JAIEP

NiOA: A Novel Metaheuristic Algorithm Modeled on the Stealth and Precision of Japanese Ninjas

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

This paper presents a new metaheuristic optimization algorithm called the Ninja Optimization Algorithm (NiOA) owing to its characteristics such as stealth, precision, and adaptability of the ninjas of Japan. NiOA is proposed to avoid high exploration and exploitation costs within such complex search spaces and to avoid the problem of getting trapped in local optima. The algorithm imitates ninja searching techniques because it has a scanning phase, adapted to search large areas to look for answers, while the more specific phase is used to refine the answers found. The performance of NiOA is compared with other benchmark optimization functions and some of the frequently used CEC 2005 benchmarks. These benchmarks are well suited to test unimodal and multimodal optimization problems of good quality. Experimental results prove that NiOA can significantly provide better optimization results regarding solution quality, convergence rate, and time complexity, suggesting that NiOA is a robust algorithm for solving high-dimensional large-scale optimization problems. Furthermore, it reveals that NiOA is applicable to solve different kinds of problem spaces, signifying that NiOA can be used in practice on scientific and engineering problems.

Keywords: Ninja Optimization Algorithm, metaheuristic, exploration and exploitation, complex problem solving, adaptive optimization

MSC: 68T20; 62J05; 93B45

Doi: https://doi.org/10.21608/jaiep.2024.386693

Received: April 22, 2024 Revised: September 9, 2024 Accepted: October 15, 2024

1 introduction

Metaheuristic optimization algorithms have become a valuable source for solving optimization problems that conventional methods fail to solve satisfactorily. These algorithms offer customizable structures that can be applied to potentially significant solution spaces in the high dimensionality where exact or deterministic type methods can be inapplicable due to the existence of non-linearity, multimodality, or a lack of knowledge of the shape of the objective function. Several metaheuristic algorithms have been introduced in recent decades based on numerous natural processes and phenomena, such as evolution, swarm intelligence, and physics. Again, metaheuristic techniques have been found particularly useful due to flexibility in implementation and capability to provide reasonable suboptimal solutions within acceptable computer time for practically an extensive array of scientific and engineering problems [1–3].

The success of metaheuristic algorithms is rooted in their ability to balance two essential aspects of optimization: exploration and exploitation. Exploration is the capacity of the algorithm to search in areas of the solution space that it has not visited to avoid being trapped with local optima. At the same time, exploitation focuses on fiddling around the promising search space region to converge to optimum search points. Finding this balance effectively sets a good metaheuristic approach above the rest, as being bad at this weakens a metaheuristic [4–6].

Recently, new metaheuristic algorithms have continued to be developed and are derived from various aspects of animalology, natural processes or anthropology. Such algorithms include Particle Swarm Optimization (PSO), Genetic Algorithm (GA), Grey Wolf Optimizer (GWO), and Firefly Algorithm (FA). They are promising algorithms for solving single-objective as well as multi-objective optimization problems. However, given that several methods are available on the topic, the current work indicates that there is still room for enhancing methodological research, especially about how to enhance convergence speed and solving methods for complicated constraint problems. The operating reliability of the optimization methods across various problem areas [7–9].

In this paper, a newly developed metaheuristic algorithm called the Ninja Optimization Algorithm (NiOA) is based on the features of the ninjas in Japan's history. In this case, the ninja terminology of movement, execution, and environment provides the best framework for constructing a sound optimization strategy. NiOA uses this characteristic to construct an optimizer to successfully search between exploration and exploitation and to build a practical approach to the stochastic character of the solutions space [10–12].

A new approach is presented by the Ninja Optimization Algorithm (NiOA), which includes several features that can enhance the general performance of mainstream metaheuristic algorithms. In particular, NiOA establishes a contextual movement phase for the search and an exact expansion phase for the exploitation. This enables NiOA to adapt and fine-tune the solutions provided for high-dimensional problems and successfully solve complex optimization problems without premature convergence to a suboptimal solution. In contrast, the quality of the solutions increases continuously [13–15].

Altogether, the Ninja Optimization Algorithm presents a new idea in the expanding area of metaheuristic optimization and plays a vital role in developing more effective optimization algorithms that can solve significant, complex problems in the real world.

