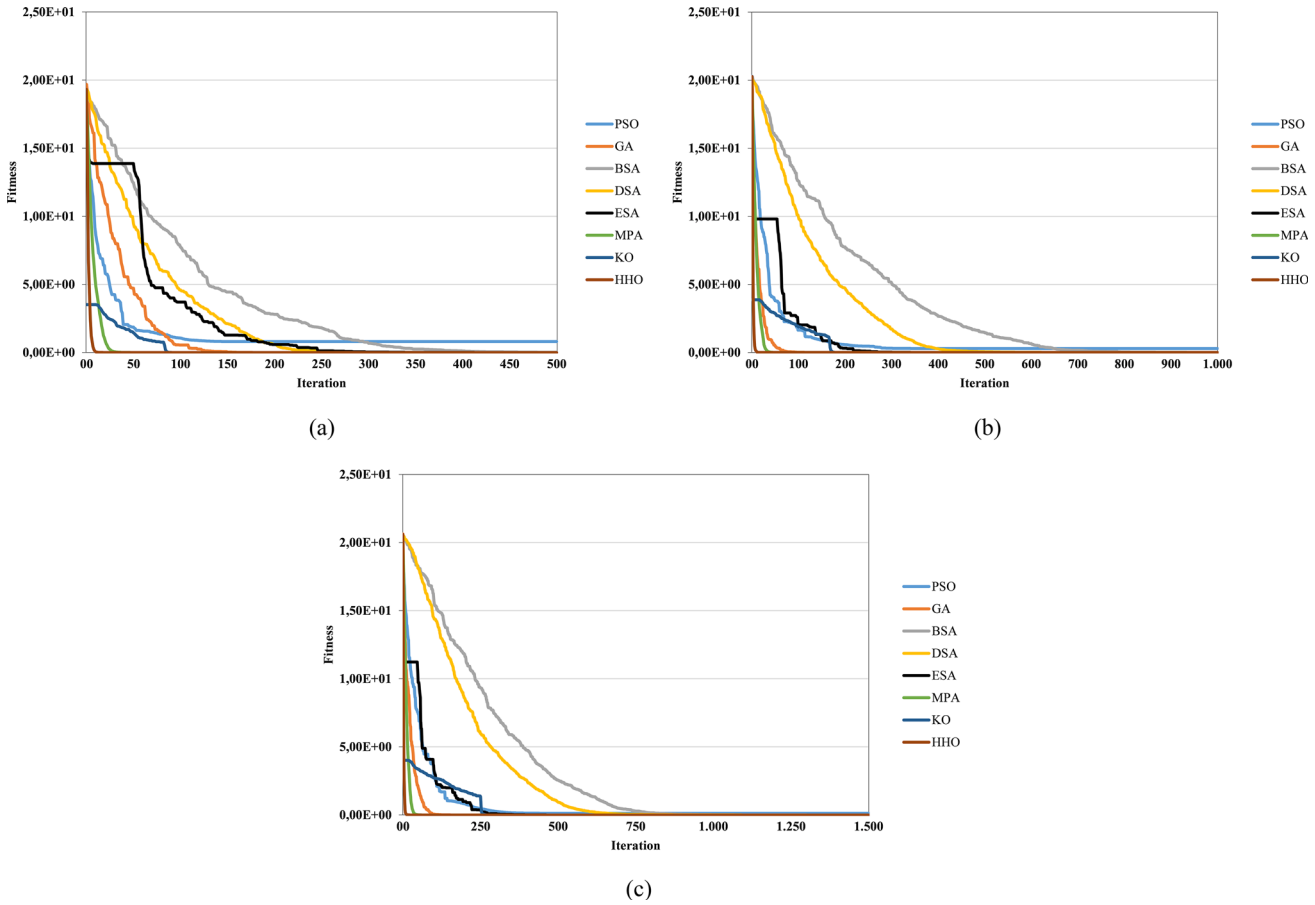


Fig. 8 Ackley function in **a** 10 dimension, **b** 20 dimension and **c** 30 dimension

The competition results and analysis are published in [58]. We can see in Table 8 that our proposed ESA algorithm has a score of 63,96. Due to our hardware limitations, we have to limit our iteration limit to $10E + 06$ and population size to 30. If we look at the competition results average iteration is between $7 * 10E + 8$ and $8 * 10E + 10$, and the population size is between 200 and 31623. Even with the hardware limitations, our proposed algorithm achieved good results. If we examine the [58], our score is above the bottom in the primary and middle in the secondary algorithms.

4.4 Comparison of the results of the clustering problem

We used the Davies–Bouldin index (DB-Index) for the performance analysis of the clustering problem. For all datasets, cluster number ”k” is started with two and ends with ten. It means that we have thirty-six test cases. We calculated the Davies–Bouldin index for each test case for ESA and K-Means algorithm and compared them. The comparison of the DB-Indexes is given in Table 9, and graphical comparisons of the DB-Indexes for the ESA and K-Means algorithm are given in Fig. 11.

