




Article

Analyzing Physics-Inspired Metaheuristic Algorithms in Feature Selection with K-Nearest-Neighbor

Jayaraju Priyadarshini ¹, Mariappan Premalatha ¹ , Robert Čep ² , Murugan Jayasudha ¹ and Kanak Kalita ^{3,*} 

- ¹ School of Computer Science and Engineering, Vellore Institute of Technology, Chennai 600027, India
² Department of Machining, Assembly and Engineering Metrology, Faculty of Mechanical Engineering, VSB-Technical University of Ostrava, 17. Listopadu 2172/15, 708 00 Ostrava, Czech Republic
³ Department of Mechanical Engineering, Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Avadi 600062, India
* Correspondence: drkanakkalita@veltech.edu.in

Abstract: In recent years, feature selection has emerged as a major challenge in machine learning. In this paper, considering the promising performance of metaheuristics on different types of applications, six physics-inspired metaphor algorithms are employed for this problem. To evaluate the capability of dimensionality reduction in these algorithms, six diverse-natured datasets are used. The performance is compared in terms of the average number of features selected (AFS), accuracy, fitness, convergence capabilities, and computational cost. It is found through experiments that the accuracy and fitness of the Equilibrium Optimizer (EO) are comparatively better than the others. Finally, the average rank from the perspective of average fitness, average accuracy, and AFS shows that EO outperforms all other algorithms.

Keywords: optimization; non-traditional algorithms; feature reduction; KNN; algorithms



Citation: Priyadarshini, J.; Premalatha, M.; Čep, R.; Jayasudha, M.; Kalita, K. Analyzing Physics-Inspired Metaheuristic Algorithms in Feature Selection with K-Nearest-Neighbor. *Appl. Sci.* **2023**, *13*, 906. <https://doi.org/10.3390/app13020906>

Academic Editors: Juan A. Gómez-Pulido and Michele Girolami

Received: 12 September 2022

Revised: 30 December 2022

Accepted: 6 January 2023

Published: 9 January 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Data mining is the process of finding meaningful information or extracting knowledge from large amounts of data. Data mining has the challenging problem of dealing with huge data dimensions. When working with data that has a large number of dimensions, even the advantages of technology can be a hassle [1]. The data-mining process may suffer due to a huge number of dimensions. It may also require a lot of computing time and space. Traditional machine-learning (ML) methods cannot handle these huge datasets [2]. The dataset is made up of several samples that collectively give information about a specific case of the problem. Each sample has a variety of attributes or features. The dataset may have several superfluous or duplicate attributes, in addition to its huge dimensionality. The model may be complex, and the dataset may include a substantial amount of noise. The best subset of the useful features that will contribute to the output is chosen via a pre-processing technique called feature selection (FS) [2]. FS can reduce the training time as well as the huge number of dimensions in the data. Moreover, the model's accuracy is enhanced in addition to the simplification of the model and the best utilization of computing resources [3].

The two main FS approaches are wrapper methods and filter methods. The major drawback of the filter methods is that they work independently of the ML classifiers and do not take any input from them [4]. Meanwhile, the wrapper method uses the classifier directly and picks the features using an optimization algorithm [5]. Optimization algorithms provide the advantage of choosing an optimal or nearly optimal subset of features in a reasonable amount of time as opposed to the conventional exhaustive search. An exhaustive search becomes impractical, because it finds the solution by creating all feasible feature subsets (2^m different solutions for m features) [6]. In the literature, optimization algorithms are categorized into several groups, such as evolution-based algorithms,

swarm-based algorithms, human behavior-inspired algorithms, physics-inspired algorithms, etc. [7]. Swarm-based algorithms mimic the collective but decentralized intelligence of living creatures, such as birds [8], wolves [9], whales [10], bacteria [11], etc. Evolutionary algorithms mimic the emergence of the fittest and healthiest individuals over generations. A few examples are the Genetic Algorithm (GA) [12], Differential Evolution (DE) [13], Biogeography-Based Optimization (BBO) [14], etc. Human behavior-inspired algorithms mimic the collective intelligent behavior of human beings in different real-life situations, such as politics [15], sports [16], corporations [7], etc. Finally, physics-based algorithms are inspired by the laws of nature, such as the gravitational law [17], black holes [18], galaxies [19], etc.

In recent years, metaphor-based algorithms have extensively been used to solve FS problems from different domains. Examples include feature selection using Particle Swarm Optimization (PSO) for document clustering [20], the use of a real-valued Grasshopper Optimization Algorithm (GOA) for feature selection [21], hybridization of the Whale Optimization Algorithm (WOA) and Simulated Annealing (SA) for the feature selection problem [22], feature selection for intrusion detection in wireless mesh networks incorporating genetic operators in WOA [23], the incorporation of levy flight and opposition-based learning in chaotic Cuckoo Search (CS) for feature selection [24], feature selection using Moth Flame Optimization (MFO) [25], feature selection using the Firefly Algorithm (FA) [26], the hybridization of SA with Harris Hawk Optimization (HHO) for the feature selection problem [27] and feature selection using binary Teaching–Learning-Based Optimization (TLBO).

According to the No-Free-Lunch (NFL) theorem [28], no single optimization algorithm is capable of solving every optimization problem by outperforming all other optimization techniques. Because of this, one optimizer can perform better than the others on some problems, but not on all of them. Hence, it is crucial to compare several optimization algorithms on a variety of datasets to find the optimum solution to the feature selection problem. Since there are hundreds of optimization algorithms in the literature, in this study, a few well-known and highly cited physics-inspired algorithms are chosen for this purpose. The rationale is to carry out a comparison of the various metaphors drawn from physics and evaluate their effectiveness. To evaluate the performance of these algorithms, six small-to-large-sized classification datasets are used. The accuracy, convergence, and average fitness of these algorithms are compared. This paper has the following contributions:

- The main novelty of our paper lies in its comparative analysis of six well-cited physics-inspired metaphor algorithms for the problem of feature selection.
- To the best of our knowledge, this is the first time these physics-inspired algorithms have been compared for this specific problem, and our findings provide valuable insights into their performance.
- Our study also has broader implications for the field of machine learning and data mining, as it helps to shed light on the effectiveness of different optimization algorithms for feature selection.
- Our work contributes to the growing body of research on metaheuristics and their potential applications in machine learning and data mining, and it highlights the potential value of using physics-inspired optimization algorithms for feature selection.
- Additionally, our use of variable-sized classification datasets allows us to assess the applicability of these algorithms on a wide range of problems, making our results more generalizable and applicable to practitioners.

Overall, we believe that our paper represents a significant contribution to the field and has the potential to impact the way practitioners approach the problem of feature selection. The rest of the paper is organized as follows. The methodology is discussed in Section 2. Section 3, namely, the Results and Discussion, covers the results and comparative analysis of all six algorithms, and the concluding remarks are given in Section 4.

2. Methodology

2.1. Wrapper Method for Feature Selection

For feature selection, we employed a wrapper method. To accomplish their task, wrapper techniques use a learning algorithm that applies a search strategy to explore the space of feasible feature subsets, ranking them according to the quality of their performance in a specific algorithm. In most cases, wrapper approaches outperform filter methods, since the feature selection process is tailored to the specific classification algorithm being employed. Wrapper methods, on the other hand, are prohibitively time- and resource-intensive for high-dimensional data, since they require evaluating each feature set with the classifier algorithm. Figure 1 depicts the way in which wrapper methods function.

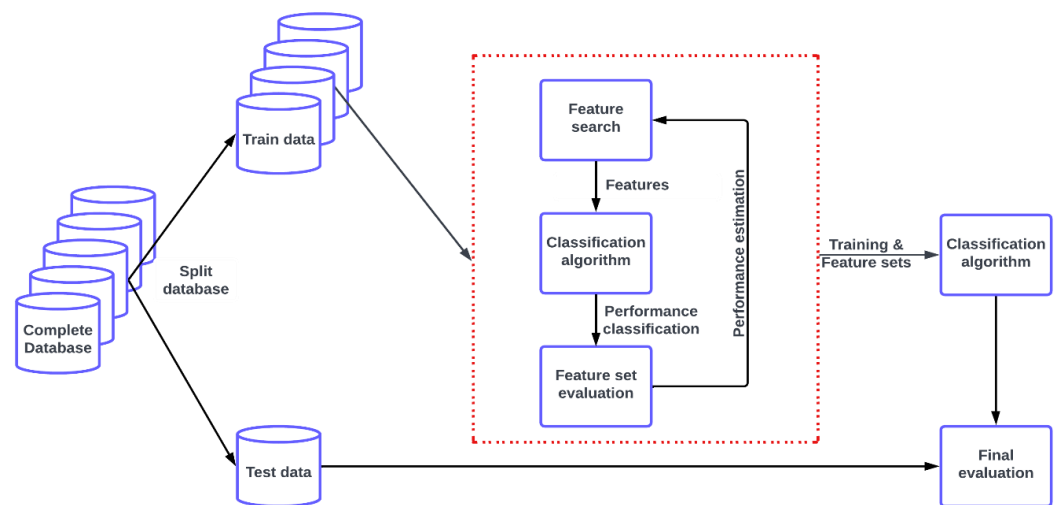


Figure 1. Wrapper feature selection framework.