2 Literature Review

Most metaheuristic algorithms are applied for solving worldwide optimization issues that are difficult to solve. The initial strategies which formed the basis of classical methods include Genetic Algorithms (GA) and Particle Swarm Optimization (PSO) while the new strategies are characterized by efficiency in search and the ability to solve large and complex problems. This section offers an evaluation of metaheuristic advancements with a focus on the gaps that led to the concept development of the Ninja Optimization Algorithm (NiOA).

Deep neural networks, or DNNs, have become recognized as influential machine learning metaheuristics adopted and embraced in many practice areas since they can learn features from big data sets independently. DNNs are otherwise unique in terms of their structures and parameters, which may be optimized depending on the task. Nevertheless, the training of DNNs may consume considerable time when training big data or when used in a computationally intensive application. Moreover, finding a deep learning model's best-performing and feasible architecture within a limited time frame also appears more problematic. Thus, metaheuristics, including swarm intelligence (SI) and evolutionary computing (EC), are excellent optimization approaches based on specific theories and functions. Such approaches are general and have shown reasonable practicability in various contexts, so they can be used to fine-tune DNN models. This paper aims to present a significant review of the current state-of-the-art optimization techniques for the most essential tasks of DNNs, mainly SI and EC methods. [16] also explores the importance of these optimization techniques in defining the correct

hyperparameters and architecture for DNNs, especially when dealing with big data. Moreover, it lists a few directions in which evolutionary methods for DNNs can be improved, discussing some open problems and future trends for this fast-growing sub-lineage in the field.

The paper [17] presents a novel nature-inspired meta-heuristic algorithm, the Spider Wasp Optimization (SWO) algorithm. This algorithm is derived from female spider wasps' hunting, nest construction, and mating practices. Thus, the SWO algorithm includes several specific update techniques that adjust for various problem classes and their necessities for exploration and exploitation. The performance of the proposed SWO algorithm was evaluated through comparisons with nine recently developed and well-established meta-heuristic algorithms across four different benchmarks. We evaluated the algorithm performance on four benchmarks, including (1) a set of 23 essential benchmark functions that are unimodal and multimodal, (2) the CEC2017 benchmark, (3) the CEC2020 benchmark, and (4) the CEC2014 benchmark. These were some of the parameters used in measuring the performances of the proposed SWO algorithm. Furthermore, the capability of the SWO algorithm was tested on two engineering problems: the welded beam design problem and the pressure vessel design problem, along with parameter estimations of single-diode, double-diode and triple-diode photovoltaic models. These real-world optimization techniques were further employed to assess the algorithm's performance. The experimental analysis confirms that the SWO algorithm performs well compared to some of the most popular meta-heuristic techniques on four verified benchmarks and appears superior for solving realistic optimization problems.

In [18], the authors propose a Light Spectrum Optimizer (LSO), a new metaheuristic algorithm based on the physical phenomenon, for continuous optimization problems. The LSO algorithm is based on Light Dispersion in which several light rays pass through the rain droplets at diffusion angles, producing the rainbow of colors you see. This natural process is mimicked in the LSO to help search the multidimensional plane for an optimal solution. Three tests were thoroughly provided to verify the usability of the introduced LSO algorithm. In the first experiment, LSO was applied to the CEC 2005 benchmark suite and compared with many metaheuristic benchmark algorithms. The second experiment assessed the performance of LSO on four single-objective optimization test functions from CEC competitions: CEC2014, CEC2017, CEC2020, and CEC2022. These were then compared with the results from eleven essential optimization algorithms. Apart from benchmark testing, the LSO algorithm is also used in several engineering design problems, where it compares with the different algorithms identified in the literature. The obtained experiment results, confirmed by the subsequent statistical analysis, revealed the LSO algorithm's benefits and showed that it outperforms several representatives of established and recent optimization algorithms. These findings show that the Light Spectrum Optimizer has excellent potential as an efficient tool for addressing any optimization problem. Testing the algorithm on various benchmark functions and real-life problems demonstrates its effectiveness and flexibility in achieving better solutions for various optimization problems.

