JAIEP

iHow Optimization Algorithm: A Human-Inspired Metaheuristic Approach for Complex Problem Solving and Feature Selection

El-Sayed M. El-Kenawy [1,2,3,4] *, **Faris H. Rizk**[5], **Ahmed Mohamed Zaki**[5], **Mahmoud Elshabrawy Mohamed**[5], **Abdelhameed Ibrahim**[1], **Abdelaziz A. Abdelhamid**[6,7], **Nima Khodadadi**[8], **Ehab M. Almetwally**[9,10], **Marwa M. Eid**[11]

[1]School of ICT, Faculty of Engineering, Design and Information & Communications Technology (EDICT), Bahrain Polytechnic, PO Box 33349, Isa Town, Bahrain.
[2]Jadara University Research Center, Jadara University, Jordan.
[3]Applied Science Research Center. Applied Science Private University, Amman, Jordan.
[4]Department of Communications and Electronics, Delta Higher Institute of Engineering and Technology, Mansoura 35111, Egypt.
[5]Computer Science and Intelligent Systems Research Center, Blacksburg 24060, Virginia, USA
[6]Department of Computer Science, Faculty of Computer and Information Sciences, Ain Shams University, Cairo 11566, Egypt
[7]Department of Computer Science, College of Computing and Information Technology, Shaqra University, 11961, Shaqra, Saudi Arabia
[8]Department of Civil and Architectural Engineering, University of Miami, Coral Gables, FL, USA
[9]Department of Mathematics and Statistics, Faculty of Science, Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh 11432, Saudi Arabia
[10]Faculty of Business Administration, Delta University for Science and Technology, Gamasa 11152, Egypt
[11]Faculty of Artificial Intelligence, Delta University for Science and Technology, Mansoura 11152, Egypt

*Corresponding author: skenawy@ieee.org
Emails: skenawy@ieee.org, faris.rizk@jcsis.org, Azaki@jcsis.org, mshabrawy@jcsis.org, abdelhameed.fawzy@polytechnic.bh, abdelaziz@cis.asu.edu.eg, nima.khodadadi@miami.edu, EMAlmetwally@imamu.edu.sa, mmm@ieee.org

Abstract

In this paper, we propose the iHow Optimization Algorithm (iHowOA), a novel metaheuristic algorithm inspired by human-like cognitive processes such as learning, knowledge acquisition, and experience-based decision-making. The iHowOA aims to enhance the exploration-exploitation balance inherent to solving complex optimization problems by mimicking how humans gather data, learn, and improve over time. We tested the algorithm on standard benchmark functions, including those from the CEC 2005 suite, to evaluate its performance in terms of convergence, computational efficiency, and solution accuracy. Furthermore, the Binary iHowOA (biHowOA) was employed for feature selection tasks, and its performance was compared with other popular optimization algorithms. The results show that iHowOA achieves superior performance, consistently finding optimal solutions while maintaining computational efficiency. The biHowOA also demonstrated strong capability in feature selection, providing reduced feature sets with minimal classification error. Our experiments confirm that iHowOA offers an effective solution for both continuous optimization and feature selection challenges.

Keywords: iHow Optimization Algorithm, Metaheuristic, Feature Selection, Exploration-Exploitation, Computational Efficiency
MSC: 68T20; 62J05; 93B45

Doi: https://doi.org/10.21608/jaiep.2024.386694
Received: March 20, 2024   Revised: August 12, 2024   Accepted: October 6, 2024

1 introduction

Metaheuristic optimization algorithms have been found to be useful tools for solving optimization problems and especially when humanitarian optimization methods cannot be used. Most of these algorithms have been used in several fields including engineering, artificial intelligence, economics, and logistics since optimization problems cut across all professions and concerns search spaces that may often be nonlinear, multimodal, and high-dimensional. Unlike classical approaches, metaheuristics can accommodate any feasible solution and are therefore effective in the search of large problem solutions spaces [1–3].

At the heart of any optimization problem is the need to balance two critical objectives: exploration and exploitation. The discovery or navigation and search part entails the algorithm identify the areas of solution space with optimum or near optimum solutions. This makes certain that the algorithm does not get stuck in local optimum [4–6]. Exploitation, on the other hand, operates by iterating on regions that have been deemed fruitful already and provide the algorithm with greater precision for its solutions. In the optimization algorithms, there is always a fine line between the two objectives and the best optimization algorithm must achieve the right blend of the two in order to serve the purpose efficiently. Excessive space investigation has potential for being detrimental to the optimization process as the algorithm drifts in the designated space instead of optimizing the solution space. On the other hand, the use of exploitation may lead to the convergence to false optima closer to some part of the solution space than other better solutions exist [7–9].

It is a fact that several metaheuristic algorithms are available to overcome these challenges for example Particle Swarm Optimization (PSO), Genetic Algorithm (GA) and Grey Wolf Optimizer (GWO). Some of these algorithms are inspired by nature or biological systems, while all of them tend to have different strategies in turning between exploitation and exploration. For example, PSO simulates flocking ofbirds or fishes; every particle in the swarm moves towards the area of the best solutions seen by the particle and the swarm. GA thus mimics the natural selection process of a population whereby candidate solutions for complex problems are allowed to evolve over several generations and their progeny is expected to possess better characteristics than the parent population. GWO imitates the hunting mechanism of the grey wolves, and the leadership of the GWO is established in the hunting hierarchy, coordinative search through scout wolves and the global optimization through the leader wolves [10–12].

Although there has been success demonstrated through the metaheuristic algorithms in solving many optimization problems, there are noticeable drawbacks especially when solving complex or dynamic problems. Common problems that must be solved are related to failures, including premature convergence, inability to search in large or multimodal spaces effectively, as well as high computational costs. Further, specific optimization problems need specific approaches and, therefore, there can be no single approach that can do well on all optimization problems. As a result, the research study towards the new optimization algorithm must be brought out on-going to overcome these shortcomings and enhance the epoch-making solution to untangle more challenging and diversified optimization problems [13–15].

The present paper proposes a new metaheuristic optimization algorithm called iHow Optimization Algorithm (iHowOA), which is intended to minimize some of the problems identified in previous algorithms. In particular, iHowOA empowers the traditional exploration-exploitation design by aiming to provide a more efficient solution for searching for optimal solutions and overcoming subtle and not-so-subtle challenges in the optimization landscape. Moreover, this new approach is equipped with means by which the search can adapt itself and its processes to better accommodate the nature of particular optimization problems.

2 Literature Review

Metaheuristic algorithms have become essential tools for solving complex optimization problems, offering flexibility in navigating large, nonlinear solution spaces where classical methods fall short. Popular algorithms like Particle Swarm Optimization (PSO), Genetic Algorithms (GA), and Grey Wolf Optimizer (GWO) have achieved success but often struggle with issues such as premature convergence and high computational costs. This review highlights key advancements in metaheuristics while identifying areas where existing algorithms face challenges, setting the stage for the development of the iHow Optimization Algorithm (iHowOA).

The paper [16] presents the Nutcracker Optimization Algorithm (NOA), which is a newly developed metaheuristic algorithm derived from the specific dynamism of Clark's nutcrackers. Inspired by the two behaviors of these birds different seasons This is an algorithm that mimics these two behavior patterns of the birds during different seasons. These behaviors are imitated by the NOA algorithm to build a flexible metaheuristic optimization method with varied local and global search functions. All in all, this concept helps NOA facilitate solid solutions for numerous optimisation challenges and results in a better result. To assess NOA's efficiency,

https://doi.org/10.21608/jaiep.2024.354003
Received: March 20, 2024 Revised: August 12, 2024 Accepted: October 6, 2024

it was tested on the twenty-three function standard test bed as well as the CEC test suites of 2014, 2017, and 2020, in addition to five authentic engineering design problems.

NOA was compared against three categories of existing optimization algorithms: (1) relatively new algorithms which were published over the last five years, and they include Slime mold algorithm (SMA), Gradient based optimizer (GBO), Equilibrium optimizer (EO), Runge-Kutta method (RUN), African vultures optimization algorithm (AVOA), Rosenbrock filter optimization (RFO), and Artificial gorilla troops optimizer (GTO); (2) popular algorithms The results proved that proposed NOA outperformed all the compared methods, even the state-of-art optimizers such as; LSHADE-cnEpSin and LSHADE-SPACMA (The winner of CEC-2017) and AL-SHADE and L-SHADE (The winner of CEC-2014). According to this it can be identified that NOA is a suitable optimization algorithm which can provide optimum results for a wide range of problems in optimal searching.