In this paper, K-Nearest Neighbor (k-NN) is used as the evaluator algorithm. The k-NN method uses a set of K neighbors to determine how an object should be categorized. A positive integer value of K is pre-decided before running the algorithm. To classify a record, the Euclidean distances between the unclassified record and the classified records are determined and ranked.

2.2. Fitness Function

The effectiveness of an optimizer is evaluated by its fitness function. The fitness function in feature selection is dependent on the classification error rate and the number of features used for classification. It is deemed to be a good solution if the selected feature subset reduces the classification error rate and the number of features chosen. The following fitness function is used in this paper [29]:

$$\downarrow \text{Fitness} = \lambda \gamma_S(D) + (1 - \lambda) \frac{|S|}{|F|} \tag{1}$$

where $\gamma_S(D)$ is the classification error computed by the classifier, $|S|$ is the reduced number of features in the new subset, $|F|$ is the total features in the dataset, and $\lambda \in [0, 1]$ is a factor corresponding to the importance of the classification performance and length of the reduced subset.

2.3. Physics-Inspired Metaphor Algorithms

In this paper, six well-cited physics-inspired metaphor algorithms are employed to solve the problem of feature selection. In this section, the functioning of these algorithms and their position-updating mechanisms are discussed.

2.3.1. Simulated Annealing

Simulated Annealing is a fundamental nature-inspired algorithm that was proposed in 1983 by Kirkpatrick et al. [30]. The source of inspiration behind this algorithm is the annealing process of metals. The process of annealing, which starts at a very high temperature and progressively cools down, is used to physically harden metals. The algorithm involves three main parameters, including the cooling rate (c), the final temperature (T_f), and the starting temperature (T_0). The starting temperature is kept very high initially, and the cooling rate gradually reduces until it reaches the final temperature. The process is mimicked by randomly generating a candidate solution. The algorithm runs iteratively, and a new solution is generated in the neighborhood of the current solution in each iteration. The fitness of the current and neighbor solutions is compared. If the fitness of the new solution is better, then the position of the current solution is updated. Moreover, the best solution keeps the best position found so far. The terminating condition of the repetitive process is reaching the T_f . In each iteration, T is updated as follows:

$$T = T * C, \quad 0 < c < 1 \quad (2)$$

SA is a global optimization algorithm, because it can explore as well as exploit the search space. The exploration is performed by updating the current solution with a worse neighboring solution in early iterations based on the value of T and the worse value of the neighboring solution. The chance of accepting the worse neighbor is computed using the following equation:

$$\exp\left(-\frac{\delta}{T}\right) \leq r \quad (3)$$

where \exp is the exponential function, δ is equal to the fitness difference of current and neighboring solutions, and r is randomly generated in the range $[0, 1]$.

2.3.2. Gravitational Search Algorithm

This algorithm is inspired by Newton's law of gravitation and the second law of motion [17]. It treats each candidate solution in the search space as an object whose mass is considered to be its fitness. Heavier objects are considered fitter than lighter objects. The objects are attached to each other with some gravitational force that causes objects to explore the search space. The heaviest object is considered the global best solution. Since the heavier objects attract other objects with more force, the whole population ultimately converges toward the heaviest object, called the global best solution. The algorithm is comprised of a few mathematical equations that are expressed below.

Force calculation: The force from an object j on an object i is calculated using the following equation:

$$F_{ij}^d(t) = G(t)M_{pi}(t) \times \frac{M_{aj}(t)}{R_{ij}(t) + \epsilon} (x_j^d(t) - x_i^d(t)) \quad (4)$$

In the above equation, G is the gravitational constant that controls the search accuracy, M_{pi} is the passive gravitational mass of solution i , M_{aj} is the active gravitational mass of solution j , the distance between solution i and solution j is denoted by R_{ij} , x^d is the position of a solution in d^{th} dimension, and ϵ is a small constant.

The ultimate force on a solution (mass) is calculated by taking the weighted sum of all the forces on that solution from the k_{best} solutions, which are calculated as follows:

$$F_i^d(t) = \sum_{j \in k_{best}, j \neq i} rand_j F_{ij}^d(t) \quad (5)$$

Acceleration calculation: Once the total force on a solution in a particular dimension d is calculated, the acceleration of the solution in that dimension can be computed using the following equation:

$$a_i^d(t) = \frac{F_i^d(t)}{M_{ii}(t)} \tag{6}$$

where M_{ii} is the mass of inertia of solution i .

Velocity calculation: Based on the acceleration, the velocity of a solution can be computed by adding the acceleration to a fraction of the previous velocity of that solution. The equation to compute velocity is given below:

$$v_i^d = rand_i \times v_i^d(t) + a_i^d(t) \tag{7}$$

Position updating: To update the position of a solution, the updated velocity is simply added to the old position of the solution, as formulated below:

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \tag{8}$$

Gravitational constant updating: To update G , the following relation is used:

$$G(t) = G_0 \exp\left(-\alpha \frac{t}{t_{max}}\right) \tag{9}$$

where G_0 is the initial gravitational constant, and α is a constant. t and t_{max} represent the current and final iteration numbers.

2.3.3. Sine Cosine Algorithm

The Sine Cosine Algorithm (SCA) [31] has a very unique source of inspiration. It utilizes two sine and cosine functions to update the position of solutions when searching the space to find the global optimum. The position-updating model of this algorithm is very simple and is formulated below:

$$X_i^{t+1} = \begin{cases} X_i^t + r_1 \times \sin(r_2) \times |r_3 P_i^t - X_i^t|, & r_4 < 0.5 \\ X_i^t + r_1 \times \cos(r_2) \times |r_3 P_i^t - X_i^t|, & r_4 \geq 0.5 \end{cases} \tag{10}$$

In the above equation, X_i denotes a solution in the i^{th} dimension, and P_i denotes the global best solution, namely, the destination solution in the paper. The above equation involves a few other variables that are defined below.

r_1 is an adaptive parameter that is linearly reduced with the course of iterations. It starts from a prefixed value and linearly decreases in each iteration. It is computed as follows:

$$r_1 = \alpha - t \frac{\alpha}{T} \tag{11}$$

where α is constant.

- r_2 is randomly generated in the range of 0 to 2π .
- r_3 is also a random number that is generated in the range of 0 to 2.
- r_4 is also a random number that is generated in the range of 0 to 1, and based on its value, it is decided whether to use the sine function or the cosine function in updating the position of the current solution.

When multiplied by r_1 , the range of values provided by $\sin(r_2)$ and $\cos(r_2)$ shifts from $[-1, 1]$ to $[-2, 2]$. Due to a linear decrease in the values of the parameter r_1 , the range begins at $[-2, 2]$ and linearly declines to $[0, 0]$ during iterations. The position-updating equation of SCA creates two regions around the destination P: an inner region that promotes exploitation and an outer region to promote exploration. The precondition to search the inner region is $\{-1 \leq r_1 \times \cos(r_2) \leq 1\}$ or $\{-1 \leq r_1 \times \sin(r_2) \leq 1\}$,

and the precondition to search the outer region is $\{r_1 X \cos(r_2)\}$, or $\{r_1 X \sin(r_2)\}$ gives a value greater than 1 or lesser than -1 .

2.3.4. Atom Search Optimization

Atom Search Optimization (ASO) [32], which is inspired by molecular dynamics, has shown a tremendous performance on a variety of applications in the literature. Each atom is considered a candidate solution, and the mass is mapped with the fitness in the optimization algorithm, where the higher the mass, the fitter the solution. Every atom in the population pulls or repels other atoms in the search space. The heavier atoms generate more force and pull lighter objects rapidly, and the heavier objects are pulled slowly towards the others due to their mass. The slowly moving atoms create exploitation in the algorithm, because they can search more locally, whereas the rapidly moving atoms allow the algorithm to explore the search space because of longer and quicker jumps. The algorithm starts with random initializations. In every iteration, the atoms move and accelerate, and the location of the atom that has performed the best up to that point is also likewise adjusted. Atomic acceleration is also caused by two other factors: L-J potential and constraint forces. The acceleration helps to update the velocity of the solutions (atoms). Finally, the velocity is added to the previous position to update the current position of the solution. The position-updating mechanism of the algorithm is discussed below.