We can see in Table 9 that ESA has better clustering performance than K-Means in all test cases. Also, if we look at the dataset basis, ESA conducted better clustering than K-Means on each dataset. In test cases where k is two, ESA and K-Means algorithms have the same DB-Index. The situation mentioned earlier refers to the fact that ESA has conducted the same performance on the clustering problem as the popular clustering algorithm K-Means. That proves that our algorithm can solve the clustering problem.

DB-Indexes are used to determine the optimal number ”k” and perform clustering analysis. Smaller DB-Indexes indicate generated clusters are compact and far from each other. For example, we can see in Fig. 11 that ESA has generated better clusters than the K-Means algorithm on all datasets for all tested k values.

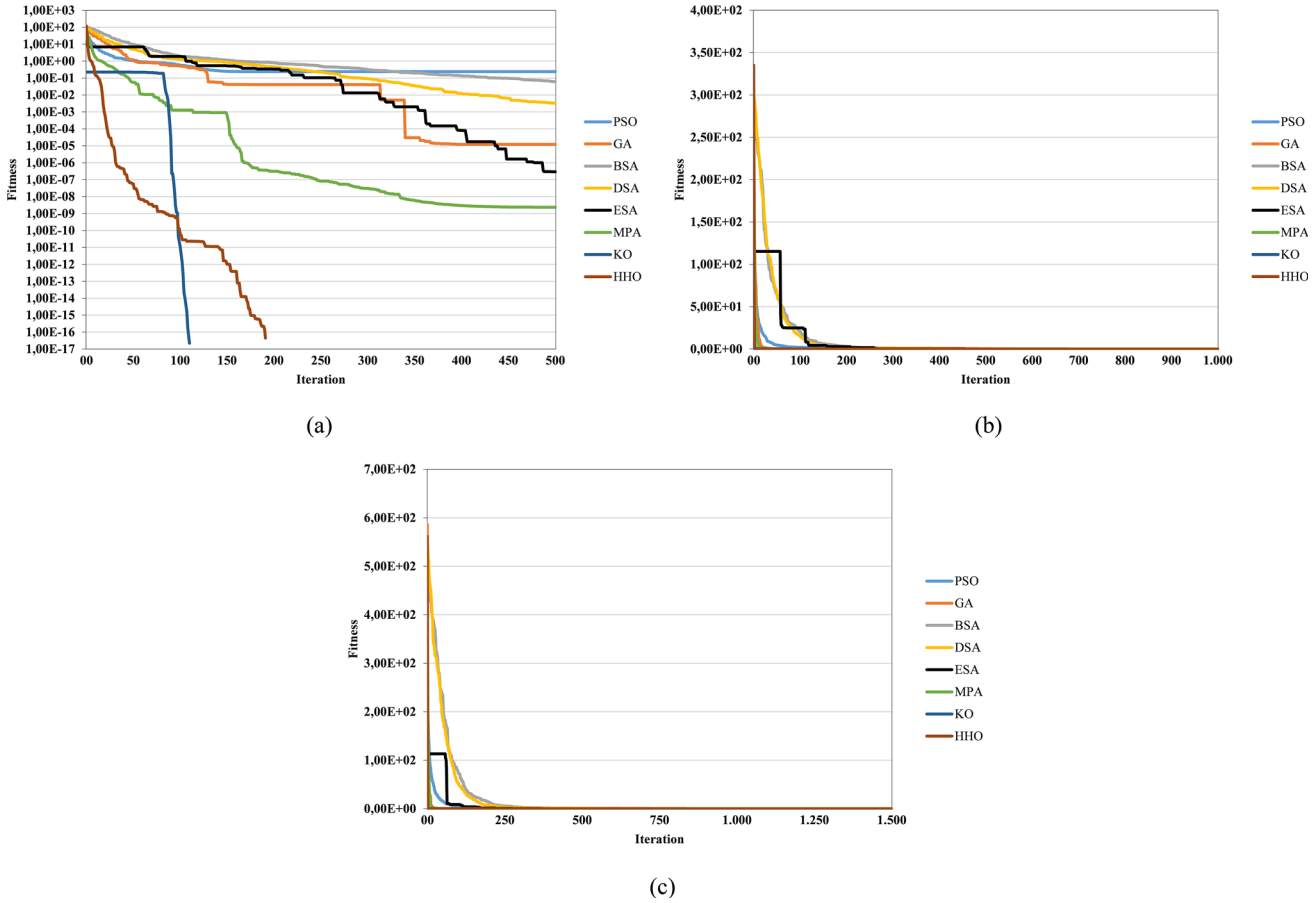


Fig. 9 Griewank function in **a** 10 dimension, **b** 20 dimension and **c** 30 dimension

5 Conclusion

This paper introduces a new nature-inspired metaheuristic optimization algorithm called the ESA. It is based on the movement of electricity in high-resistive areas. Like other metaheuristic algorithms, the ESA needs a population to start a search. This population consists of search agents who have a memory and a lifespan. Also, the ESA has a unique structure called negative and positive poles. Suppose we match these terms with electrical terms. Then, we can say that population is like the voltage, search agents are electrons, and pole structure is electrical polarity structure like negative and positive. Thus, electrons in opposite poles tend to move to each other. This move tendency is the basis of the search mechanism of the ESA.