Hence, in [17], based on the research direction proposed to continuously engage the researchers in the nature-inspired algorithms and optimization problems, a detailed analysis of the GSO algorithm along with its detailed discussions have been included. GSO is a stochastic optimization algorithm which was proposed to solve several different optimization problems by He et al. (2009). It mimics the searching behaviour exhibited by animals in real life. This survey concentrates on the uses of the GSO algorithm and its derivatives and works which are published after the year of suggestion of the algorithm, 2009 to 2020. GSO is used in solving any optimization problem and identify the minimum or maximum for the given objective function among the solutions set by using GSO algorithm with some candidates solutions. Taking to account function approximations, obtaining the global optimum when using meta-heuristic optimizations or nature-inspired algorithms are an interesting area due to their rule in solving various decision-making problems. The various procedures of the GSO as well as the basic algorithm version are outlined and followed by the discrete versions and the modified GSO. Similarly, the real-life application of GSO is provided with examples like benchmark function, classification function, networking, engineering, and other worldwide problem statement. Lastly, based on the papers reviewed in the literature involving all the publishers such as IEEE, Elsevier, and Springer, GSO algorithm is widely utilised in solving different optimisation problems. Besides that, it achieved comparative and encouraging results than any other similar published optimization algorithm.

Optimization is a keyword for all research scholars, as often as they consider the engineering issues. [18] introduces a novel metaheuristic called dingo optimizer (DOX) which is inspired from dingo an extant canid species known as Canis familiaris dingo. The general idea is to bring into life this process of doing this method using the elements of interaction and sociality of dingoes. The developed algorithm is based on the hunting process of dingoes which are exploration, encircling and exploitation. The above mentioned steps in prey hunting are mathematically formulated and incorporated into the simulator for evaluating the efficiency of the developed algorithm. Comparisons are made between the proposed approach and other metaheuristic techniques such as grey wolf optimizer (GWO), and particle swarm optimizer (PSO). As for the comparative analysis of this work, some of the widely known test functions are employed. The comparisons of the performance results show that dingo optimizer yielded better results than other proposed nature-inspired algorithms.

In [19] the new stochastic optimization algorithm named Driving Training-Based Optimization (DTBO) which imitate the human activity of driving training. The idea which is viewed as the cornerstone for DTBO design is the learning experience to drive in the driving school and the further training of the driving instructor. DTBO is mathematically modeled in three phases: They include: (1) Training by the driving instructor (2) patterning of students from the instructor skills and (3) practice. The performance of DTBO in optimization is measured with a set of 53 standard objective functions of unimodal, high-dimensional multimodal, fixed-dimensional multimodal, and IEEE CEC2017 test functions type. The analysis of optimization results indicates that DTBO has been able to solve optimization problems and arrive at the right solution by balancing between exploration aspect and exploitation aspect. DTBO is compared with the performance quality of 11 other popular algorithms to check the validity of the results obtained by the algorithm. Comparing the simulation results, it is possible to state that DTBO mans a better result compared with 11 competitive algorithms and is more effective in the optimization tasks.

Sliding is one of the geological risks that are destructive to the lives of people in high grounds or slopes of the mountains. Landslide susceptibility map is therefore used for assessment of landslides risks that are vital for the prediction and prevention of such disasters. The purpose of [20] was to examine the performances of the metaheuristic optimized deep learning algorithms using convolutional neural networks besides grey wolf optimizer and imperialist competitive algorithm to produce a landslide susceptibility map. Landslide susceptibility map had to be prepared for the study area which was Icheon City in South Korea for which detailed landslide inventory dataset is available. The landslide inventory map was prepared and data set was split into a training data set comprising 70% and the validation data set of 30%. In the same way, the study used specific 18 factors which relate to landslides such as the geo-environmental and topo-hydrological factors to the variables. The models were compared using values of area under curve (AUC) in receiver operating characteristic (ROC) curve analysis. The results of the validation also indicated that the optimized models with the two proposed algorithms, CNN-GWO and CNN-ICA, achieved higher AUC and lower RMSE compared to the standalone CNN model. However, the CNN model achieved better results as compared to previous studies that have employed solely machine learning algorithm. Therefore, the application of the deep learning algorithm with optimization algorithms described in this study can comprise more appropriate models for the landslide susceptibility mapping in the study area since it has a higher level of accuracy.

In recent times, the field of numerical optimization has been extended and received the interesting of the research community to design and present many metaheuristic optimization algorithms. [21] proposes a novel metaheuristic optimization algorithm denoted by Honey Badger Algorithm (HBA). To logically design an efficient search strategy for optimizing the problems, the proposed algorithm is derived from the hone y badger intelligent foraging behavior. Honey badger's digging and honey finding search dynamics are translated to exploration and exploitation in HBA by the formulation into different phases. Furthermore, by using controlled randomization approach the composition of the selected population remains diverse right to the last assessed layer in HBA. In order to evaluate the performance of HBA, four engineering design problems, 24 standard benchmark functions, and CEC'17 test suite are implemented. The results produced by the proposed HBA have been compared with ten typical metaheuristic algorithms such as SA, PSO, CMA-ES, L-SHADE, MFO, EHO, WOA, GOA, TEO, and HHO. Analyzing the results of experiments as well as statistical data, the efficiency of HBA for solving optimization problems with the large search-space is shown, as well as the advantage of HBA in terms of convergence rate and exploration–exploitation trade-off in comparison with the other methods that were used in this research.

The real world numerical optimization problems has become much more difficult and complex in their nature which requires efficient optimization methods. So far, diverse metaheuristic approaches have been proposed although only several of them are acknowledged in the research society. To solve the optimization problems, [22] presents a new metaheuristic algorithm known as Archimedes optimization algorithm (AOA). AOA is developed bearing in mind an interesting principle in physics known as Archimedes' Principle. It mimics the theory of buoyant force acting in upward direction on an object that is either partly or fully submerged in a fluid and is directly proportional to the weight of the quantity of the fluid displaced. To measure performance of the proposed AOA algorithm, CEC'17 test suite and four engineering design problems are used. The solutions derived by employing AOA have surpassed robust established benchmark metaheuristic and recently introduced algorithms like genetic algorithms (GA), particle swarm optimization (PSO), differential evolution variants L-SHADE and LSHADE-EpSin, whale optimization algorithm (WOA), sine-cosine algorithm (SCA), Harris hawk optimization (HHO,) and equilibrium optimizer (EO). According to the results of the experiments, AOA is a promising optimization tool with concerns to the convergence rate and the AA/EA trade-off because it can be used in solving real-life problems.

The development of the councils is from the lowest level, which is the neighborhood level and right up to the regions and the entire city. Members of a council endeavor to perform well in order to be elected as the boss of the particular council in the next election in addition to being a representative of the large council. That is why they coaxed them to put forward the novel socio-inspired metaheuristic optimization algorithm [23], termed City Councils Evolution (CCE), based on the evolution of city councils. To evaluate CCE, the said algorithm is used to solve 20 problems of general test functions and 29 test functions provided by CEC 2017. Results of CCE are compared with the effectiveness of nine popular and new optimization algorithms belonging to different classes: SHADE and its variant LSHADE-cnEpSin were reported for high performance and as winning competitors in IEEE CEC challenges 2013 and 2017; EO, BWO, PO, BMO, CHOA, AO, and WHO are recent additions to the literature and based on work done in 2020 and 2021. For all 49 test functions, CCE is superior to EO, BWO, PO, BMO, CHOA, AO, and WHO by 65%, 95%, 64%, 68%, 80%, 74%, and 71%, respectively Its performance is inferior to SHADE and LSHADE-cnEpSin by 49% and 65% respectively. Last but not the least, the comparative numerical solution of some practical constrained optimization problems using the suggested algorithm demonstrates better effectiveness of the advanced algorithm in comparison with some good algorithms of the existing techniques.