The population is generated by randomly generating position and velocity vectors for each atom in the population.

$$X_i = [X_i^1, X_i^2, \dots, X_i^D] \quad (12)$$

$$V_i = [V_i^1, V_i^2, \dots, V_i^D] \quad (13)$$

The fitness of each solution in the population is computed, and the global best X_{best} is determined.

The mass of each atom is computed using the following equation:

$$m_i = \frac{M_i}{\sum_{j=1}^N M_j} \quad (14)$$

where M is computed from the fitness of the current solution, the best solution, and the worst solution.

The value of K is computed, where K denotes the size of the subset of atoms:

$$K = N - (N - 2) \sqrt{\frac{t}{T}} \quad (15)$$

where N is the size of the population.

The interaction force on an atom is calculated, which is accomplished using the following equation:

$$F_i^d = \sum_{j \in K} rand_j F_{ij}^d \quad (16)$$

where $rand_j$ is a random number in the range of $[0, 1]$.

The constraint force is computed using the following equation:

$$G_i^d = \lambda (X_{best}^d - X_i^d) \quad (17)$$

where λ is the Langrangian multiplier that is computed as follows:

$$\lambda = \beta e_T^{-20t} \quad (18)$$

where β is the multiplier weight.

Once the mass, constraint forces, and interaction forces are computed, the acceleration is computed as follows:

$$a_i^d = \frac{F_i^d}{m_i^d} + \frac{G_i^d}{m_i^d} \tag{19}$$

Once the acceleration is computed, the velocity of an atom can be computed as follows:

$$V_i^d(t + 1) = r_1 V_i^d(t) + a_i^d(t) \tag{20}$$

Using the updated velocity, the position of a solution is updated as follows:

$$X_i^d(t + 1) = X_i^d(t) + V_i^d(t + 1) \tag{21}$$

2.3.5. Henry Gas Solubility Optimization

Henry Gas Solubility Optimization (HGSO) is inspired by Henry’s gas law [33], which is stated below:

“At a constant temperature, the amount of a given gas that dissolves in a given type and volume of liquid is directly proportional to the partial pressure of that gas in equilibrium with that liquid”.

This law can be interpreted as the partial pressure of a gas and the solubility of that gas being directly proportional. If one increases, then the other increases, too. This relation is expressed through the following equation:

$$S_g = H \times P_g \tag{22}$$

where the gas solubility is denoted by S_g , Henry’s constant is denoted by H , and the partial pressure of the gas is represented by P_g . The proportionality constant H is highly dependent on the temperature, as it varies with the change in the temperature. In HGSO, each gas particle is considered a candidate solution, whereas all particles collectively make up the population. Initially, gas particles (population) are randomly generated, and then gas particles update their positions in the course of iterations by exploring and exploiting the search space. HGSO involves the following steps.

Population initialization: A population of N gas particles is randomly generated using the following equation:

$$X_{i(t+1)} = X_{\{min\}} + r \times (X_{\{max\}} - X_{\{min\}}) \tag{23}$$

where X_i denotes the initial position of the i^{th} solution, $X_{\{min\}}$ and $X_{\{max\}}$ are the lower and upper bounds of the problem function under consideration, r is a randomly generated real number between 0 and 1, and t is the iteration number.

The properties of each search agent in HGSO can be initiated using the following equation:

$$H_{j(t)} = l_1 \times rand(0, 1), P_{\{i,j\}} = l_2 \times rand(0, 1), C_j = l_3 \times rand(0, 1) \tag{24}$$

where $H_{j(t)}$ represents Henry’s constant for the j^{th} cluster, $P_{\{i,j\}}$ denotes the partial pressure of the i^{th} particle in the j^{th} cluster, and C_j indicates the initial constant value for the j^{th} cluster.

Clustering: This step divides the search agents into K clusters to map different types of gases, where the same types of gases are grouped into a cluster. Therefore, each cluster has the same value of Henry’s constant H_j .

Fitness Evaluation: In this step, each search agent in the j^{th} cluster is evaluated through the objective function to find the best solution $X_{j,best}$ in the j^{th} cluster. Once all the clusters are evaluated, then the gases are ranked to find the global best particle X_{best} .

Update Henry’s coefficient: The partial pressure of each gas particle changes in each iteration. Therefore, the value of Henry’s coefficient H_j is updated using the following equation:

$$H(t + 1) = \exp\left\{\left(-C_j \times \left(\frac{1}{T} - \frac{1}{T^0}\right) \times H_{j(t)}\right)\right\}, ; T(t) = \exp\left\{\left(-\frac{t}{t_{\{max\}}}\right)\right\} \quad (25)$$

where H_j represents the value of Henry’s constant for the j^{th} cluster, T indicates the temperature, T^0 denotes a reference temperature equivalent to 298.15 K, and $t_{\{max\}}$ represents the maximum iterations.

Update solubility: In this step, the solubility $S_{\{i,j\}}$ of the i^{th} particle in the j^{th} cluster is updated using the following equation:

$$S_{\{i,j\}(t)} = K \times H_{j(t+1)} \times P_{\{i,j\}(t)} \quad (26)$$

where K is a constant, and $P_{\{i,j\}}$ is the partial pressure of gas i in cluster j .

Update position: The properties of particles computed in the previous steps are utilized to update the position of the i^{th} gas particle in the j^{th} cluster according to the following equation:

$$X_{\{i,j\}(t+1)} = X_{\{i,j\}(t)} + F \times r_1 \times \gamma \times \left(X_{\{j, best\}(t)} - X_{\{i,j\}(t)}\right) + F \times r_2 \times a \times \left(S_{\{i,j\}(t)} \times X_{\{best\}(t)} - X_{\{i,j\}(t)}\right) \quad (27)$$

$$\gamma = \beta \times \exp\left\{\left(\frac{F_{\{best\}(t)} + \epsilon}{F_{\{i,j\}(t)} + \epsilon}\right)\right\}, ; \epsilon = 0.05 \quad (28)$$

where the position of the i^{th} search agent in the j^{th} cluster is represented by X_{ij} , the best agent in the j^{th} cluster is denoted by $X_{j,best}$, and the global best particle in the entire population is represented by X_{best} . Moreover, r_1 and r_2 are two random values in the range $[0, 1]$, t is the current iteration, F is a flag used for diversification purposes and changes the direction of the solution, γ indicates the ability of the i^{th} particle in the j^{th} cluster to interact with other agents in its cluster, a represents the impact of other gases on the i^{th} particle, β is fixed as $\beta = 1$, F_{ij} is the fitness of the i^{th} particle in the j^{th} cluster, and $F_{\{best\}}$ is the fitness of the best particle.

Escape from local optimum: To avoid stagnation in local optima, all the particles are evaluated, and the worst N_w agents are selected and reinitialized using the following equation:

$$N_w = N \times (rand(c_2 - c_1) + c_1), ; c_1 = 0.1 ; and ; c_2 = 0.2 \quad (29)$$

where N is the population size. Moreover, c_1 and c_2 are constants that define the percentage of worst particles.

2.3.6. Equilibrium Optimizer (EO)

Control volume mass balance models, which are used to estimate both dynamic and equilibrium states, serve as an inspiration for the Equilibrium Optimizer (EO), a recently proposed physics-inspired algorithm [34]. The particles are considered to be the solutions, and their positions map the concentration of the particles. This algorithm constructs an equilibrium pool of five reference solutions (four best so-far particles and one arithmetic mean of them) called equilibrium candidates. Each particle updates its position with reference to a randomly selected candidate from the pool. The algorithm is aided by two carefully designed parameters called the exponential term (F) and the generation rate (G). Moreover, a concept of memory saving is used, which allows a solution to update its concentration only if it improves as compared to its previous concentration. The exploration, exploitation, and the balance between them are controlled through these parameters: the equilibrium pool and the generation probability.

EO uses a mass-balance equation to describe the conservation of mass within a system. The generic mass-balance equation is given as:

$$V \frac{dC}{dt} = QC_{\{eq\}} - QC + G \tag{30}$$

where $V \frac{dC}{dt}$ represents the rate of change of mass in a control volume, Q is the flow rate, the concentration at an equilibrium state is denoted by $QC_{\{eq\}}$, and G mimics the mass generation rate. Here, $\frac{dC}{dt}$ can also be solved in terms of $\frac{Q}{V}$ and $\frac{G}{V}$ denoted by λ or the turnover rate (i.e., $\lambda = \frac{Q}{V}$). Therefore, the above equation can be reconstructed as:

$$\frac{dC}{\lambda C_{\{eq\}} - \lambda C + \frac{G}{V}} = dt \tag{31}$$

By taking the integration of the above equation, we obtain:

$$C = C_{\{eq\}} + (C_0 - C_{\{eq\}})F + \frac{G}{\lambda V}(1 - F) \tag{32}$$

which is used as an updating rule for each particle, where F is calculated as follows:

$$F = \exp[-\lambda(t - t_0)] \tag{33}$$

where t_0 and C_0 represent the initial start time and concentration. In this algorithm, each particle is a solution, and its position represents its concentration. The mathematical formulation of EO is discussed in the following steps.