To determine the efficiency and reliability of the proposed algorithm, the ESA is tested with four benchmark functions and solved the clustering problem, which is an np-hard problem with four different datasets. Benchmark functions are commonly used in literature, and data sets are well-known datasets. All benchmark function results are compared with seven different algorithms: GA, PSO, BSA, DS, MPA, KO, and HHO, and these algorithms have similar search mechanisms to the proposed ESA. As a result, the proposed ESA exhibits gratifying exploitation, exploration, and convergence characteristics with respect to GA, PSO, BSA, and DS and have similar results with MPA, KO, and HHO. Furthermore, with a great convergence rate, the ESA provides better results than the GA, PSO, BSA, and DS. Most of the time, while other algorithms cannot find the exact solutions, the ESA finds the exact solutions. Besides that, the statistical findings support the test results and show that ESA performs better than the other compared algorithms. The test results of the 100 Digit Challenge show us that our algorithm can handle complex problems even with limitations. Furthermore, ESA handled the problem efficiently for clustering performance and performed better clustering capability than the K-Means algorithm.

We can see that if we increase the time and iteration limit for solving the problems, our proposed algorithm can yield better results. Because of the design of the proposed algorithm, the search mechanism works at a slow but persistent rate to search valuable search spaces. Instead of using big steps for exploration, our algorithm uses mostly small steps.