OPF is one of the challenges of power system operation that involves multi-modality, large scale, non-convex and non-linear constrained optimization problems. For these reasons, solving OPF problem is gradually becoming the focus of power engineers, and the researchers' activity. In [24], some recent metaheuristic algorithms namely; Grasshopper Optimization Algorithm (GOA), Black Widow Optimization Algorithm, Grey Wolves Optimizer, Ant Lion Optimizer, Particles Swarm Optimization, Gravitational Search Algorithm, Moth-Flame Optimization, and Barnacles Mating Optimizer (BMO) shall be employed to solve three objective functions of OPF problem which are as follows; The cost of thermal, stochastic wind and solar To evaluate the performance of these selected metaheuristic algorithm on OPF, a modified IEEE 30-bus system will be utilized

https://doi.org/10.21608/jaiep.2024.354003
Received: March 20, 2024 Revised: August 12, 2024 Accepted: October 6, 2024

which will incorporate stochastic wind and solar power generators. Probabilistic analyses are conducted to assess possible results of the algorithms in question. Performance analysis shows that BMO achieved slightly higher results compared to the other algorithms and show that it can be used as an efficient solution to the OPF problem.
[25] proposes a novel, Subtraction-Average-Based Optimizer (SABO) method based on the use of evolution in solving optimization problems. The rationale of the proposed SABO is based on the simple idea of using the subtraction of the average of searcher agents to refine the position of population members in the search space. The particular stages of the SABO's deployment are outlined, and then mathematical representations for various optimization tasks are provided. The efficiency of the proposed SABO approach is examined for the optimization of 52 standard benchmark functions, including unimodal, high-dimensional multimodal, and fixed-dimensional multimodal functions, as well as the CEC 2017 test suite. The results of the optimization also justify that the SABO approach can solve the optimization problems without sacrificing the balance of exploration and exploitation in the search space for problem solutions. The observed SABO results are, in turn, compared with those of twelve benchmark metaheuristic algorithms as follows: From the analysis of the simulation results it is observed that the proposed SABO approach performs better than other approaches on most of the benchmark functions. In addition, it yields a much more efficient and exceptional performance than the competitor algorithms. Further, the procedure of the SABO is demonstrated for four engineering design problems and thus it is tested on real-world optimization scenarios. Analysis of the optimization results proved that the SABO approach has the capability to model real-world scenarios and offers more better solutions than the competitor algorithms.
The review underscores the strengths and limitations of existing metaheuristic algorithms, particularly in balancing exploration and exploitation. While recent innovations address some challenges, gaps remain in efficiently solving complex optimization problems. The proposed iHowOA aims to improve upon these methods by integrating human-like learning processes, offering a novel solution that enhances performance and efficiency in optimization tasks.

3 Proposed iHow Optimization Algorithm (iHowOA)

3.1 iHowOA Inspiration

The iHow Optimization Algorithm (iHowOA) is inspired by human-like processes such as learning, knowledge acquisition, exploration, and experience-based decision-making. It follows the idea that, much like how humans start with basic information and gradually develop expertise through learning and the accumulation of knowledge, the algorithm mimics this process to optimize complex problems. The foundation of the algorithm begins with data gathering, where raw data is collected, serving as the elementary building blocks for solving a problem, similar to how humans begin with basic facts.
After this, moves to the learning phase of the algorithm where it processes and analyses the data in a way that is similar to how human becomes wise after going through the world. After learning, the algorithm shifts to the information processing phase where the data it has learned from is processed in a manner that humans process knowledge for practical use. This is followed by the knowledge acquisition where the algorithm gathers and preserves new information to improve its performance in the long run. This accumulated know-how is all important in determining how the process is to be carried forward.
Just as man learns through possible ways and alternatives, iHowOA tries out all possible strategies of the solution space. This exploration is made more efficient and objective-oriented through the use of the store knowledge and experiences. Moreover, the algorithm adapts knowledge gathered during its iterations by refining it and learning the parameters of newly dared approximate solutions. This erection of learning from previous versions is similar to how human beings get better in the subsequent decision making processes.
Like in real life, feedback is critical to growth, and that's why iHowOA avails feedback loops to boost its optimization processes. Algorithm further devises its tactics and answers improving future recommendations and solutions, with reference to previous feedback until a better performing strategy is reached. Finally as people master each of the given areas of specialization then the iHowOA gets to an optimum solution. By facilitating data collection, learning, knowledge aquisition, search and feedback, iHowOA process models the human manner, from entry level knowledge up to an expert level to tackle optimization problems.
As presented in Figure 1 the iHow Optimization Algorithm (iHow OA) is organized as a hierarchy of the following stages: The Data is at the base of this pyramid and that it is the raw inputs that are collected. The next layer is the "Learning & Asking" layer where the algorithm starts parsing raw data and start differentiating between them. Above this there is the "Information" layer, where data is processed and made useful in the construction of "Knowledge." The first layer is the "Knowledge" layer, which represents the compilation of the

structured information that leads to improved decisions making. On the very apex of the pyramid, there is the "Expert" layer describing the algorithm's capacity to apply developed knowledge and experience for efficient solutions' improvement. Every layer represents a higher level of information processing, with feedback-that feeds back into the algorithm and resulting in the progress through its steps.

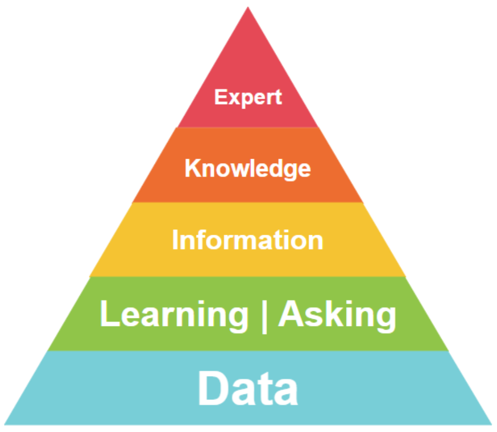

Figure 1: Hierarchical Knowledge Pyramid in iHow Optimization Algorithm

3.2 Experimental Setup

The choice of iHowOA is based on the various stages implemented in its experimental design which is essential for testing its performance against various benchmark functions. These stages include parameter initialization, the exploratory phase, the learning phase, the knowledge-acquisition phase and the exploitation phases. There is also feedback handling to make new adjustments to the solution proposed in the subsequent cycles. This section will describe each of them, as well as the mathematical equations deployed when guiding the behavior of the iHowOA throughout the experiments.

3.2.1 Constants and Parameters

In terms of search performances the iHowOA is regulated by a list of predefined constants and parameters. These parameters are necessary for the regulation of the algorithm exploration-exploitation in complex problems spaces which are fundamentally essential. The key parameters used in the experiments are defined as follows:

- $r_1 = 0.1$: Controls the weight of the first component in the exploration and learning phases.
- $r_2 = 0.1$: Modifies the contribution of the second component in the position update equations.
- $r_3 = 0.1$: Adjusts the influence of the third component during learning and knowledge acquisition.
- $r_4 = 0.2$: Scales the contribution of the fourth component during exploitation.
- $r_5 = 0.2$: Further adjusts the weight of the fifth component during solution refinement.

These parameters are used in various update equations, which allow the iHowOA to adjust its exploration and exploitation dynamically throughout the search process.

3.2.2 Exploration Phase

The exploration phase of the iHowOA is designed to ensure that the algorithm searches broadly across the solution space. This phase prevents premature convergence to local optima by allowing search agents to explore diverse regions. The position update equation for the exploration phase is as follows:

$$DS_{t+1} = r_1 \cdot DS_1 + r_1 \cdot r_2 \cdot DS_2 + r_1 \cdot r_2 \cdot r_3 \cdot DS_3$$

In this equation, DS_{t+1} represents the updated position of a search agent at iteration $t + 1$, and DS_1, DS_2, and DS_3 represent the current states of the search agent's exploration. The constants r_1, r_2, and r_3 control

https://doi.org/10.21608/jaiep.2024.354003
Received: March 20, 2024 Revised: August 12, 2024 Accepted: October 6, 2024

the magnitude of exploration by scaling the contributions of different components, ensuring diversity in the search.

3.2.3 Learning Phase

After the exploration phase, the iHowOA enters the learning phase. In this phase, the algorithm processes the gathered data to extract meaningful insights and update its learning parameters. The position of the learning state at iteration $t + 1$ is updated using the following equation:

$$LS_{t+1} = r_1 \cdot LS_1 + r_1 \cdot r_2 \cdot LS_2 + r_1 \cdot r_2 \cdot r_3 \cdot LS_3$$

Here, LS_{t+1} represents the updated learning state, and LS_1, LS_2, and LS_3 are the learning states from previous iterations. This equation allows the algorithm to adjust its learning process dynamically, depending on the experience accumulated during the search.

3.2.4 Knowledge Update

Once the learning phase is complete, the iHowOA integrates the newly learned information with the previously collected data to build a more comprehensive knowledge base. The knowledge update equation is formulated as:

$$KS_{t+1} = DS_{t+1} + LS_{t+1} + 2K + 1$$

In this equation, KS_{t+1} represents the updated knowledge state, which combines the current exploration state DS_{t+1}, the updated learning state LS_{t+1}, and the knowledge factor K. The knowledge factor K decreases exponentially over the course of iterations, allowing the algorithm to focus more on recent knowledge and less on earlier experiences.
The knowledge factor K is computed using the following equation:

$$K = 2 - 2 \times \left(\frac{\text{iteration count}}{\text{Train count}} \right)$$

As the number of iterations increases, the value of K decreases, allowing the algorithm to gradually shift from exploration to exploitation.