Initialization and function evaluation: The first step is to initialize the particles' concentration according to the following equation:

$$X_{\{m\}}^{\{init\}} = X_{\{min\}} + rand_m(X_{\{max\}} - X_{\{min\}}) \tag{34}$$

where $X_{\{m\}}^{\{init\}}$ represents the initial concentration of the m^{th} particle, $X_{\{max\}}$ shows the maximum, and $X_{\{min\}}$ shows the minimum values.

Equilibrium pool and candidates X_e : In this algorithm, four equilibrium candidates (good solutions) are determined to guide other particles and promote exploration. Moreover, a particle constructed by taking the arithmetic mean of all these candidates is also used, which promotes exploitation. These candidates are then assembled to form an equilibrium pool. Each particle updates its position with respect to a randomly selected candidate from the pool.

Exponential term (E): This term is used in position updating to balance exploration and exploitation. The exponential term is computed as follows:

$$E = e^{-\{\lambda\}(t-t_0)} \tag{35}$$

where t and t_0 are computed by following equations, respectively:

$$t = \left(1 - \frac{Iter}{Max_iter}\right)^{\left(\frac{a_2 Iter}{Max_iter}\right)} \tag{36}$$

$$t_0 = \frac{1}{\lambda} \ln\left(-a_1 sign(\{r\} - 0.5) \left[1 - e^{-\{\lambda\}t}\right]\right) + t \tag{37}$$

In the above equation, a large value of a_1 promotes exploration, and a large value of a_2 promotes exploitation. The $sign(\{r\} - 0.5)$ controls the direction of exploration and exploitation. Using the above equations, E is computed as follows:

$$E = -a_1 sign(\{r\} - 0.5) [e^{-\{\lambda\}t} - 1] \tag{38}$$

Generation rate: In this algorithm, this term is used as a solution finder by taking short steps. The generation rate control parameter shows the probability of the generation term in the updating process. The generation probability is used to calculate the number of particles that employ the generation term to readjust their state. The final position updating of EO based on all the above steps is formulated below:

$$X = \{X\}_{\{eq\}} + \left(\{X\} - \{X\}_{\{eq\}} \right) \cdot \{E\} + \frac{R}{\{\lambda\}V} (1 - E) \quad (39)$$

3. Results and Discussion

In this section, the performance of the previously discussed algorithms is compared on six well-known datasets. All the algorithms are implemented in MATLAB v2019. The experiments are run on a Windows platform with an Intel(R) Core (TM) i7 CPU @3.40 GHz and 24 GB RAM.

3.1. Datasets

To evaluate the performance of the algorithms, six datasets, namely, breast cancer, German, heart, ionosphere, ovarian cancer, and sonar are used. To evaluate the performance from different aspects, we have tried to include mixed types of datasets, including small-featured (heart disease) to large-featured (ovarian cancer) and small-sized (sonar) to large-sized (German) datasets. The details regarding all datasets are given in Table 1.

Table 1. Datasets selected for this study.

| Dataset | Symbol | Number of Instances | Number of Features | Source |
|----------------|--------|---------------------|--------------------|---|
| Breast cancer | DS1 | 569 | 30 | https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(diagnostic) , accessed on 12 September 2022 |
| German | DS2 | 1000 | 24 | https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data) , accessed on 12 September 2022 |
| Heart | DS3 | 303 | 13 | https://archive.ics.uci.edu/ml/datasets/heart+Disease , accessed on 12 September 2022 |
| Ionosphere | DS4 | 351 | 34 | https://archive.ics.uci.edu/ml/datasets/ionosphere , accessed on 12 September 2022 |
| Ovarian cancer | DS5 | 216 | 4000 | Conrads et al. [35] |
| Sonar | DS6 | 208 | 60 | https://archive.ics.uci.edu/ml/datasets/Connectionist+Bench+(Sonar,+Mines+vs.+Rocks) , accessed on 12 September 2022 |

Breast cancer dataset: In this dataset, fine needle aspirate of a breast mass digital image is used to calculate features. The image's cell nuclei are characterized in terms of their appearance and location [36]. The diagnosis, i.e., the response parameter, is binary (M = malignant, B = benign).

German dataset: Prof. Hofmann created the original dataset, which consists of 1000 entries and 20 categorical/symbolic attributes. Each record in this dataset is an individual who has been extended credit by a financial institution. People are ranked as either "good credit risks" or "bad credit risks" based on several factors.

Heart dataset: Cleveland, Hungary, Switzerland, and the Long Beach V database are the four parts of this 1988 dataset. It has a total of 76 traits, including the response attribute, but only 14 have been used in any of the published trials. Whether or not the patient has a cardiac disease is what is being targeted in the "target" section. Zero (0) indicates the absence of disease, and a value of 1 indicates the presence of disease.

Ionosphere dataset: The radar equipment that gathered this data is located in Goose Bay, Labrador. The total transmitted power of this system is on the order of 6.4 kilowatts, and it is generated via a phased array of sixteen high-frequency antennas. Radar returns with a clear ionospheric structure are considered to be of high quality. Those that do not are

considered “bad” returns, since their transmissions are unable to attenuate the ionosphere. All of the 34 features are continuous.

Ovarian cancer dataset: There are a total of 216 patients included in this dataset, 121 of whom have ovarian cancer and 95 of whom do not. There are almost 4000 individual spectroscopic readings provided for each patient, each of which expresses a biomarker. Patients are likely to share many genes and biomarkers due to the substantial correlation between high-dimensional biological and genetic datasets.

Sonar dataset: The 111 patterns in this dataset were created by reflecting sonar sounds off a metal cylinder at different angles and in different environments. It includes 97 more patterns discovered in rocks exposed to the same conditions. The sonar signal being sent out is a frequency-modulated chirp that gradually increases in pitch. For the cylinder, the dataset includes signals collected at angles spanning 90 degrees, and for the rock, signals were collected at angles spanning 180 degrees.

3.2. Parameter Settings

No algorithm may perform well without optimizing its parameters. However, finding the right parameters for an algorithm is itself an optimization problem. To obtain the maximum of each algorithm, several combinations of parameters were tried, in addition to using the combinations suggested by the authors of these algorithms in their original papers. It was also kept in mind that the number of iterations or function evaluations remains the same for all algorithms. The parameter settings used for each algorithm are presented in Table 2.

Table 2. Parameter settings of all algorithms.

| Algorithm | Parameter | Value |
|-----------------------------------|--|-------|
| Common for all algorithms | k | 5 |
| | Maximum iterations (t_{max}) | 200 |
| | Number of search agents (N) | 30 |
| | Number of independent runs | 20 |
| | Ratio of validation data | 0.2 |
| Simulated Annealing | Cooling rate (c) | 0.93 |
| | Initial temperature (T_0) | 100 |
| Gravitational Search Algorithm | Initial gravitational constant (G_0) | 100 |
| | Constant, (α) | 20 |
| Sine Cosine Algorithm | Constant, (α) | 2 |
| Atom Search Optimization | Depth weight, (α) | 50 |
| | Multiplier weight, (β) | 0.2 |
| Henry Gas Solubility Optimization | Number of gas types | 2 |
| | K | 1 |
| | Influence of other gas, (α) | 1 |
| | β | 1 |
| | l_1 | 0.05 |
| | l_2 | 100 |
| Equilibrium Optimizer | l_3 | 0.01 |
| | a_1 | 2 |
| | a_2 | 1 |
| | Generation probability | 0.5 |
| | V | 1 |

To assure consistency in results, each algorithm was run 20 times in a row on all datasets. Furthermore, each dataset was split into training and test sets, where 80% of samples were used for training, and 20% of samples were used for testing. For classification, the k -Nearest Algorithm (KNN) was used. The KNN classifier is a well-liked wrapper method due to its straightforward implementation and the fact that, in comparison to other classifiers, it only requires one parameter, k . The value of k is tuned by performing several experiments, and it was found that $k = 5$ is the most suitable value.

3.3. Performance Evaluation

To evaluate and compare the performance of each algorithm, different experiments were performed, and the performance in terms of fitness, accuracy, mean feature subsets, and convergence was compared.