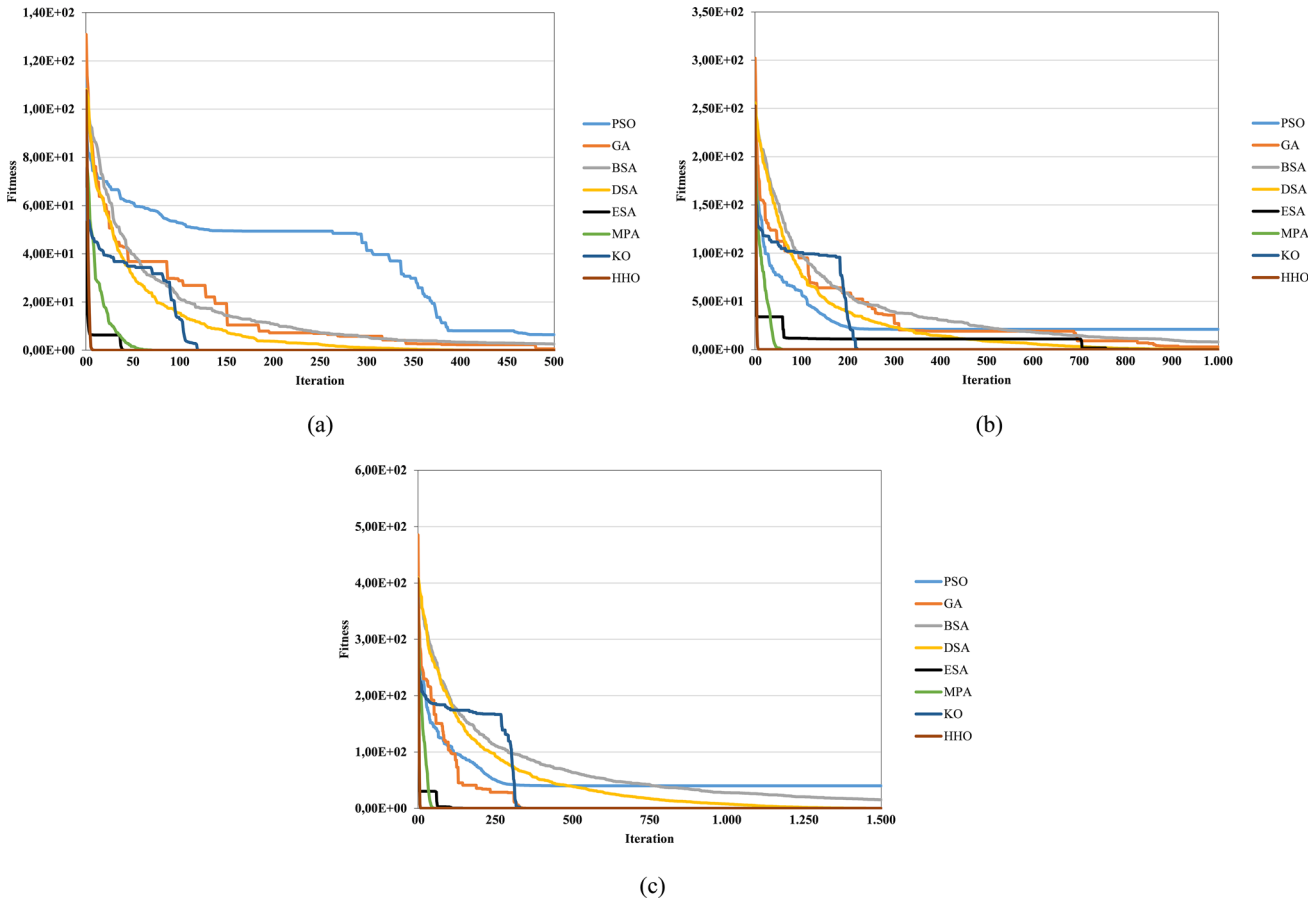


Fig. 10 Rastrigin function in **a** 10 dimension, **b** 20 dimension and **c** 30 dimension

Table 5 Mean ranks of the algorithms

Algorithm	Mean rank
PSO	7,33
GA	4,17
BSA	7,08
DS	5,92
MPA	3,13
KO	3,21
HHO	2,21
ESA	2,96

Table 6 Friedman test statistics of the algorithms

Friedman test statistics	
N	12
Chi-Square X^2	60, 842672
df	7
p	$1,0245E-10$

Therefore the iteration number for finding the exact solution is increased. Thanks to the failure mechanism and the initialization scheme, even in challenging problems proposed algorithm never traps at the local points. Results show that the proposed algorithm can get near the exact solution but, due to iteration limits, in some cases, never reaches the exact solution. The proposed algorithm differs from other meta-heuristics compared to its initialization scheme, pole search mechanism, and update strategy of the best solutions.

We used a fixed ratio for exploration and exploitation movements. A dynamic ratio can be applied to the algorithm to overcome the slowness and late convergence. We used the diffusion coefficient of wood in our experiments; for future work, different diffusion coefficients can be used, and the effect on the algorithm can be compared. In addition, different initialization schemes of the initial population can be examined.

Table 7 *p* values for the Wilcoxon signed rank test

Algorithm	PSO	GA	BSA	DS	MPA	KO	HHO	ESA
PSO	1	0,049860204	0,157939311	0,136097381	0,002217721	0,049860204	0,002217721	0,002217721
GA		1	0,002217721	0,346521712	0,020879263	0,085281059	0,020767229	0,042522478
BSA			1	0,034170473	0,002217721	0,002217721	0,002217721	0,002217721
DS				1	0,002217721	0,002217721	0,002217721	0,004741768
MPA					1	0,498962299	0,017960478	1
KO						1	0,10880943	0,400236144
HHO							1	0,035465865
ESA								1

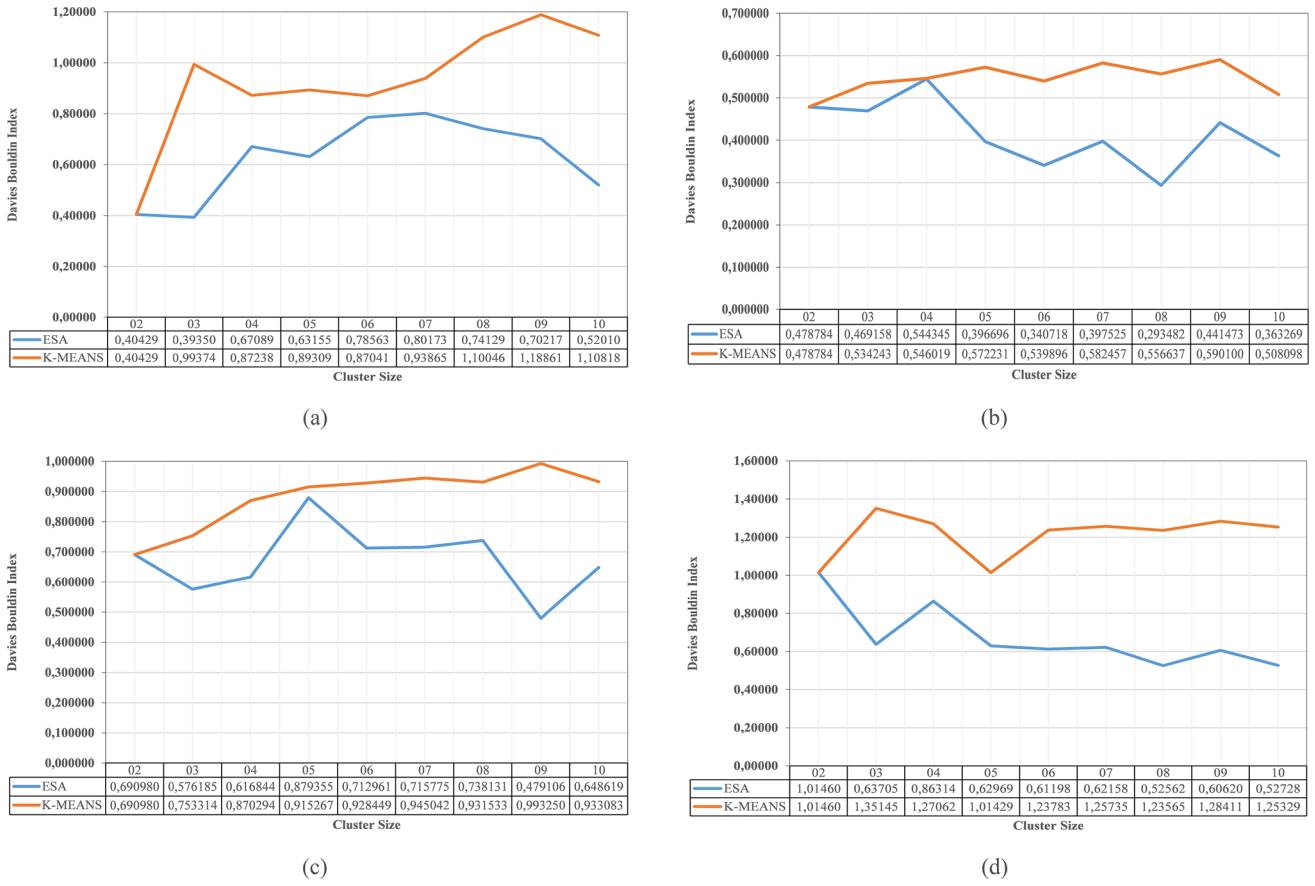
Table 8 Fifty runs for each function sorted by the number of correct digits