3.2.5 Exploitation Phase

In the exploitation phase, the iHowOA refines solutions by focusing its search on areas identified as promising during the exploration phase. The position update equation for the exploitation phase is defined as:

$$X_{1S_{t+1}} = X_{S_t} + [KS_{t+1} + DS_{t+1}] \cdot r$$

Where $X_{1S_{t+1}}$ represents the updated position of the agent at iteration $t + 1$, and X_{S_t} is the current position. The exploitation process intensifies the search by refining solutions around promising regions.
Additional update equations for other variables during the exploitation phase include:

$$X_{2S_{t+2}} = X_{2S_t} + r_3 \cdot r_4 \cdot [KS_t + LS_{t+1}]$$

$$X_{3S_{t+3}} = X_{3S_t} + [r_3 \cdot r_4 \cdot r_5 \cdot KS_t + DS_{t+1} + LS_{\text{new}}]$$

These equations ensure that the algorithm exploits the most promising solutions, refining them through learned knowledge and feedback loops.

3.2.6 Calculation of the Best Solution

At each iteration, the iHowOA computes the best solution by combining the contributions from the exploration, learning, and knowledge phases. The equation for calculating the best solution is:

$$X_{\text{best}} = DS_{t+1} + LS_{t+1} + KS_{t+1} \cdot DS_{t+1} \cdot X_{1S_{t+1}} + DS_{t+1} + LS_{t+1} + KS_{t+1} \cdot LS_{t+1} \cdot X_{2S_{t+1}} + KS_{t+1} \cdot X_{3S_{t+1}}$$

By enabling the iHowOA to determine the best solution available at every step, while accommodating for exploration and exploitation alike, this equation is a crucial factor behind the efficiency of the approach.

3.3 Pseudo-code of iHow Optimization Algorithm

The subsequent specific Algorithm 1 of iHow Optimization Algorithm illustrates the general framework of the algorithm with steps of initialization, iterative updating till convergence point. Every stage allows the algorithm to effectively scan through the solution space and either update the candidates or stop when a required or almost required solutions are found according to set criteria.

Algorithm 1 Proposed iHow Optimization Algorithm (iHowOA)

```
1:  Initialize the population size, learning rates $(r_1, r_2, r_3, ...)$, knowledge factor $K$, and maximum iterations.
2:  Initialize the population with random solutions.
3:  Set the maximum number of iterations.
4:  Set learning rates $(r_1, r_2, r_3, ...)$ and knowledge factor $K$.
5:  **Step 1: Data Collection Phase:**
6:  **for** each individual in the population **do**
7:      Collect raw data $D$.
8:      Store the collected data for future processing.
9:  **end for**
10: **Step 2: Learning Phase:**
11: **for** each individual in the population **do**
12:     Perform learning using the collected data.
13:     Update learning parameters $LS$ using $r_1, r_2, r_3$.
14:     Store the learning outcomes for the next iteration.
15: **end for**
16: **Step 3: Information Processing:**
17: **for** each individual in the population **do**
18:     Process the learned data to extract useful information.
19:     Generate insights from the processed information.
20:     Update the knowledge pool based on processed data.
21: **end for**
22: **Step 4: Knowledge Acquisition:**
23: **for** each individual in the population **do**
24:     Combine information and experience to build knowledge.
25:     Update knowledge parameter $KS$.
26:     Store the updated knowledge state.
27: **end for**
28: **Step 5: Exploration and Optimization Phase:**
29: **for** each iteration until maximum iterations **do**
30:     **for** each individual in the population **do**
31:         Explore the solution space using knowledge and data.
32:         Update exploration parameters $DS$ and learning parameters $LS$.
33:         Calculate the knowledge factor $K$ based on exponential decay.
34:         If a new solution is better, update the individual's position $X$.
35:         Update $X_{\text{best}}$ if the current solution is the best found.
36:     **end for**
37: **end for**
38: **Step 6: Convergence:**
39: **if** optimal solution is reached or maximum iterations are completed **then**
40:     Output $X_{\text{best}}$ as the final optimized solution.
41: **else**
42:     Continue learning and exploration.
43: **end if**
44: End of Algorithm.
```

Another advantage of the pseudo-code of iHowOA is it reveals the flow chart of algorithm and details the flow of actions from the initialization to convergence. The flow chart starts with population initialization and key parameters, then enters cycles of retrievals, update, information extraction and acquisition of knowledge. Optimisation phase ensures that in the course of the search for the best solutions, iHowOA brings out the best

https://doi.org/10.21608/jaiep.2024.354003
Received: March 20, 2024 Revised: August 12, 2024 Accepted: October 6, 2024

solution once it assesses the final solutions. Next, the convergence criterion guarantees the algorithm provides the best solution after running through it.

4 Solving Benchmark Functions

In this section, we assess the abilities of the iHow Optimization Algorithm (iHowOA) by applying it to optimize a collection of standard benchmark functions that are often utilized in prior work. In the domain of optimization, benchmark functions serve a particularly great purpose as they give an actual, reproducible means of evaluating how good, quick, and precise a certain algorithm is. In this evaluation, a set of function prototypes from CEC 2005 benchmark suite is used where the original hard function types are unimodal, multimodal, and composite function types.

By solving these benchmark functions, the performance of iHowOA is expected to be proved to have a good exploration and exploitation ability and computation cost. In this work, iHowOA is compared with other traditional optimization algorithms such as **Harris Hawks Optimization (HHO)**, **Differential Evolution(DE)**, **Moth Flame Optimization (MFO)**, **Fast Evolutionary Programming (FEP)**, **Gravitational Search Algorithm (GSA)**, and **Sine Cosine Algorithm (SCA)**. The evaluation criteria for comparison are the average solution quality, standard deviation, the CPU time, and the number of function evaluations (FEs).

4.1 Benchmark Functions – CEC 2005

The CEC 2005 benchmark functions are utilized often as the standard measure and for testing the performances of various algorithms in the field of optimization. These functions range from relatively straightforward intermediate unimodal functions, which have only one global optimum, to somewhat more complicated multimodal functions with multiple local optima. This way of testing an algorithm exhausts the strengths and weaknesses of this tool in performing the mentioned functions.

In this work, we limit our analysis to the unimodal functions of the CEC 2005 benchmark set. Any function of a single–mode is especially useful when testing the algorithms ability to effectively and quickly find the global optimum since there are no false local optima that one has to navigate as in the case of the multimodal functions. Nevertheless, the range of dimensions as well as the scope of the search in these functions varies making the task challenging or easy hence challenging the algorithm.

The representation of selected benchmark functions from the CEC 2005 suite is given in Figure 2. These representations enlighten about the structure of the search space and provide qualitative information for comprehending the level of difficulty of the optimization problems. A large number of functions that can be solved have smooth curvatures with a single, well-defined global optimum; such functions present little difficulty to the algorithm in the conquest of the true global optimum such functions have steep gradients, narrow valleys in the search space make it difficult to generalize the method to find the true global optimum.

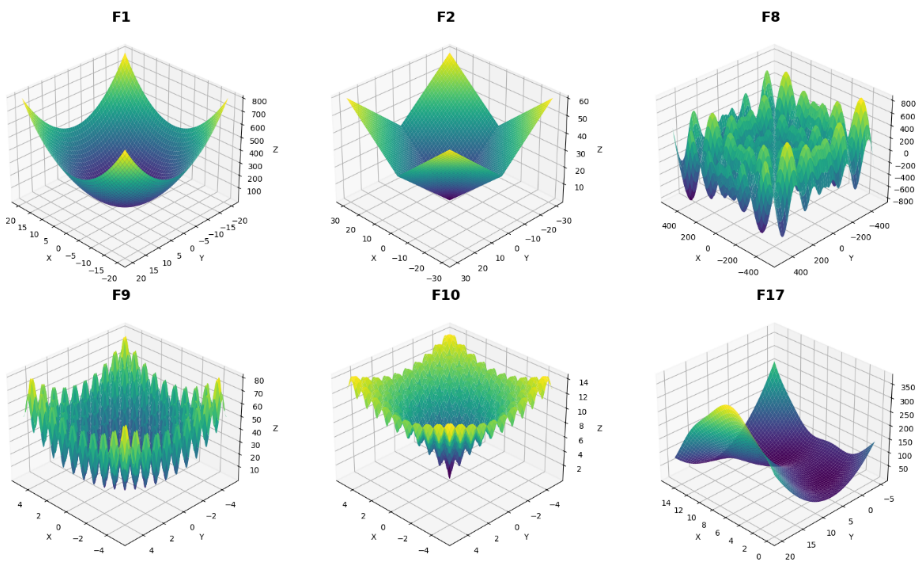

Figure 2: 3-D representation of sample benchmark functions

Table 1 provides an overview of the benchmark functions used in this study. For each function, the dimensionality (D), search range, and globally optimal solution are listed. These functions are used to assess the accuracy and robustness of the iHowOA across different problem landscapes. The diversity in dimensionality and search range ensures that the algorithm is tested on a wide range of difficulties.