3.3.1. Fitness Comparison

The fitness of a solution is the value that is returned by the objective function against that solution. It measures how good or bad a solution is. The best fitness attained by each algorithm for each dataset is presented in Table 3. As can be seen from the results, EO outperformed all the other algorithms in five cases. However, the performance of HGSO was comparable in some cases (DS1 and DS5) and was even better in the case of DS4. Based on collective performance, EO can be given the first rank, and HGSO can be given the second rank.

Table 3. Best fitness values of all algorithms.

| Dataset | SA | GSA | SCA | ASO | HGSO | EO |
|---------|---------|---------|----------------|----------------|----------------|----------------|
| DS1 | 0.01869 | 0.01758 | 0.0198 | 0.01157 | 0.0198 | 0.01157 |
| DS2 | 0.22692 | 0.20258 | 0.20092 | 0.19268 | 0.19888 | 0.18153 |
| DS3 | 0.13662 | 0.08635 | 0.10285 | 0.08712 | 0.11781 | 0.06985 |
| DS4 | 0.08574 | 0.04419 | 0.02946 | 0.05951 | 0.01532 | 0.01591 |
| DS5 | 0.02798 | 0.00453 | 0.00001 | 0.00407 | 0.00001 | 0.00001 |
| DS6 | 0.07777 | 0.00333 | 0.02515 | 0.02665 | 0.04863 | 0.02515 |

To have a clear picture and better understanding, the average fitness and standard deviation of each algorithm on all datasets are compared in Figure 2. The average depicts a slightly different picture. The average fitness of EO is no longer at rank 1 on DS1, DS3, and DS6; however, EO retained its rank in the case of DS2 and DS5. Furthermore, EO secured the first rank for the DS4, on which HGSO was better in the case of best fitness. On the contrary, HGSO could not maintain its performance. However, ASO maintained its performance on DS1 and outperformed the other algorithms on DS6. It is important to mention here SA was the worst performer in nearly all cases.

3.3.2. Comparison of Classification Accuracy

In this subsection, the classification accuracy and the average number of features selected by each algorithm are compared. The accuracy is the percentage of correctly classified test samples, and the average features selected (AFS) is the average number of features to which an algorithm reduces the dimensions of a dataset. AFS is computed by taking the average of the number of features selected by an algorithm in all 20 runs. The best accuracies obtained by each algorithm on all datasets are presented in Table 4. It is evident from the results that EO outperformed all other algorithms on all datasets. However, ASO performed equally well in three cases: DS1, DS5, and DS6. Simulated Annealing, on the other hand, was the worst performer on all datasets. It is important to note that the easiest dataset to classify was DS5, because all algorithms attained 100% on this dataset; however, SA was the only one that had an accuracy of less than 100% on this dataset. In this analysis, DS2 remained the toughest benchmark, on which the best performer could not obtain more than 82% accuracy.

In addition to comparing best accuracies, the average accuracy of all runs along with standard deviations are compared in Figure 3. When comparing best accuracies, EO outperformed on all datasets, but due to inconsistent performance in different runs and a slightly higher standard deviation, EO was no longer the best performer on DS1, DS3, and DS6. However, its performance was comparable. On the other hand, ASO managed to outperform EO on DS1 and DS6, and both were equally good at DS2. Moreover, the performance of GSA was noticeably the best on DS3, whereas SA was still the worst performer on all datasets.

Table 4. Best classification accuracy of all algorithms.

| Dataset | SA | GSA | SCA | ASO | HGSO | EO |
|---------|---------|----------|----------------|----------------|----------------|----------------|
| DS1 | 0.98561 | 0.98561 | 0.98561 | 0.99281 | 0.98561 | 0.99511 |
| DS2 | 0.775 | 0.8 | 0.8 | 0.81 | 0.805 | 0.82 |
| DS3 | 0.86667 | 0.91667 | 0.9 | 0.91667 | 0.88333 | 0.93333 |
| DS4 | 0.91429 | 0.95714 | 0.97143 | 0.94286 | 0.98571 | 0.98571 |
| DS5 | 0.97674 | 1 | 1 | 1 | 1 | 1 |
| DS6 | 0.92683 | 1 | 0.97561 | 0.97561 | 0.95122 | 0.97561 |

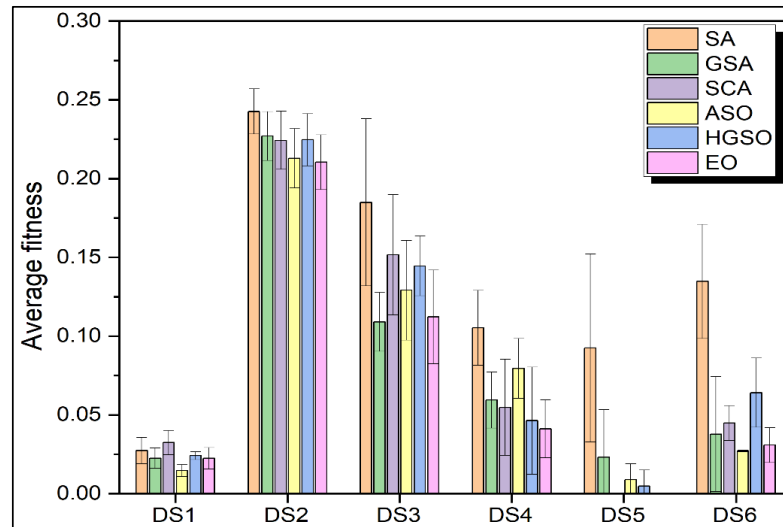


Figure 2. Average fitness and standard deviation of all algorithms on all selected datasets.

Finally, the average number of features selected by each algorithm (AFS) along with their standard deviations are presented in Table 5. As shown by the results, HGSO and SCA provided a minimum average number of features on two datasets each, whereas GSA and EO gave minimum AFS on one dataset each. However, if we also relate these results with the best accuracies, then the gain in terms of AFS of a few algorithms does not compensate for their low accuracies. For example, HGSO gave the minimum AFS on DS3, but its accuracy as compared to EO on DS3 was significantly low. Similarly, HGSO also gave the minimum AFS on DS6, but its accuracy was 2.5% lower than SCA, whereas the difference in AFS of both algorithms was just 0.2.

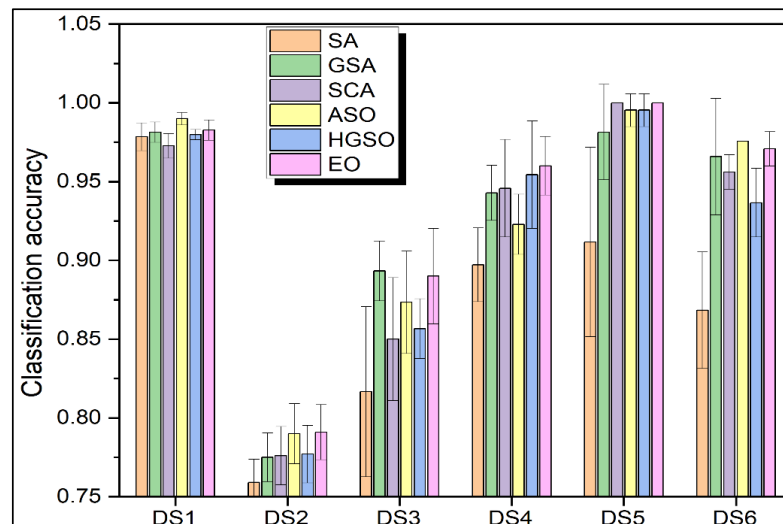


Figure 3. Average classification accuracy and standard deviation of all algorithms on all selected datasets.

Table 5. Mean selected feature subsets of all algorithms and their standard deviations.

| Dataset | | SA | GSA | SCA | ASO | HGSO | EO |
|---------|------|----------|----------------|----------------|-----------|------------|----------------|
| DS1 | AFS | 5.4 | 3.6 | 4.8 | 4.2 | 3.8 | 4.8 |
| | Std. | 2.07364 | 0.54772 | 0.44721 | 0.83666 | 0.83666 | 0.83666 |
| DS2 | AFS | 9.6 | 10.2 | 6 | 11.8 | 9.4 | 8.4 |
| | Std. | 1.81659 | 2.16795 | 1.41421 | 1.30384 | 3.91152 | 1.14018 |
| DS3 | AFS | 4.4 | 4.4 | 4 | 4.8 | 3.4 | 4.2 |
| | Std. | 1.67332 | 0.54772 | 1.41421 | 1.09545 | 0.89443 | 0.83666 |
| DS4 | AFS | 11.6 | 10 | 3.4 | 10.6 | 3.6 | 5 |
| | Std. | 4.92950 | 2.91548 | 0.89443 | 3.43511 | 1.14018 | 0.70711 |
| DS5 | AFS | 1986.6 | 1891.8 | 18 | 1682.6 | 77.2 | 3.4 |
| | Std. | 37.35371 | 73.07667 | 28.53945 | 107.37225 | 60.95654 | 0.54772 |
| DS6 | AFS | 25.8 | 23.8 | 8 | 17.8 | 7.8 | 11.4 |
| | Std. | 5.84808 | 4.14729 | 2.91548 | 3.11448 | 5.01996 | 4.15933 |

Amazingly, ASO, which gave some good results in terms of best accuracy, was unable to minimize dimensions as superbly as the other algorithms did. ASO frequently provided double the AFS as the best AFS provided by any other algorithm, as seen in DS2, DS4, DS5, and DS6.