Function	Number of correct digits											Score
	0	1	2	3	4	5	6	7	8	9	10	
1	0	0	0	0	0	0	0	0	0	0	50	10
2	0	0	1	1	24	23	0	0	0	0	0	4,32
3	0	0	0	0	0	0	0	0	0	0	50	10
4	0	0	5	11	34	0	0	0	0	0	0	3,58
5	0	0	0	0	0	1	4	34	10	1	0	7,12
6	0	0	0	9	22	4	5	10	0	0	0	4,7
7	0	0	38	2	4	4	2	0	0	0	0	2,6
8	0	0	0	0	0	0	4	2	6	37	1	8,58
9	0	0	0	0	0	0	0	0	0	0	50	10
10	0	0	4	40	5	1	0	0	0	0	0	3,06
											Total:	63,96

Table 9 Comparison of the DB-indexes

Dataset	Algorithm	K=2	K=3	K=4	K=5	K=6	K=7	K=8	K=9	K=10
Iris	ESA	**0,40429**	**0,39350**	**0,67089**	**0,63155**	**0,78563**	**0,80173**	**0,74129**	**0,70217**	**0,52010**
	K-Means	**0,40429**	0,99374	0,87238	0,89309	0,87041	0,93865	1,10046	1,18861	1,10818
Wine	ESA	**0,478784**	**0,469158**	**0,544345**	**0,396696**	**0,340718**	**0,397525**	**0,293482**	**0,441473**	**0,363269**
	K-Means	**0,478784**	0,534243	0,546019	0,572231	0,539896	0,582457	0,556637	0,590100	0,508098
Seeds	ESA	**0,690980**	**0,576185**	**0,616844**	**0,879355**	**0,712961**	**0,715775**	**0,738131**	**0,479106**	**0,648619**
	K-Means	**0,690980**	0,753314	0,870294	0,915267	0,928449	0,945042	0,931533	0,993250	0,933083
HCV	ESA	**1,01460**	**0,63705**	**0,86314**	**0,62969**	**0,61198**	**0,62158**	**0,52562**	**0,60620**	**0,52728**
	K-Means	**1,01460**	**1,35145**	**1,27062**	**1,01429**	**1,23783**	**1,25735**	**1,23565**	**1,28411**	**1,25329**

Bold values are the smallest values of each corresponding test case



	02	03	04	05	06	07	08	09	10
ESA	0,40429	0,39350	0,67089	0,63155	0,78563	0,80173	0,74129	0,70217	0,52010
K-MEANS	0,40429	0,99374	0,87238	0,89309	0,87041	0,93865	1,10046	1,18861	1,10818

	02	03	04	05	06	07	08	09	10
ESA	0,478784	0,469158	0,544345	0,396696	0,340718	0,397525	0,293482	0,441473	0,363269
K-MEANS	0,478784	0,534243	0,546019	0,572231	0,539896	0,582457	0,556637	0,590100	0,508098

	02	03	04	05	06	07	08	09	10
ESA	0,690980	0,576185	0,616844	0,879355	0,712961	0,715775	0,738131	0,479106	0,648619
K-MEANS	0,690980	0,753314	0,870294	0,915267	0,928449	0,945042	0,931533	0,993250	0,933083