Solving optimization problems in different branches of science is a great challenge that has to be met using suitable optimization methods. In [19], the authors present the Waterwheel Plant Algorithm (WWPA), a metaheuristic with a stochastic nature mimicked from natural processes. The basic idea of WWPA is based entirely on the natural behavior of the waterwheel plant during the hunting process, in which plants are used as searchers for prey. An overview of the mathematical structure of WWPA is discussed to solve the challenging optimization issues comprehensively. The efficiency of the WWPA was evaluated using 23 objective functions of the unimodal and multimodal types. The optimization analysis of unimodal functions shows WWPA's good exploitation capabilities to converge to the optimal solution of the given functions. However, optimizing multimodal functions shows that WWPA can explore the environment and find a significant optimal solution regarding the search space. In addition, to assess its applicability in practice, the WWPA was applied to three engineering design tasks. In these contexts, the performance of the algorithm was evaluated using seven other metaheuristic algorithms that are commonly used. Overall, the simulation results and analyses point to the high efficiency of the WWPA application for solving various optimization problems and identify it as the best quantitative method for reaching the optimum.

The paper [20] introduces a new metaheuristic algorithm for population-based search named the Gazelle Optimization Algorithm (GOA) that tries to mimic gazelles' survival strategy in a world entirely of predators. Each day, the gazelle understands that if it fails to outrun and out wit the predators, he becomes the dish of the day, thus to survive the gazelles have to get away from their predators every time. This information is essential for proposing a new metaheuristic algorithm based on the gazelle's survival characteristics to solve global optimization issues. The exploitation phase of the algorithm models the gazelles, which graze when the predator is not in sight or when the latter is following one of them. Interestingly, the operation of the GOA goes into the exploration phase as soon as a predator is sighted. The exploration phase includes when the gazelle tries to leave the path of a predator by moving faster and gaining a safety zone. These two phases are then performed cyclically about the termination condition and the solution of optimal solutions concerning these optimization

https://doi.org/10.21608/jaiep.2024.386693
Received: April 22, 2024 Revised: September 9, 2024 Accepted: October 15, 2024

problems. The efficiency and stability of the developed algorithm as a member of the optimization toolbox were verified using optimization test functions and chosen engineering design problems (15 conventional, 10 combined functions and 4 problems based on mechanical engineering designs). The performance of the proposed GOA is compared with nine other competing algorithms available in the literature. The simulation results that were yielded in this study corroborate the efficacy and effectiveness of the GOA algorithm in problem-solving as compared to the other nine competitive algorithms that are present in the literature. Also, the standard statistical analysis test conducted on the result proved that GOA competently solves the selected optimization problems. It also revealed that GOA was at least on par with, or in some instances quite close to, some state-of-the-art algorithms. Furthermore, the results reveal that GOA is an effective optimization algorithm that can be applied across various optimization in the domain.

Feature extraction is a critical step in a machine learning process, and although multiclass feature selection is more challenging, many classifications are binary. The problem of feature selection is usually focused on decreasing the dimensionality of the feature set and, at the same time, on the accuracy of the performance model. Various classifications of datasets exist, which can be done in different ways. However, metaheuristic algorithms receive a considerable amount of interest in solving a variety of problems in optimization. Therefore, [21] aims to provide a systematic review of the literature for solving multiclass feature selection problems using a metaheuristic algorithm that can help classifiers to select features closer, if not the best, in terms of speed and accuracy. Metaheuristic algorithms have also been presented in four primary categories depending on their behavior: Evolutionary-based algorithms, Swarm-Intelligence algorithms, Physics-based algorithms, and human-based algorithms, although some of the literature contributed more to categorization. In addition, the categories mentioned above-presented lists of metaheuristic algorithms. While searching for the solution to the problems concerning multiclass feature selection, only the papers that view metaheuristic algorithms as applied to the problems of multiclass feature selection published between 2000 and 2022 were considered as to their various categories and descriptions in detail. Some application areas for some of the metaheuristic algorithms applied for multiclass feature selection with their variations are described below.