3.3.3. Convergence Analysis

Convergence is the arrival of a stable point at which the solution stops improving any further. However, if an algorithm converges in very early iterations at a poor suboptimal point, then it is called premature convergence. In this section, we compare these algorithms based on how well they can converge, how well they can avoid premature convergence, and how quickly they can converge. The convergence curves of all these algorithms against each dataset are plotted in Figure 4. First of all, if we discuss the convergence capability, then all algorithms converged in less than the first half of the iterations for all datasets. If we talk about premature convergence, then SA converged prematurely in most cases; however, SCA on DS1, HGSO on DS3, and GSA on DS5 also converged prematurely. Finally, if we analyze their convergence speed, then EO demonstrated very good convergence speed on DS4, DS5, and DS6. In addition to that, ASO also demonstrated good speed on DS1 and DS3, and SCA was better than all the other algorithms on DS2. If we rank these algorithms based on the convergence capabilities, then EO secured the first rank, SCA was the second best, and SCA managed the third-best position.

The convergence speed of all algorithms is also measured in the average number of seconds. The average computational time along with standard deviations are presented in Figure 5. The results showed that SA took the least amount of time to converge on all datasets, which is obviously due to its premature convergence. Another reason may be that it is a single-solution algorithm. Similarly, GSA was the second-fastest algorithm on five out of six datasets. However, if we talk about the computational time taken by the best performers such as EO and ASO, then ASO took much less time than EO on all datasets except DS5.

3.3.4. Overall Performance Analysis

To find the overall best algorithm, we computed the ranks of all algorithms from three perspectives: average fitness, average accuracy, and average features selected (AFS). Once the ranks were determined, the average rank of each algorithm was computed on all datasets. The overall ranks and average rank of all algorithms on each dataset are presented in Table 6. As the results illustrate, the best rank was attained by EO, which was 1.88. However, the second-best position was shared by SCA and HGSO. It is important to mention that SA was the worst performer on the list.

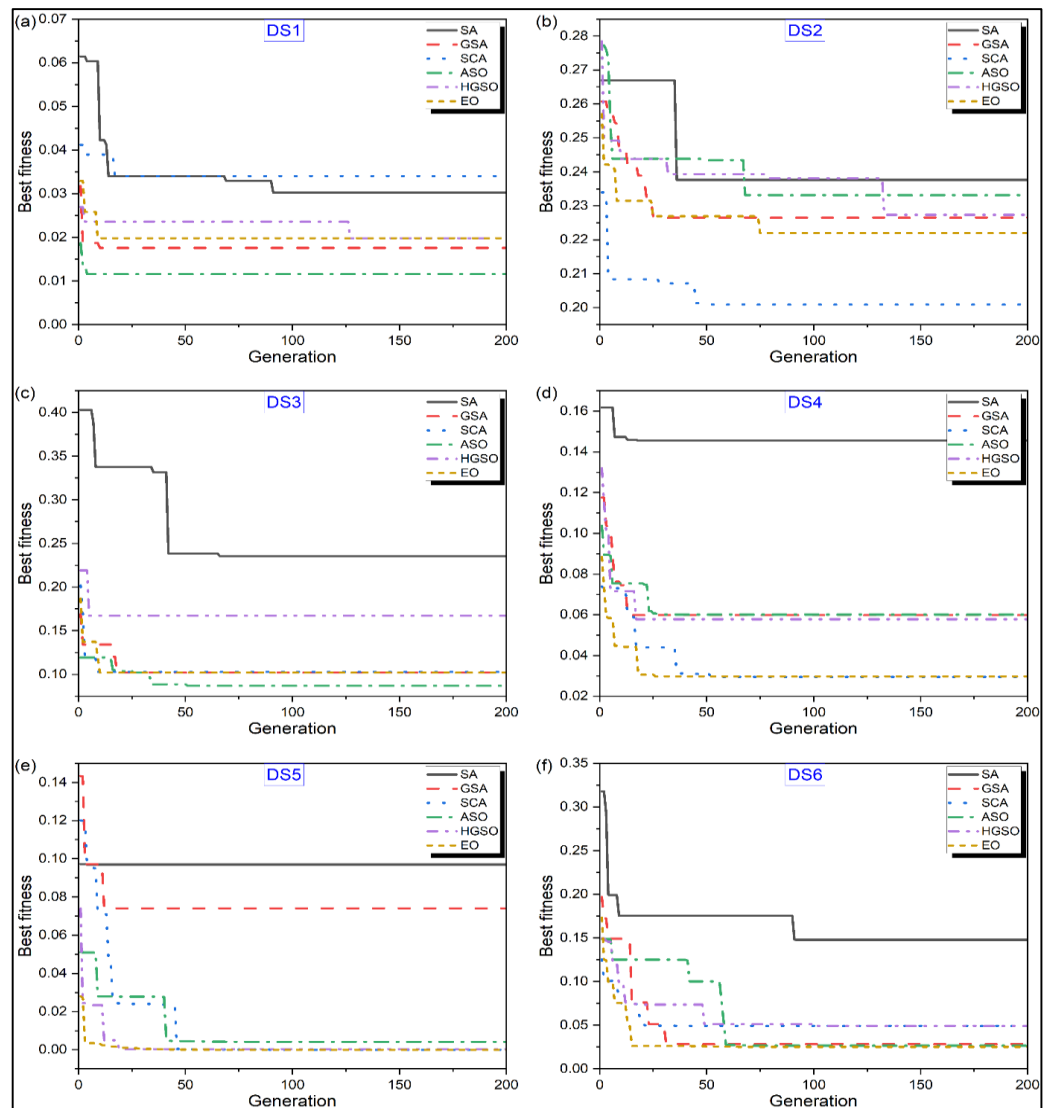


Figure 4. Convergence curves of all algorithms on all selected datasets i.e., (a) Breast cancer (DS1) dataset, (b) German (DS2) dataset, (c) Heart (DS3) dataset, (d) Ionosphere (DS4) dataset, (e) Ovarian cancer (DS5) dataset, (f) Sonar (DS6) dataset.

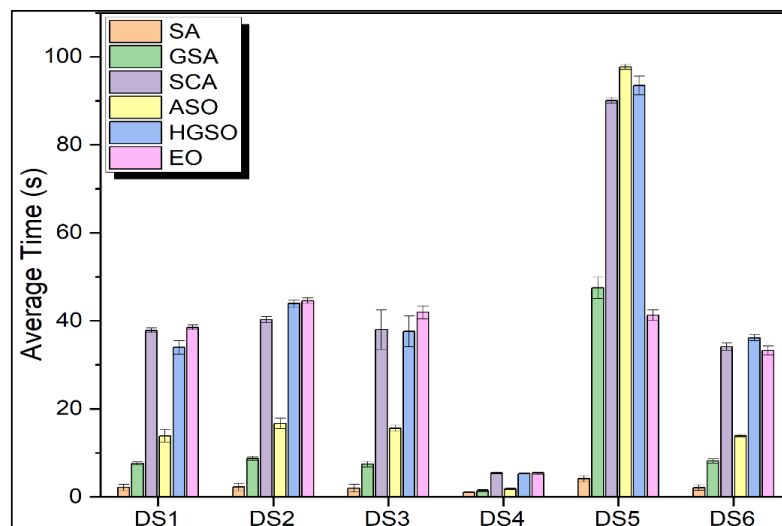


Figure 5. Average computational time and standard deviation of all algorithms on all selected datasets.