	02	03	04	05	06	07	08	09	10
ESA	1,01460	0,63705	0,86314	0,62969	0,61198	0,62158	0,52562	0,60620	0,52728
K-MEANS	1,01460	1,35145	1,27062	1,01429	1,23783	1,25735	1,23565	1,28411	1,25329

Fig. 11 DB-indexes of ESA and K-means algorithm on **a** Iris dataset, **b** Wine dataset, **c** Seeds dataset and **d** HCV dataset

References

1. Wright, S.J.: Optimization. Encyclopedia Britannica (2016). https://www.britannica.com/science/optimization. Accessed 29 June 2021
2. Eiselt, H.A.; Sandblom, C.-L.: Heuristic algorithms. In: Integer Programming and Network Models, pp. 229–258. Springer, Berlin, Heidelberg (2000). https://doi.org/10.1007/978-3-662-04197-0_11
3. Wolpert, D.H.; Macready, W.G.: No free lunch theorems for optimization. IEEE Trans. Evolut. Comput. **1**(1), 67–82 (1997). https://doi.org/10.1109/4235.585893
4. Ho, L.V.; Trinh, T.T.; De Roeck, G.; Bui-Tien, T.; Nguyen-Ngoc, L.; Abdel Wahab, M.: An efficient stochastic-based coupled model for damage identification in plate structures. Eng. Fail. Anal. **131**, 105866 (2022). https://doi.org/10.1016/j.engfailanal.2021.105866
5. Al Thobiani, F.; Khatir, S.; Benaissa, B.; Ghandourah, E.; Mirjalili, S.; Abdel Wahab, M.: A hybrid PSO and Grey Wolf optimization algorithm for static and dynamic crack identification. Theor. Appl. Fract. Mech. **118**, 103213 (2022). https://doi.org/10.1016/j.tafmec.2021.103213
6. Ho, L.V.; Nguyen, D.H.; Mousavi, M.; De Roeck, G.; Bui-Tien, T.; Gandomi, A.H.; Wahab, M.A.: A hybrid computational intelligence approach for structural damage detection using marine predator algorithm and feedforward neural networks. Comput. Struct. **252**, 106568 (2021). https://doi.org/10.1016/j.compstruc.2021.106568
7. Cuong-Le, T.; Minh, H.-L.; Khatir, S.; Wahab, M.A.; Tran, M.T.; Mirjalili, S.: A novel version of Cuckoo search algorithm for solving optimization problems. Expert Syst. Appl. **186**, 115669 (2021). https://doi.org/10.1016/j.eswa.2021.115669
8. Holland, J.H.: Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. The MIT Press, Cambridge (1992)
9. Kennedy, J.; Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN'95 - International Conference on Neural Networks, vol. **4**, pp. 1942–1948. (1995). IEEE, Perth, WA, Australia (1995) https://doi.org/10.1109/ICNN.1995.488968. Accessed 29 June 2021
10. Civicioglu, P.: Backtracking search optimization algorithm for numerical optimization problems. Appl. Math. Comput. **219**(15), 8121–8144 (2013). https://doi.org/10.1016/j.amc.2013.02.017
11. Civicioglu, P.: Transforming geocentric cartesian coordinates to geodetic coordinates by using differential search algorithm. Comput. Geosci. **46**, 229–247 (2012). https://doi.org/10.1016/j.cageo.2011.12.011
12. Faramarzi, A.; Heidarinejad, M.; Mirjalili, S.; Gandomi, A.H.: Marine predators algorithm: a nature-inspired metaheuristic. Expert Syst. Appl. **152**, 113377 (2020). https://doi.org/10.1016/j.eswa.2020.113377
13. Minh, H.-L.; Sang-To, T.; Abdel Wahab, M.; Cuong-Le, T.: A new metaheuristic optimization based on K-means clustering algorithm and its application to structural damage identification. Knowl. Based Syst. **251**, 109189 (2022). https://doi.org/10.1016/j.knosys.2022.109189
14. Heidari, A.A.; Mirjalili, S.; Faris, H.; Aljarah, I.; Mafarja, M.; Chen, H.: Harris hawks optimization: Algorithm and applications.