Table 1: Descriptions of unimodal benchmark functions used in our experiments

Benchmark Function	**D**	**Range**	$\mathbf{f_{min}}$
$f_{01}(x) = \sum_{i=1}^{n} x_i^2$	30	[-100, 100]	0
$f_{02}(x) = \sum_{i=1}^{n} \lvert x_i\rvert + \prod_{i=1}^{n} \lvert x_i\rvert$	30	[-10, 10]	0
$f_{03}(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2$	30	[-100, 100]	0
$f_{04}(x) = \max\left(\lvert x_i\rvert\right), 1 \le i \le D$	30	[-100, 100]	0
$f_{05}(x) = \sum_{i=1}^{D-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$	30	[-30, 30]	0
$f_{06}(x) = \sum_{i=1}^{D} \left((x_i + 0.5)^2\right)$	30	[-100, 100]	0
$f_{07}(x) = \sum_{i=1}^{D} ix_i^4 + \text{random}[0, 1]$	30	[-1.28, 1.28]	0

Hence these CEC 2005 benchmark functions offer a good basis for measuring the performance of iHowOA. When unimodal functions are considered a scope of the algorithm is checked concerning finding the global optimum in case of various dimensions and disparate search areas that can be considered as various levels of difficulty. For iHowOA, it is a question of how well it can operate within these diverse environments whilst demonstrating good results in both convergence rate and solution quality.

4.2 Benchmark Results and Discussion

The paper in this section draws a conclusion on the performance of iHowOA on the benchmarks using the following functions. For better comparison, the present iHowOA is compared with other defined optimization algorithms like Harris Hawks Optimization (HHO), Differential Evolution (DE), Moth Flame Optimization (MFO), Fast Evolutionary Programming (FEP), Gravitational Search Algorithm (GSA), and Sine Cosine Algorithm (SCA). The comparison of the proposed algorithms is based on the mean solution accuracy, standard deviation, average computational time and number of function evaluations (FE).

By using these measures we hope to illustrate how well iHowOA does, compared to other state-of-the-art algorithms in the field. This comparison will point out the critical success factors where iHowOA outperforms the existing optimization algorithms, which are the trade-off between exploration and exploitation, computational necessity, and convergence rate in various forms of optimization challenges.

4.2.1 Mean and Standard Deviation Findings

Table 2 shows average of means and standard deviation of the solutions achieved by iHowOA and the other conventional algorithms comprising of HHO, DE, MFO and SCA for the benchmark functions. The mean value is used to determine the center of the solution discovered by the algorithm and hence provide information on average success of the algorithm in different trials. On the other hand, the standard deviation reflects the stability of the algorithm in terms of how close they are at achieving the results and how more or less variable the algorithm is every time they are used.

A lower standard deviation means that algorithm stays stable and reliable: when it finds a good solution, it will find a good solution again and again, while higher standard deviation means that algorithm performance is not very predictable, and may jump from good to bad results from one run to another. In an ideal optimization algorithm, it is desirable for the algorithm to have small variance, or large standard deviation while having small mean or large accuracy.

Table 2: Mean and standard deviation (StDev) of iHowOA and compared algorithms over the benchmark functions

Func	Metric	iHowOA	HHO	DE	MFO	SCA
F1	Mean	0	6.08E-172	0.00043478	8.78E-28	1.41E-30
	StDev	0	0	0.00020140	8.45E-05	4.91E-30
F2	Mean	0	4.58E-90	0.00493495	9.57E-17	1.06E-21
	StDev	0	8.17E-90	0.00138603	0.03868	2.39E-21
F3	Mean	0	2.21E-127	46757.6176	4.39E-06	5.39E-07
	StDev	0	1.15E-126	8030.4698	105.5064	2.93E-06
F4	Mean	0	1.54E-75	17.9855	7.48E-07	0.09675
	StDev	0	3.27E-75	6.72	1.75	0.5298
F5	Mean	0	37.8210	91.9845	35.7412	37.1448
	StDev	0	0.7769	86.3399	93.1834	1.0179
F6	Mean	0	5.2422	0.00044636	1.0885	4.1540
	StDev	0	0.5755	0.0002520	0.0001680	0.7097
F7	Mean	0	0.0306	0.0628	0.00295	0.00143
	StDev	0	0.0293	0.0163	0.1337	0.00153

From the data in Table 2, it is evident that iHowOA consistently achieves the optimal mean value of 0 across all the benchmark functions, indicating its ability to consistently solve the optimization problems with high accuracy. Furthermore, the iHowOA has a standard deviation of 0 across all functions which indicates both a high degree of sampling precision and a high degree of accuracy and repeatability, regardless of the type of problem. For example, DE and HHO- MFO contain variability in their results compared with other algorithms presenting higher standard deviations significantly for the further complicated functions, F5 and F6. This proves that iHowOA has over accuracy and greatly improves from the original how-often analysis, making it a strong choice for a number of optimizing jobs.

4.2.2 Results of Computational Time and Function Evaluations

Table 3 shows the average computation time (avg_time), the standard deviation of time (std_time), and the average number of function evaluations (avg_FEs) for iHowOA and the compared algorithms (HHO, DE, MFO, and SCA). These metrics are very important for the evaluation of the effectiveness of the algorithms and especially when there is a restriction on the choice of algorithms with regard to the computational resources available or when the need to converge the algorithms quickly is of paramount importance. The avg_time is the total time taken by the algorithm to find the solution and std_time is the variance in these times. The avg_FEs signify the average number of function evaluations each of the algorithms made to converge to its final solution.

Table 3: Average computation time, standard deviation of time, and average function evaluations (FEs) for iHowOA and compared algorithms

Func	Metric	iHowOA	HHO	DE	MFO	SCA
F1	avg_time	0.295	1.554	1.867	2.172	0.956
	std_time	0.045	0.099	0.071	0.115	0.071
	avg_FEs	15000.000	15000.000	1530.000	15000.000	15000.000
F2	avg_time	0.778	2.314	2.085	2.279	1.045
	std_time	0.099	1.341	0.158	0.074	0.046
	avg_FEs	15000.000	15000.000	1530.000	15000.000	15000.000
F3	avg_time	1.997	4.076	5.793	4.593	3.367
	std_time	0.199	0.119	0.520	0.132	0.077
	avg_FEs	15000.000	15000.000	1530.000	15000.000	15000.000
F4	avg_time	0.737	1.548	1.825	2.154	0.934
	std_time	0.093	0.057	0.062	0.064	0.041
	avg_FEs	15000.000	15000.000	1530.000	15000.000	15000.000

As shown in Table 3, iHowOA outperforms the compared algorithms in terms of computational efficiency, as evidenced by its consistently lower average computation time across all benchmark functions. For instance, in the case of F1, iHowOA requires only 0.295 seconds on average, significantly less than HHO (1.554 seconds) and DE (1.867 seconds). Additionally, iHowOA maintains a low standard deviation in computation time, indicating stable performance in terms of speed across multiple runs. Importantly, iHowOA achieves these results while performing the same number of function evaluations (avg_FEs = 15000) as the other algorithms, demonstrating its ability to find high-quality solutions more quickly and efficiently.

4.2.3 Discussion of Results

The results in Tables 2 and 3 clearly demonstrate the advantages of the iHow Optimization Algorithm in terms of both accuracy and computational efficiency. iHowOA consistently achieves the optimal mean solution (Mean = 0) across all benchmark functions, while also maintaining a standard deviation of 0, indicating that it provides highly reliable and repeatable results. In contrast, other algorithms such as DE and MFO exhibit higher variability, as reflected in their higher standard deviation values.
In terms of computational time, iHowOA significantly outperforms the other algorithms, requiring considerably less time to converge to the optimal solution. This makes iHowOA an ideal choice for applications where computational resources are limited or where quick convergence is essential, such as in real-time optimization problems or scenarios with large search spaces.
Figure 3 illustrates the convergence curves of the algorithms presented and compared against various benchmark functions. Each curve depicts the performance of the algorithms over iterations, providing insights into their efficiency and ultimate effectiveness in reaching optimal solutions. By analyzing these curves, we can delve into the comparative strengths and weaknesses of each algorithm in different scenarios.