Table 6. Average rank of all averages (Fitness, accuracy, and AFS).

| Dataset | Stats | SA | GSA | SCA | ASO | HGSO | EO |
|-----------|--------------------|------|------|------|------|------|-------------|
| DS1 | Avg. fitness rank | 5 | 3 | 6 | 1 | 4 | 2 |
| | Avg. accuracy rank | 5 | 3 | 6 | 1 | 4 | 2 |
| | AFS rank | 6 | 1 | 4 | 3 | 2 | 4 |
| DS2 | Avg. fitness rank | 6 | 5 | 3 | 2 | 4 | 1 |
| | Avg. accuracy rank | 6 | 5 | 3 | 2 | 4 | 1 |
| | AFS rank | 4 | 5 | 1 | 6 | 3 | 2 |
| DS3 | Avg. fitness rank | 6 | 1 | 5 | 3 | 4 | 2 |
| | Avg. accuracy rank | 6 | 1 | 5 | 3 | 4 | 2 |
| | AFS rank | 4 | 4 | 2 | 6 | 1 | 3 |
| DS4 | Avg. fitness rank | 6 | 4 | 3 | 5 | 2 | 1 |
| | Avg. accuracy rank | 5 | 4 | 3 | 5 | 2 | 1 |
| | AFS rank | 6 | 4 | 1 | 5 | 2 | 3 |
| DS5 | Avg. fitness rank | 6 | 5 | 1 | 4 | 3 | 1 |
| | Avg. accuracy rank | 6 | 5 | 1 | 4 | 3 | 1 |
| | AFS rank | 6 | 5 | 2 | 4 | 3 | 1 |
| DS6 | Avg. fitness rank | 6 | 3 | 4 | 1 | 5 | 2 |
| | Avg. accuracy rank | 6 | 3 | 4 | 1 | 5 | 2 |
| | AFS rank | 6 | 5 | 2 | 4 | 1 | 3 |
| Avg. Rank | | 5.61 | 3.66 | 3.11 | 3.33 | 3.11 | 1.88 |

3.4. Comparison with Other Methods from the Literature

In this section, the top three physics-inspired metaheuristic algorithms are compared with the state of the art. For this comparison, results from other KNN-metaheuristic combinations reported by Elminaam et al. [37] were chosen. The metaheuristics chosen for comparison draw their metaphor inspiration from various sources. For example, the Grey Wolf Optimizer (GWO) and Whale Optimization Algorithm (WOA) may be classified as mammal inspired, whereas moth Flame Optimization (MFO) and the Butterfly Optimization Algorithm (BFO) are insect inspired. Similarly, Harris Hawk Optimization (HHO) and the Marine Predator Algorithm (MPA) are inspired by preying behavior seen in nature. Additionally, results based on popular ML algorithms such as Naive Bayes, Logistic Regression, Random Forest, Support Vector Machine (SVM), K-NN, Decision Tree, and Stochastic Gradient Descent (SGD) are also compared, along with their principal component analysis (PCA)-enhanced versions.

In terms of classification accuracy (Table 7), in two out of the three datasets compared, EO outperformed all the other methods. In fact, for the breast cancer dataset and ionosphere dataset, EO was on average 12.75% and 7.12% better, respectively, than the metaheuristics presented in [37]. In the sonar dataset, too, EO and SCA were within 2.5% of the best solution reported in [37]. Additionally, when compared with the ML algorithms, the EO solution for the breast cancer dataset was on average 17.89% better. An average superiority of 5.68% was seen for EO when compared with the PCA-ML methods reported in [38].

The average features selected for the breast cancer, ionosphere, and sonar datasets by the various metaheuristics are reported in Table 8. It can be observed that the feature reductions by the current physics-inspired metaheuristics are much higher. For the breast cancer, ionosphere, and sonar datasets, the average percent feature reduction achieved by the three physics-inspired algorithms was 85.11%, 88.24%, and 84.89%, respectively, and for the metaheuristic algorithms from [37], it was only 67.62%, 64.71%, and 67.14%, respectively.

Thus, from the comprehensive comparisons shown so far, it is clear that the current KNN hybridized physics-inspired metaheuristic algorithms (especially EO, SCA, and HGSO) are superior to those reported in the literature. Moreover, it is seen that even solutions by hybridized ML algorithms (for example, by dimensionality reduction techniques such as PCA) were inferior to current solutions. This is worth highlighting, since the current wrapper methods are much simpler in terms of computational complexity as compared to the PCA-hybridized ML methods.

Table 7. Comparison of classification accuracy with literature results.

| Method | Breast Cancer | % Improvement & | Ionosphere | % Improvement | Sonar | % Improvement |
|------------------------------|---------------|-----------------|------------|---------------|-------|---------------|
| EO | 0.995 | Best Solution | 0.986 | Best Solution | 0.976 | 2.46% |
| SCA | 0.986 | 0.91% | 0.971 | 1.54% | 0.976 | 2.46% |
| HGSO | 0.986 | 0.91% | 0.986 | Best Solution | 0.951 | 5.15% |
| GWO [37] | 0.970 | 2.58% | 0.951 | 3.68% | 0.970 | 3.09% |
| MFO [37] | 0.605 | 64.46% | 0.774 | 27.39% | 0.547 | 82.82% |
| WOA [37] | 0.973 | 2.26% | 0.957 | 3.03% | 0.976 | 2.46% |
| SSA [37] | 0.982 | 1.32% | 0.985 | 0.10% | 1.000 | Best Solution |
| BOA [37] | 0.903 | 10.19% | 0.901 | 9.43% | 0.881 | 13.51% |
| HHO [37] | 0.929 | 7.10% | 0.929 | 6.14% | 0.833 | 20.05% |
| MPA [37] | 0.982 | 1.32% | 0.985 | 0.10% | 0.976 | 2.46% |
| Naive Bayes [38] | 0.845 | 17.75% | - | - | - | - |
| Logistic Regression [38] | 0.879 | 13.20% | - | - | - | - |
| Random Forest [38] | 0.995 | Best Solution | - | - | - | - |
| SVM [38] | 0.620 | 60.48% | - | - | - | - |
| K-NN [38] | 0.900 | 10.56% | - | - | - | - |
| Decision Tree [38] | 0.880 | 13.07% | - | - | - | - |
| SGD [38] | 0.903 | 10.19% | - | - | - | - |
| PCA-Naive Bayes [38] | 0.975 | 2.05% | - | - | - | - |
| PCA-Logistic Regression [38] | 0.975 | 2.05% | - | - | - | - |
| PCA-Random Forest [38] | 0.962 | 3.43% | - | - | - | - |
| PCA-SVM [38] | 0.942 | 5.63% | - | - | - | - |
| PCA-K-NN [38] | 0.921 | 8.03% | - | - | - | - |
| PCA-Decision Tree [38] | 0.905 | 9.94% | - | - | - | - |
| PCA-SGD [38] | 0.916 | 8.62% | - | - | - | - |

& % improvement achieved by the best solution with respect to the compared algorithms.

Table 8. Comparison of AFS with literature results.

| Method | Breast Cancer | % Feature Reduction & | Ionosphere | % Feature Reduction | Sonar | % Feature Reduction |
|----------|---------------|-----------------------|------------|---------------------|-------|---------------------|
| EO | 4.8 | 84% | 5 | 85% | 11.4 | 81% |
| SCA | 4.8 | 84% | 3.4 | 90% | 8 | 87% |
| HGSO | 3.8 | 87% | 3.6 | 89% | 7.8 | 87% |
| GWO [37] | 7 | 77% | 4 | 88% | 11 | 82% |
| MFO [37] | 6 | 80% | 23 | 32% | 31 | 48% |
| WOA [37] | 8 | 73% | 7 | 79% | 26 | 57% |
| SSA [37] | 11 | 63% | 14 | 59% | 16 | 73% |
| BOA [37] | 12 | 60% | 20 | 41% | 26 | 57% |
| HHO [37] | 12 | 60% | 10 | 71% | 20 | 67% |
| MPA [37] | 12 | 60% | 6 | 82% | 8 | 87% |

& % Feature reduction is calculated as 100% minus the ratio of AFS by each algorithm and maximum features in the corresponding dataset. A higher value of % feature reduction is desired.

4. Conclusions

In this paper, six well-cited physics-inspired metaphor algorithms were employed for feature selection. Feature selection is one of the major challenges being faced in the field of data mining and machine learning. The objective of this research was to identify the most promising physics-inspired algorithms for the problem of feature selection. To accomplish this, six small- to large-sized datasets were used. The performance of EO was found to be superior on most of the datasets, and the metrics that were used for the comparative analysis were taken from the literature and included accuracy, fitness, the average number of features selected, and convergence analysis. The current physics-inspired metaphor algorithms, especially EO, SCA, and HGSO comprehensively outperformed other metaheuristics, as well as the ML-based solutions seen in recent literature. Based on our findings, we highly recommend using EO for the feature selection problem.