When applying the recently developed and efficient swarm intelligence algorithms, finding the solution for mechanical design problems is often challenging. There are too many challenges to be solved, including mixed decision variables, different constraints, errors, objectives conflict, and many local optima. [22] evaluates nine categories of metaheuristic algorithms involving SSA, MVO, MFO, ASO, EBO, QSA, EO, ES and HSOGA. These algorithms are tested for efficiency on eight mechanical design problems based on solution quality and convergence, confirming that these algorithms are well suited to be applied to application problems.

As can be observed, every metaheuristic optimization algorithm needs some form of initialization; generally, the initialization step is done randomly for such optimizers. However, initialization can have some significant impacts on the performances of such algorithms. [23] compares 22 different initialization methods on the convergence and accuracy of five optimizers: DE, PSO, CS, ABC and GA. To discuss the possible effects of initialization, population size and the number of iterations, we have employed 19 different test functions with different properties and modalities. The result of the statistical ranking test, which shows considerable ranking differences, proves that the estimated coefficient is valid at 43. Notably, 37% of the functions using the DE algorithm indicate a high sensitivity when they are initialized differently; on the other hand, 73%. Of the functions deployed in a function Algorithm that employs both PSO and CS algorithms, 68% are sensitive to various initialization techniques. The simulations also reveal that initialization affects DE less than PSO and CS. Second, it can be seen that the effect of the population size is even more substantial under the restriction of the maximum feasible number of FEs. Particle swarm optimization usually involves a larger population size of particles, while the cuckoo search requires a population of a few cuckoo birds. One must also note that the differential evolution-solving capability relies more on the number of iterations, and yet a small population size with more iterations yields a better solution. Moreover, compared with other algorithms, ABC's convergence is more sensitive to initialization, and such an initialization does not significantly impact GA. Probability distribution includes beta, exponential and Rayleigh distributions, which can sometimes lead to improved performance.

The paper [24] presents a novel metaheuristic optimization algorithm, called the K-means Optimizer (KO), for solving optimization problems in numerical function optimization and engineering design problems. The KO algorithm utilizes the K-means clustering technique to define the centroid vectors of cluster regions at each iteration. They use two contrasting movement patterns to balance exploration (searching for new areas) and exploitation (optimizing existing solutions). The feasibility of a strategy for exploration or exploitation depends on a parameter that controls whether a search agent stays in a region without any improvement to the self. One of the significant uses of KO was in solving the SDI issue for a complex 3D concrete structure, a seven-story building with a total height of 25m. 2 meters. In this structure, the finite element (FE) model was solved by using the SAP2000 software. For the first time, a sub-program was created that allows data

exchange between SAP2000 and MATLAB in real time using the Open Application Programming Interface (OAPI) library to update the FE model. A statistical assessment of the KO algorithm was then made, and the Wilcoxon rank-sum and Friedman ranking tests were used. The. The work results show that the KO algorithm surpasses other algorithms on the mentioned benchmark functions. Thus, the results of this study support the ability and efficiency of the K-means Optimizer to perform various optimization problems, especially in structure damage identification (SDI).

Metaheuristics can, therefore, be described as computational techniques that are used to help control the search in a particular search space to solve an optimization problem. Due to the widespread use of large data sets in different fields, there is a constant demand to improve the metaheuristic algorithms and introduce new ones with high accuracy and performance. Given these goals, [25] presents a new meta-heuristic optimization algorithm called Crystal Structure Algorithm (CryStAl). The CryStAl algorithm is based on adding the basis to the lattice points and the crystal shaping by the points' symmetrical arrangement often observed in atoms, molecules or ions of crystalline minerals such as quartz. This natural phenomenon is then used to create the algorithm to achieve the right balance between the exploration of the search space and the exploitation. A total of 239 mathematical functions were employed to assess the performance of the CryStAl algorithm, which were dividedwaso four groups. In this study, the performance of the proposed CryStAl algorithm was tested and compared with 12 other classical and modern metaheuristic algorithms taken from the literature. The minimum, mean, and standard deviation of the KLD values and the number of function evaluations for a given tolerance were computed and reported for CryStAl and the other competitors. These outcomes, accompanied by detailed statistical analysis, proved that CryStAl algorithm has spectacular performance, which surpass several other metaheuristic techniques in most cases.