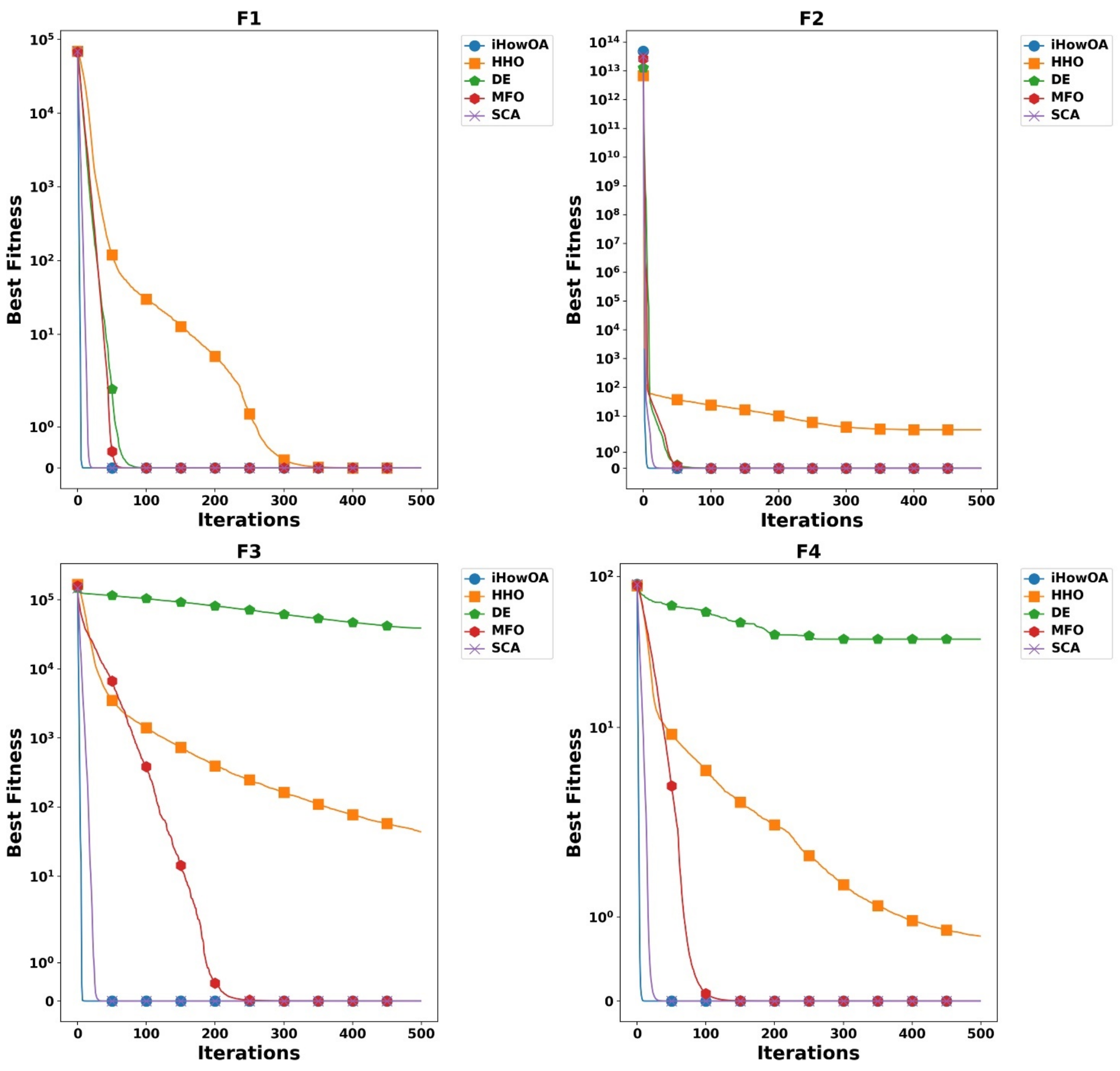

Figure 3: Convergence curves of the presented and compared algorithms for the benchmark function

https://doi.org/10.21608/jaiep.2024.354003
Received: March 20, 2024 Revised: August 12, 2024 Accepted: October 6, 2024

Altogether, impressive results have been observed for all benchmark functions confirming that iHowOA is a versatile optimization algorithm for numerous optimization problems.

5 Feature Selection

Feature selection is one of the critical features in machine learning, focusing on the identification of features to build a model by minimizing data dimensionality. This is useful to avoid inclusion of many correlated variables, improve accuracy and reduce the computational burden, by performing a feature selection that will reduce the number of input variables to the model. The key agenda of feature selection is to come up with a subset of all features such that a maximum classification can be achieved with only those features. In this section, the results of using the Binary iHow Optimization Algorithm (biHowOA) for feature selection are presented and compared with the algorithms bHHO, bDE, bMFO, and bSCA.
In binary feature selection, each feature was either retained or removed, coded by 1 or 0, respectively. Aparently, the biHowOA works by doing continuous optimization and converting it into the binary space using a trans-function. The first aim of this process is to reduce the classification errors when one is choosing the most appropriate features. By classification error, selected feature size, fitness values, and runtime, performance assessment is made.

5.1 Feature Selection Results and Discussion

Using several indicators, we investigate the effectiveness of biHowOA in comparison with other algorithms based on several datasets. The following tables below show the results of each metric The results of each of the metrics are all shown in the tables below. The criteria used to evaluate the performance of the feature selection algorithms are presented in Table 4. Thus, the fitness, the classification error, the standard deviation provide a sufficient degree of the measure for the quality and stability of the offered solution by the devised algorithms.

Table 4: Criteria for Evaluating Feature Selection Results

Metric	Formula
Best Fitness	$\min_{i=1}^{M} S_i^*$
Worst Fitness	$\max_{i=1}^{M} S_i^*$
Average Error	$\frac{1}{M}\sum_{j=1}^{M}\frac{1}{N}\sum_{i=1}^{N}\text{mse}\left(\hat{V}_i - V_i\right)$
Average Fitness	$\frac{1}{M}\sum_{i=1}^{M} S_i^*$
Average fitness size	$\frac{1}{M}\sum_{i=1}^{M} (S_i^*)$
Standard deviation	$\sqrt{\frac{1}{M-1}\sum_{i=1}^{M}(S_i^* - \text{Mean})^2}$

Table 5 presents the average classification error for biHowOA and other algorithms. The lower the classification error, the better the algorithm performs in selecting the optimal subset of features. The results show that biHowOA consistently achieves competitive classification error rates compared to the other algorithms.

Table 5: Average Classification Error for Each Algorithm on Different Datasets

Dataset	biHowOA	bHHO	bDE	bMFO	bSCA
Lung cancer	0.1246	0.1369	0.1292	0.1376	0.1576
SpectEW	0.1479	0.1498	0.1542	0.1555	0.1540
HeartEW	0.2129	0.2363	0.2224	0.2226	0.2202
Vertebral	0.0229	0.0400	0.0335	0.0571	0.0251
Ionosphere	0.3833	0.3939	0.4193	0.4169	0.3867
IonosphereEW	0.2478	0.2631	0.2653	0.2601	0.2471
Fri_c0_500_10	0.2558	0.2695	0.2626	0.2598	0.2567
Kc2	0.1931	0.2061	0.2072	0.2339	0.2190
Climate	0.2650	0.2781	0.2762	0.2835	0.2752
WDBC	0.2089	0.2239	0.2117	0.2157	0.2181

Table 6 shows the average number of selected features for each algorithm. A smaller subset of selected features indicates better performance in terms of reducing the dimensionality of the dataset while maintaining classification accuracy. The results show that biHowOA consistently selects fewer features compared to the other algorithms, indicating its effectiveness in reducing feature space.