Future Gener. Comput. Syst. **97**, 849–872 (2019). https://doi.org/10.1016/j.future.2019.02.028
15. Nanda, S.J.; Panda, G.: A survey on nature inspired metaheuristic algorithms for partitional clustering. Swarm Evol. Comput. **16**, 1–18 (2014). https://doi.org/10.1016/j.swevo.2013.11.003
16. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, vol. **1**, pp. 281–297. University of California Press, Berkeley (1967)
17. Bezdek, J.C.; Boggavarapu, S.; Hall, L.O.; Bensaid, A.: Genetic algorithm guided clustering. In: Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence, pp. 34–39. IEEE, Orlando, FL, USA (1994). https://doi.org/10.1109/ICEC.1994.350046. Accessed 29 June 2021
18. Sarkar, M.; Yegnanarayana, B.; Khemani, D.: A clustering algorithm using an evolutionary programming-based approach. Pattern Recognit. Lett. **18**(10), 975–986 (1997). https://doi.org/10.1016/S0167-8655(97)00122-0
19. Kuncheva, L.I.; Bezdek, J.C.: Selection of cluster prototypes from data by a genetic algorithm. In: Fifth European Congress on Intelligent Techniques and Soft Computing, pp. 1683–1688 (1997)
20. Murthy, C.A.; Chowdhury, N.: In search of optimal clusters using genetic algorithms. Pattern Recognit. Lett. **17**(8), 825–832 (1996). https://doi.org/10.1016/0167-8655(96)00043-8
21. Maulik, U.: Genetic algorithm-based clustering technique. Pattern Recognit. **33**, 1455–1465 (2000)
22. Krishna, K.; Narasimha Murty, M.: Genetic K-means algorithm. IEEE Trans. Syst. Man Cybern. Part B (Cybern.) **29**(3), 433–439 (1999). https://doi.org/10.1109/3477.764879
23. Sheng, W.; Liu, X.: A hybrid algorithm for k-medoid clustering of large data sets. In: Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753), pp. 77–82. IEEE, Portland, OR, USA (2004). https://doi.org/10.1109/CEC.2004.1330840. Accessed 29 June 2021
24. Lu, Y.; Lu, S.; Fotouhi, F.; Deng, Y.; Brown, S.J.: FGKA: a fast genetic K-means clustering algorithm. In: Proceedings of the 2004 ACM Symposium on Applied Computing. SAC '04, pp. 622–623. Association for Computing Machinery, Nicosia, Cyprus (2004). https://doi.org/10.1145/967900.968029. Accessed 29 June 2021
25. Lu, Y.; Lu, S.; Fotouhi, F.; Deng, Y.; Brown, S.J.: Incremental genetic K-means algorithm and its application in gene expression data analysis. BMC Bioinform. **5**(1), 172 (2004). https://doi.org/10.1186/1471-2105-5-172
26. Cowgill, M.C.; Harvey, R.J.; Watson, L.T.: A genetic algorithm approach to cluster analysis. Comput. Math. Appl. **37**(7), 99–108 (1999). https://doi.org/10.1016/S0898-1221(99)00090-5
27. Tseng, L.; Yang, S.-B.: A genetic approach to the automatic clustering problem. Pattern Recognit. (2001). https://doi.org/10.1016/S0031-3203(00)00005-4
28. Bandyopadhyay, S.; Maulik, U.: Nonparametric genetic clustering: comparison of validity indices. IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.) **31**(1), 120–125 (2001). https://doi.org/10.1109/5326.923275
29. Bandyopadhyay, S.; Maulik, U.: Genetic clustering for automatic evolution of clusters and application to image classification. Pattern Recognit. **35**(6), 1197–1208 (2002). https://doi.org/10.1016/S0031-3203(01)00108-X
30. van der Merwe, D.W.; Engelbrecht, A.P.: Data clustering using particle swarm optimization. In: The 2003 Congress on Evolutionary Computation, 2003. CEC '03., pp. 215–220. IEEE, Canberra, ACT, Australia (2003). https://doi.org/10.1109/CEC.2003.1299577. Accessed 29 June 2021
31. Cohen, S.C.M.; de Castro, L.N.: Data clustering with particle swarms. In: 2006 IEEE International Conference on Evolutionary Computation, pp. 1792–1798. IEEE, Vancouver, BC, Canada (2006). https://doi.org/10.1109/CEC.2006.1688524. Accessed 29 June 2021
32. Chuang, L.-Y.; Hsiao, C.-J.; Yang, C.-H.: Chaotic particle swarm optimization for data clustering. Expert Syst. Appl. **38**(12), 14555–14563 (2011). https://doi.org/10.1016/j.eswa.2011.05.027
33. Yang, F.; Sun, T.; Zhang, C.: An efficient hybrid data clustering method based on K-harmonic means and Particle Swarm Optimization. Expert Syst. Appl. **36**(6), 9847–9852 (2009). https://doi.org/10.1016/j.eswa.2009.02.003
34. Huang, K.Y.: A hybrid particle swarm optimization approach for clustering and classification of datasets. Knowl. Based Syst. **24**(3), 420–426 (2011). https://doi.org/10.1016/j.knosys.2010.12.003
35. Xu, R.; Xu, J.; Wunsch, D.C.: Clustering with differential evolution particle swarm optimization. In: IEEE Congress on Evolutionary Computation, pp. 1–8 (2010). https://doi.org/10.1109/CEC.2010.5586257. ISSN: 1941-0026
36. Kaya, Y.; Uyar, M.; Tekin, R.: A Novel Crossover Operator for Genetic Algorithms: Ring Crossover. arXiv:1105.0355 [cs] (2011). Accessed 2021-06-29
37. Jalali Varnamkhasti, M.; Lee, L.S.; Abu Bakar, M.R.; Leong, W.J.: A genetic algorithm with fuzzy crossover operator and probability. Adv. Oper. Res. **2012**, 1–16 (2012). https://doi.org/10.1155/2012/956498
38. Vrajitoru, D.: Crossover improvement for the genetic algorithm in information retrieval. Inf. Process. Manag. **34**(4), 405–415 (1998). https://doi.org/10.1016/S0306-4573(98)00015-6
39. Srinivas, M.; Patnaik, L.M.: Adaptive probabilities of crossover and mutation in genetic algorithms. IEEE Trans. Syst. Man Cybern. **24**(4), 656–667 (1994). https://doi.org/10.1109/21.286385
40. Lipowski, A.; Lipowska, D.: Roulette-wheel selection via stochastic acceptance. Phys. A **391**(6), 2193–2196 (2012). https://doi.org/10.1016/j.physa.2011.12.004
41. Abuiziah, I.; Shakarneh, N.: A review of genetic algorithm optimization: operations and applications to water pipeline sdystems. Int. J. Math. Comput. Sci. **7**(12), 1782–1788 (2014)
42. Sivaraj, R.; Ravichandran, T.: A review of selection methods in genetic algorithm. Int. J. Eng. Sci. Technol. **3**(5), 3792–3797 (2011)
43. Razali, N.; Geraghty, J.: Genetic algorithm performance with different selection strategies in solving TSP. In: Proceedings of the World Congress on Engineering, vol. **2**. London, pp. 1–6 (2011)
44. Abdoun, O.; Abouchabaka, J.: A comparative study of adaptive crossover operators for genetic algorithms to resolve the traveling salesman problem. Int. J. Comput. Appl. **31**(11), 49–57 (2011)
45. Einstein, A.; Brown, R.: Investigations on the Theory of the Brownian Movement. Dover Publ., New York (1967)
46. Anonymous: How To Electrify Wood | Cut The Wood. https://cutthewood.com/diy/how-to-electrify-wood/ Accessed 29 June 2021
47. Price, K.V.; Awad, N.H.; Ali, M.Z.; Suganthan, P.N.: Problem Definitions and Evaluation Criteria for the 100-Digit Challenge Special Session and Competition on Single Objective Numerical Optimization. Technical report, Nanyang Technological University, Singapore (2018). http://www.ntu.edu.sg/home/epnsugan/
48. Fisher, R.A.: UCI Machine Learning Repository: Iris Data Set (1988). https://archive.ics.uci.edu/ml/datasets/iris. Accessed 29 June 2021
49. Forina, M.; Leardi, R.; Armanino, C.; Lanteri, S.: UCI Machine Learning Repository: Wine Data Set (1991). https://archive.ics.uci.edu/ml/datasets/Wine. Accessed 29 June 2021
50. Charytanowicz, M., Niewczas, J., Kulczycki, P., Kowalski, P., Lukasik, S.: UCI machine learning repository: seeds data set (2012). https://archive.ics.uci.edu/ml/datasets/seeds Accessed 29 June 2021
51. Lichtinghagen, R.; Klawonn, F.; Hoffmann, G.: UCI Machine Learning Repository: HCV data Data Set (2020). https://archive.ics.uci.edu/ml/datasets/HCV+data Accessed 29 June 2021