Author Contributions: Conceptualization, R.Č. and K.K.; Data curation, J.P., M.P. and M.J.; Formal analysis, J.P., M.P. and M.J.; Investigation, J.P., M.P. and M.J.; Methodology, R.Č. and K.K.; Software, R.Č. and K.K.; Validation, J.P., M.P. and M.J.; Visualization, J.P., M.P. and M.J.; Writing—original draft, J.P., M.P. and M.J.; Writing—review and editing, R.Č. and K.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available through email upon request to the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Köppen, M. The curse of dimensionality. In Proceedings of the 5th Online World Conference on Soft Computing in Industrial Applications (WSC5), Online, 4–18 September 2000; Volume 1, pp. 4–8.
2. Ikotun, A.M.; Almutari, M.S.; Ezugwu, A.E. K-Means-Based Nature-Inspired Metaheuristic Algorithms for Automatic Data Clustering Problems: Recent Advances and Future Directions. *Appl. Sci.* **2021**, *11*, 11246. [\[CrossRef\]](#)
3. Khalid, S.; Khalil, T.; Nasreen, S. A survey of feature selection and feature extraction techniques in machine learning. In Proceedings of the Science and Information Conference (SAI), London, UK, 27–29 August 2014; pp. 372–378.
4. Porkodi, R. Comparison of filter based feature selection algorithms: An overview. *Int. J. Innov. Res. Technol. Sci.* **2014**, *2*, 108–113.
5. Jović, A.; Brkić, K.; Bogunović, N. A review of feature selection methods with applications. In Proceedings of the 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 25–29 May 2015; pp. 1200–1205.
6. Brezočnik, L.; Fister, I.; Podgorelec, V. Swarm Intelligence Algorithms for Feature Selection: A Review. *Appl. Sci.* **2018**, *8*, 1521. [\[CrossRef\]](#)
7. Askari, Q.; Saeed, M.; Younas, I. Heap-based optimizer inspired by corporate rank hierarchy for global optimization. *Expert Syst. Appl.* **2020**, *161*, 113702. [\[CrossRef\]](#)
8. Rahman, A.; Sokkalingam, R.; Othman, M.; Biswas, K.; Abdullah, L.; Kadir, E.A. Nature-Inspired Metaheuristic Techniques for Combinatorial Optimization Problems: Overview and Recent Advances. *Mathematics* **2021**, *9*, 2633. [\[CrossRef\]](#)
9. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [\[CrossRef\]](#)
10. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [\[CrossRef\]](#)
11. Passino, K.M. Bacterial foraging optimization. *Int. J. Swarm Intell. Res. (IJSIR)* **2010**, *1*, 1–16. [\[CrossRef\]](#)
12. Whitley, D. A genetic algorithm tutorial. *Stat. Comput.* **1994**, *4*, 65–85. [\[CrossRef\]](#)
13. Storn, R. On the usage of differential evolution for function optimization. In Proceedings of the North American Fuzzy Information Processing, Berkeley, CA, USA, 19–22 June 1996; pp. 519–523.
14. Simon, D. Biogeography-based optimization. *IEEE Trans. Evol. Comput.* **2008**, *12*, 702–713. [\[CrossRef\]](#)
15. Askari, Q.; Younas, I.; Saeed, M. Political Optimizer: A novel socio-inspired meta-heuristic for global optimization. *Knowl.-Based Syst.* **2020**, *195*, 105709. [\[CrossRef\]](#)
16. Fadakar, E.; Ebrahimi, M. A new metaheuristic football game inspired algorithm. In Proceedings of the 2016 1st Conference on Swarm Intelligence and Evolutionary Computation (CSIEC), Bam, Iran, 9–11 March 2016; pp. 6–11.
17. Rashedi, E.; Nezamabadi-Pour, H.; Saryazdi, S. GSA: A Gravitational Search Algorithm. *Inf. Sci.* **2009**, *179*, 2232–2248. [\[CrossRef\]](#)
18. Hatamlou, A. Black hole: A new heuristic optimization approach for data clustering. *Inf. Sci.* **2013**, *222*, 175–184. [\[CrossRef\]](#)
19. Zerigat, D.H.; Benasla, L.; Belmadani, A.; Rahli, M. Galaxy-based search algorithm to solve combined economic and emission dispatch. *UPB Sci. Bull. Ser. C Electr. Eng.* **2014**, *76*, 209–220.
20. Abualigah, L.M.; Khader, A.T.; Hanandeh, E.S. A new feature selection method to improve the document clustering using particle swarm optimization algorithm. *J. Comput. Sci.* **2018**, *25*, 456–466. [\[CrossRef\]](#)
21. Zakeri, A.; Hokmabadi, A. Efficient feature selection method using real-valued grasshopper optimization algorithm. *Expert Syst. Appl.* **2019**, *119*, 61–72. [\[CrossRef\]](#)
22. Mafarja, M.M.; Mirjalili, S. Hybrid Whale Optimization Algorithm with simulated annealing for feature selection. *Neurocomputing* **2017**, *260*, 302–312. [\[CrossRef\]](#)
23. Vijayanand, R.; Devaraj, D. A Novel Feature Selection Method Using Whale Optimization Algorithm and Genetic Operators for Intrusion Detection System in Wireless Mesh Network. *IEEE Access* **2020**, *8*, 56847–56854. [\[CrossRef\]](#)
24. Kelidari, M.; Hamidzadeh, J. Feature selection by using chaotic cuckoo optimization algorithm with levy flight, opposition-based learning and disruption operator. *Soft Comput.* **2021**, *25*, 2911–2933. [\[CrossRef\]](#)
25. Zawbaa, H.M.; Emary, E.; Parv, B.; Sharawi, M. Feature selection approach based on moth-flame optimization algorithm. In Proceedings of the 2016 IEEE Congress on Evolutionary Computation (CEC), Vancouver, BC, Canada, 24–29 July 2016; pp. 4612–4617.
26. Selvakumar, B.; Muneeswaran, K. Firefly algorithm based feature selection for network intrusion detection. *Comput. Secur.* **2019**, *81*, 148–155.
27. Abdel-Basset, M.; Ding, W.; El-Shahat, D. A hybrid Harris Hawks optimization algorithm with simulated annealing for feature selection. *Artif. Intell. Rev.* **2021**, *54*, 593–637. [\[CrossRef\]](#)
28. Wolpert, D.H.; Macready, W.G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [\[CrossRef\]](#)

29. Too, J.; Liang, G.; Chen, H. Memory-based Harris hawk optimization with learning agents: A feature selection approach. *Eng. Comput.* **2021**, *38*, 4457–4478. [[CrossRef](#)]
30. Bertsimas, D.; Tsitsiklis, J. Simulated annealing. *Stat. Sci.* **1993**, *8*, 10–15. [[CrossRef](#)]
31. Mirjalili, S. SCA: A Sine Cosine Algorithm for solving optimization problems. *Knowl.-Based Syst.* **2016**, *96*, 120–133. [[CrossRef](#)]
32. Zhao, W.; Wang, L.; Zhang, Z. Atom search optimization and its application to solve a hydrogeologic parameter estimation problem. *Knowl.-Based Syst.* **2019**, *163*, 283–304. [[CrossRef](#)]
33. Hashim, F.A.; Houssein, E.H.; Mabrouk, M.S.; Al-Atabany, W.; Mirjalili, S. Henry gas solubility optimization: A novel physics-based algorithm. *Future Gener. Comput. Syst.* **2019**, *101*, 646–667. [[CrossRef](#)]
34. Faramarzi, A.; Heidarinejad, M.; Stephens, B.; Mirjalili, S. Equilibrium optimizer: A novel optimization algorithm. *Knowl.-Based Syst.* **2020**, *191*, 105190. [[CrossRef](#)]
35. Conrads, T.P.; Fusaro, V.A.; Ross, S.; Johann, D.; Rajapakse, V.; Hitt, B.A.; Steinberg, S.M.; Kohn, E.C.; Fishman, D.A.; Whitely, G.; et al. High-resolution serum proteomic features for ovarian cancer detection. *Endocr.-Relat. Cancer* **2004**, *11*, 163–178. [[CrossRef](#)] [[PubMed](#)]
36. Street, W.N.; Wolberg, W.H.; Mangasarian, O.L. Nuclear feature extraction for breast tumor diagnosis. In Proceedings of the IS&T/SPIE 1993 International Symposium on Electronic Imaging: Science and Technology, San Jose, CA, USA, 31 January–5 February 1993; Volume 1905, pp. 861–870.
37. Elminaam, D.S.A.; Nabil, A.; Ibraheem, S.A.; Houssein, E.H. An Efficient Marine Predators Algorithm for Feature Selection. *IEEE Access* **2021**, *9*, 60136–60153. [[CrossRef](#)]
38. Ibrahim, S.; Nazir, S.; Velastin, S.A. Feature Selection Using Correlation Analysis and Principal Component Analysis for Accurate Breast Cancer Diagnosis. *J. Imaging* **2021**, *7*, 225. [[CrossRef](#)] [[PubMed](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.