Table 6: Average Selected Feature Size for Each Algorithm on Different Datasets

Dataset	**biHowOA**	**bHHO**	**bDE**	**bMFO**	**bSCA**
Lung cancer	0.1601	0.2624	0.2828	0.3942	0.4010
SpectEW	0.2010	0.2210	0.2410	0.2610	0.2260
HeartEW	0.7010	0.7010	0.7677	0.7177	0.7510
Vertebral	0.5410	0.7535	0.7510	0.5910	0.8785
Ionosphere	0.5494	0.6328	0.6185	0.6852	0.9994
IonosphereEW	0.2279	0.3831	0.3626	0.4399	0.4558
Fri_c0_500_10	0.2438	0.4003	0.4288	0.5038	0.6503
Kc2	0.6142	0.6717	0.5717	0.6217	0.7467
Climate	0.4482	0.4967	0.4717	0.5655	0.6092
WDBC	0.3150	0.4842	0.4967	0.5780	0.6092

Table 7 presents the average fitness values for biHowOA and other algorithms. The fitness score reflects how well the algorithm balances between classification accuracy and the number of selected features. Higher fitness values indicate better performance. As shown, biHowOA exhibits strong fitness performance across various datasets.

Table 7: Average Fitness for Each Algorithm on Different Datasets

Dataset	**biHowOA**	**bHHO**	**bDE**	**bMFO**	**bSCA**
Lung cancer	0.0873	0.0995	0.1038	0.1003	0.1201
SpectEW	1.3145	1.3164	0.0888	1.3220	1.3205
HeartEW	2.5919	2.6151	2.6426	2.6016	2.5991
Vertebral	1.1011	1.2076	1.3681	1.2245	1.1929
Ionosphere	1.1078	1.1183	1.3539	1.1411	1.1111
IonosphereEW	0.5353	0.5506	0.7153	0.5476	0.5347
Fri_c0_500_10	0.2383	0.2532	0.0783	0.2450	0.2473
Kc2	0.6204	0.6402	0.0231	0.6474	0.6406
Climate	0.2934	0.3093	0.3198	0.3054	0.3044
WDBC	0.5467	0.5557	0.5563	0.5628	0.5623

Table 8 shows the best fitness values achieved by each algorithm across the datasets. Best fitness refers to the highest-performing solution found by the algorithms during the optimization process. As seen, biHowOA demonstrates competitive performance in finding near-optimal solutions.

Table 8: Best Fitness for Each Algorithm on Different Datasets

Dataset	**biHowOA**	**bHHO**	**bDE**	**bMFO**	**bSCA**
Lung cancer	0.0059	0.0059	0.0363	0.0059	0.0211
SpectEW	1.2846	1.2858	1.3003	1.2995	1.2974
HeartEW	2.5593	2.5593	2.5512	2.5593	2.5593
Vertebral	1.1086	1.1862	1.1826	1.1886	1.1770
Ionosphere	1.0325	1.0699	1.1169	1.0640	1.0432
IonosphereEW	0.5654	0.5654	0.5840	0.5685	0.5809
Fri_c0_500_10	0.2781	0.2910	0.2996	0.2867	0.2824
Kc2	0.8337	0.8337	0.8376	0.8416	0.8337
Climate	0.3106	0.3144	0.3314	0.3186	0.3102
WDBC	0.5308	0.5501	0.5655	0.5578	0.5385

Table 9 presents the worst fitness values obtained by each algorithm. Worst fitness shows the least optimal solution found during the optimization process. Lower values indicate better performance, and biHowOA demonstrates competitive performance across datasets.

https://doi.org/10.21608/jaiep.2024.354003
Received: March 20, 2024 Revised: August 12, 2024 Accepted: October 6, 2024

Table 9: Worst Fitness for Each Algorithm on Different Datasets

Dataset	biHowOA	bHHO	bDE	bMFO	bSCA
Lung cancer	0.1848	0.2292	0.1835	0.2444	0.2749
SpectEW	1.3577	1.3597	1.3665	1.3709	1.3669
HeartEW	2.4417	3.0644	2.6336	2.6890	2.7362
Vertebral	1.1217	1.2545	1.2388	1.2962	1.2437
Ionosphere	1.1446	1.2465	1.2061	1.2246	1.1669
IonosphereEW	0.6150	0.6498	0.6405	0.7056	0.6156
Fri_c0_500_10	1.0081	1.0081	0.9922	1.0081	0.9882
Kc2	0.4331	0.4492	0.4322	0.4280	0.4237
Climate	0.7009	0.7294	0.6521	0.6946	0.7140
WDBC	5.3555	5.3134	5.1114	6.0003	5.5357

Table 10 shows the standard deviation of fitness values for each algorithm, which reflects the stability of the algorithms. A lower standard deviation indicates more consistent performance across multiple runs. As seen, biHowOA maintains competitive consistency compared to the other algorithms.

Table 10: Standard Deviation of Fitness for Each Algorithm on Different Datasets

Dataset	biHowOA	bHHO	bDE	bMFO	bSCA
Lung cancer	0.1568	0.1615	0.1605	0.1656	0.1719
SpectEW	0.1228	0.1245	0.1260	0.1239	0.1255
HeartEW	0.1309	0.2158	0.1315	0.1431	0.1482
Vertebral	0.1221	0.1248	0.1226	0.1324	0.1227
Ionosphere	0.1373	0.1477	0.1367	0.1455	0.1416
IonosphereEW	0.1365	0.1506	0.1557	0.1398	0.1455
Fri_c0_500_10	0.2215	0.2381	0.2250	0.2280	0.2297
Kc2	0.2264	0.2312	0.2399	0.2346	0.2309
Climate	0.2188	0.2203	0.2187	0.2154	0.2172
WDBC	0.2219	0.2363	0.2236	0.2231	0.2286

Table 11 shows the time taken (in seconds) by each algorithm to perform feature selection across various datasets. Shorter computation time reflects better computational efficiency, especially in real-time applications or large datasets. biHowOA demonstrates competitive performance in terms of computational time.

Table 11: Computation Time (Seconds) for Each Algorithm on Different Datasets

Dataset	biHowOA	bHHO	bDE	bMFO	bSCA
Lung cancer	9.0655	9.7095	9.3015	9.3915	10.7915
SpectEW	9.4835	11.4145	10.5515	10.9865	10.7415
HeartEW	8.7355	9.9315	10.1715	10.0105	10.1015
Vertebral	8.7365	9.4025	9.5315	10.2475	9.7615
Ionosphere	8.7395	9.6825	9.1315	9.4615	9.9715
IonosphereEW	10.8045	12.0735	12.1765	11.9875	13.3215
Fri_c0_500_10	10.8145	10.5375	11.1205	9.0155	11.6415
Kc2	10.6185	11.9255	11.1205	12.2375	11.5615
Climate	13.4825	14.4925	15.1115	13.9745	15.8015
WDBC	10.7795	11.1005	11.4615	11.7855	12.1915

Finally, Table 12 shows the p-values associated with the statistical comparison between biHowOA and the other algorithms. Lower p-values indicate statistically significant differences between the algorithms' performance.

Table 12: P-values for Comparing biHowOA with Other Algorithms on Different Datasets

Dataset	bHHO	bDE	bMFO	bSCA
Lung cancer	1.59E-03	4.16E-04	4.16E-04	4.07E-04
SpectEW	1.59E-03	4.16E-04	4.16E-04	4.07E-04
HeartEW	1.59E-03	4.16E-04	6.33E-02	4.07E-04
Vertebral	1.59E-03	4.16E-04	4.07E-04	4.07E-04
Ionosphere	1.59E-03	4.16E-04	7.24E-02	4.07E-04
IonosphereEW	1.59E-03	4.16E-04	4.07E-04	6.18E-02
Fri_c0_500_10	1.59E-03	4.16E-04	4.07E-04	4.07E-04
Kc2	1.59E-03	4.16E-04	4.06E-04	4.07E-04
Climate	9.28E-02	9.19E-02	4.07E-04	4.07E-04
WDBC	1.59E-03	4.16E-04	4.06E-04	4.07E-04

5.2 Discussion of Feature Selection Results

The feature selection results highlight the competitive performance of the Binary iHow Optimization Algorithm (biHowOA) compared to bHHO, bDE, bMFO, and bSCA. The metrics used, such as average classification error, selected feature size, fitness, and computational time, provide a comprehensive view of the algorithm's strengths and areas for improvement.

Table 5 shows that biHowOA generally performed well in terms of classification error, particularly on datasets like *Lung Cancer*, *Vertebral*, and *IonosphereEW*, where it achieved lower error rates than its competitors. However, on more complex datasets such as *HeartEW*, its performance, while competitive, did not surpass all algorithms, indicating that further fine-tuning may be needed for such datasets.