52. Davies, D.L.; Bouldin, D.W.: A cluster separation measure. IEEE Trans. Pattern Anal. Mach. Intell. **1**(2), 224–227 (1979). https://doi.org/10.1109/TPAMI.1979.4766909
53. Conover, W.J.: Practical Nonparametric Statistics, vol. 350. Wiley, New York (1999)
54. Friedman, M.: The use of ranks to avoid the assumption of normality implicit in the analysis of variance. J. Am. Stat. Assoc. **32**(200), 675–701 (1937). https://doi.org/10.1080/01621459.1937.10503522
55. Friedman, M.: A comparison of alternative tests of significance for the problem of *m* rankings. Ann. Math. Stat. **11**(1), 86–92 (1940). https://doi.org/10.1214/aoms/1177731944
56. Sheskin, D.J.: Handbook of Parametric and Nonparametric Statistical Procedures. Chapman and Hall/CRC, New York (2003)
57. Wilcoxon, F.: Individual comparisons of grouped data by ranking methods. J. Econ. Entomol. **39**(2), 269–270 (1946). https://doi.org/10.1093/jee/39.2.269
58. Price, K.V.; Awad, N.H.; Ali, M.Z.; Suganthan, P.N.: The 2019 100-Digit Challenge on Real-Parameter, Single Objective Optimization: Analysis of Results. Technical report, Nanyang Technological University, Singapore (2019). http://www.ntu.edu.sg/home/epnsugan/