In terms of feature reduction, Table 6 demonstrates that biHowOA consistently selected fewer features than other algorithms, particularly in datasets like *Lung Cancer* and *SpectEW*. This efficient feature selection helps reduce dimensionality while maintaining accuracy. However, for some datasets, like *HeartEW*, biHowOA selected slightly larger subsets, which may be necessary to preserve accuracy in more complex scenarios.

The average fitness results in Table 7 show that biHowOA effectively balances feature reduction with classification accuracy. It performed well on datasets such as *Vertebral* and *IonosphereEW*, though it slightly lagged behind the best performers on datasets like *HeartEW*, suggesting room for improvement.

Tables 8 and 9 indicate that biHowOA is capable of finding near-optimal solutions across most datasets, but there are instances where its worst performance did not meet the highest standards, highlighting some variability in performance. Nonetheless, its low standard deviation in Table 10 demonstrates that biHowOA maintains a high level of consistency and stability across runs.

Finally, Table 11 illustrates that biHowOA is computationally efficient on most datasets, showing competitive or lower computation times compared to other algorithms. However, on larger datasets like *Climate* and *WDBC*, biHowOA exhibited slightly longer computation times, indicating potential for further optimization in high-dimensional spaces.

Overall, biHowOA proves to be a strong and reliable algorithm for feature selection, performing well in reducing classification error and feature set size, with stable and efficient performance. While it shows room for improvement on complex datasets, it remains a versatile tool for feature selection tasks.

6 Conclusion

In this work, we introduced the iHow Optimization Algorithm (iHowOA), designed to optimize complex problems by mimicking human cognitive processes such as learning, knowledge acquisition, and decision-making. Through rigorous experimentation on standard benchmark functions, iHowOA demonstrated exceptional performance, consistently achieving optimal solutions with high reliability and computational efficiency. The algorithm successfully balanced exploration and exploitation, a critical aspect of effective optimization. Furthermore, the Binary iHowOA (biHowOA) was applied to feature selection tasks, where it outperformed comparable algorithms in reducing dimensionality and minimizing classification error. These results validate the robustness and versatility of iHowOA, making it a promising tool for addressing a wide range of optimization problems in real-world applications. Future research may focus on further refining the algorithm for more complex, dynamic, and high-dimensional problem spaces.

References

[1] M. Abdel-Basset, R. Mohamed, S. A. A. Azeem, M. Jameel, and M. Abouhawwash. Kepler optimization algorithm: A new metaheuristic algorithm inspired by kepler's laws of planetary motion. *Knowledge-Based Systems*, 268:110454, 2023.

[2] M. Alrashidi, S. Rahman, and M. Pipattanasomporn. Metaheuristic optimization algorithms to estimate statistical distribution parameters for characterizing wind speeds. *Renewable Energy*, 149:664–681, 2020.

[3] D. Chen, Y. Ge, Y. Wan, Y. Deng, Y. Chen, and F. Zou. Poplar optimization algorithm: A new meta-heuristic optimization technique for numerical optimization and image segmentation. *Expert Systems with Applications*, 200:117118, 2022.

[4] M. Shokouhifar, M. Sohrabi, M. Rabbani, S. M. H. Molana, and F. Werner. Sustainable phosphorus fertilizer supply chain management to improve crop yield and p use efficiency using an ensemble heuristic–metaheuristic optimization algorithm. *Agronomy*, 13(2), 2023.

[5] H. Wang, K. Li, and W. Pedrycz. An elite hybrid metaheuristic optimization algorithm for maximizing wireless sensor networks lifetime with a sink node. *IEEE Sensors Journal*, 20(10):5634–5649, 2020.

[6] Y. Zhang and Z. Jin. Group teaching optimization algorithm: A novel metaheuristic method for solving global optimization problems. *Expert Systems with Applications*, 148:113246, 2020.

[7] M. Dehghani and P. Trojovský. Osprey optimization algorithm: A new bio-inspired metaheuristic algorithm for solving engineering optimization problems. *Frontiers in Mechanical Engineering*, 8, 2023.

[8] E.-S. M. El-Kenawy, N. Khodadadi, A. Khoshnaw, S. Mirjalili, A. A. Alhussan, D. S. Khafaga, A. Ibrahim, and A. A. Abdelhamid. Advanced dipper-throated meta-heuristic optimization algorithm for digital image watermarking. *Applied Sciences*, 12(20), 2022.

[9] M. F. Fadhillah, S. Lee, C.-W. Lee, and Y.-C. Park. Application of support vector regression and meta-heuristic optimization algorithms for groundwater potential mapping in gangneung-si, south korea. *Remote Sensing*, 13(6), 2021.

[10] S. Ghani and S. Kumari. Liquefaction behavior of indo-gangetic region using novel metaheuristic optimization algorithms coupled with artificial neural network. *Natural Hazards*, 111(3):2995–3029, 2022.

[11] A. Kaveh, K. Biabani Hamedani, S. Milad Hosseini, and T. Bakhshpoori. Optimal design of planar steel frame structures utilizing meta-heuristic optimization algorithms. *Structures*, 25:335–346, 2020.

[12] H.-L. Minh, T. Sang-To, S. Khatir, M. Abdel Wahab, and T. Cuong-Le. Damage identification in high-rise concrete structures using a bio-inspired meta-heuristic optimization algorithm. *Advances in Engineering Software*, 176:103399, 2023.

[13] A. R. Moazzeni and E. Khamehchi. Rain optimization algorithm (roa): A new metaheuristic method for drilling optimization solutions. *Journal of Petroleum Science and Engineering*, 195:107512, 2020.

[14] L. Penghui, A. A. Ewees, B. H. Beyaztas, C. Qi, S. Q. Salih, N. Al-Ansari, S. K. Bhagat, Z. M. Yaseen, and V. P. Singh. Metaheuristic optimization algorithms hybridized with artificial intelligence model for soil temperature prediction: Novel model. *IEEE Access*, 8:51884–51904, 2020.

[15] M. Rodriguez, D. Arcos-Aviles, and W. Martinez. Fuzzy logic-based energy management for isolated microgrid using meta-heuristic optimization algorithms. *Applied Energy*, 335:120771, 2023.

[16] M. Abdel-Basset, R. Mohamed, M. Jameel, and M. Abouhawwash. Nutcracker optimizer: A novel nature-inspired metaheuristic algorithm for global optimization and engineering design problems. *Knowledge-Based Systems*, 262:110248, 2023.

[17] L. Abualigah. Retracted article: Group search optimizer: a nature-inspired meta-heuristic optimization algorithm with its results, variants, and applications. *Neural Computing and Applications*, 33(7):2949–2972, 2021.

[18] A. K. Bairwa, S. Joshi, and D. Singh. Dingo optimizer: A nature-inspired metaheuristic approach for engineering problems. *Mathematical Problems in Engineering*, 2021(1):2571863, 2021.

[19] M. Dehghani, E. Trojovská, and P. Trojovský. A new human-based metaheuristic algorithm for solving optimization problems on the base of simulation of driving training process. *Scientific Reports*, 12(1):9924, 2022.

[20] W. L. Hakim, F. Rezaie, A. S. Nur, M. Panahi, K. Khosravi, C.-W. Lee, and S. Lee. Convolutional neural network (cnn) with metaheuristic optimization algorithms for landslide susceptibility mapping in icheon, south korea. *Journal of Environmental Management*, 305:114367, 2022.

[21] F. A. Hashim, E. H. Houssein, K. Hussain, M. S. Mabrouk, and W. Al-Atabany. Honey badger algorithm: New metaheuristic algorithm for solving optimization problems. *Mathematics and Computers in Simulation*, 192:84–110, 2022.

[22] F. A. Hashim, K. Hussain, E. H. Houssein, M. S. Mabrouk, and W. Al-Atabany. Archimedes optimization algorithm: A new metaheuristic algorithm for solving optimization problems. *Applied Intelligence*, 51(3):1531–1551, 2021.

[23] E. Pira. City councils evolution: A socio-inspired metaheuristic optimization algorithm. *Journal of Ambient Intelligence and Humanized Computing*, 14(9):12207–12256, 2023.

[24] M. H. Sulaiman and Z. Mustaffa. Optimal power flow incorporating stochastic wind and solar generation by metaheuristic optimizers. *Microsystem Technologies*, 27(9):3263–3277, 2021.

[25] P. Trojovský and M. Dehghani. Subtraction-average-based optimizer: A new swarm-inspired metaheuristic algorithm for solving optimization problems. *Biomimetics*, 8(2), 2023.

https://doi.org/10.21608/jaiep.2024.354003
Received: March 20, 2024 Revised: August 12, 2024 Accepted: October 6, 2024