In conclusion, lots of improvements have been made in metaheuristic optimization, but some of the present algorithms have an issue like early convergence and a poor search for solutions in the large numbers of variables. These gaps are catered for by the proposed Ninja Optimization Algorithm (NiOA), which proposes a more dynamic and balanced exploration and exploitation strategy and therefore the NiOA positions it as a new, promising method in the field of optimization.

3 Proposed Ninja Optimization Algorithm (NiOA)

The Ninja Optimization Algorithm (NiOA) is proposed here as a metaheuristic algorithm derived from the ninja concepts involving stealthiness, accuracy, and flexibility. NiOA deals with major optimization issues, namely how to prevent the algorithm from converging too early and how to escape local optima through a proper balance between exploration and exploitation. The exploration phase replicates the ninja move; he must always observe his environment and cover as many aspects of the target area as possible. On the other hand, the exploitation phase follows the same surgical ethos of the ninjas and is more about enhancing and perfecting high-potential solutions. This section discusses how NiOA was developed, where it is headed, and how its mathematical model may surpass traditional optimization tools.

3.1 Inspiration and Mechanism

The basis of the NiOA is borrowed from the ninja fighters from Japan, who are famous for their secretive, accurate, and adapting techniques. Traditionally, a ninja was a warrior who moved into a territory stealthily and used tactics that entailed a lot of talent, discipline and finesse. These qualities can readily be mapped for optimization problems since they consist of a search space with large areas to explore and focus on for speed solutions.

Ninjas needed to move around the enemy terrains, within which they needed to assess dynamics and respond appropriately to alterations. This ability to traverse without being detected represents how an optimizer has to be flexible enough to scale the peaks of the search space and the plains without being stuck at a local maximum. The NiOA embodies this in the exploration phase, replicating the ninja action where they move around areas one has not explored or it is unsafe to tread, analogous to searching for opportunities without being noticed or facing loss.

Likewise, ninjas master sophisticated and purposeful movements in a focused and calculated way to hit the weakest points of an opponent with as little energy as possible to cause as much damage as possible. This characteristic is explained in the exploitation phase of NiOA, where the algorithm categorizes its search around promising solutions, which it then narrows down with precision and velocity – similar to a ninja who targets the critical area of his body.

In addition, they can adapt to the tools in the environment, the knowledge of the environment, and deception, which are also incorporated into NiOA. The algorithm's architecture provides for transient alterations in its

https://doi.org/10.21608/jaiep.2024.386693
Received: April 22, 2024 Revised: September 9, 2024 Accepted: October 15, 2024

behavior pattern based on the present situation in the search domain. This makes it possible for NiOA to strike a nice balance between exploration and exploitation. Switching between the two should depend on the problem level, just like how ninjas adjust depending on the operation.
In this way, NiOA uses the principles of stealth, precision and activity from the film about ninja fighters to design an optimization procedure that allows to solve high-dimensional optimization tasks while simultaneously being efficient. With efficient exploration and subsequent exploitation planning, NiOA guarantees it will be able to move in the problem space fluidly and purposefully, akin to the Japanese ninjas in their operations.

3.2 Experimental Setup

3.2.1 Exploration Phase

In the exploration phase, NiOA investigates diverse potential solutions across the search space to prevent getting trapped in local optima. The following position update equation governs the exploration behavior for a search agent $L_s(t+1)$ at time step $t+1$:

$$L_s(t+1) = \begin{cases} L_s(t) + r_1 \cdot (L_s(t_1) - L_s(t_2)), & \text{if specific conditions are met,} \\ \text{Random } L_s(t') \in FS(met), & \text{otherwise.} \end{cases} \tag{1}$$

Here, r_1 is a random factor that introduces variability in the movement of the search agent, while t_1 and t_2 refer to previous iterations of the algorithm. This equation ensures that the algorithm explores new regions of the search space, thereby avoiding stagnation. If certain predefined conditions are met (such as insufficient improvement in the fitness function), the algorithm randomly selects values within the feasible solution space $FS(met)$, further promoting exploration.
In addition, the position of the second variable, $D_s(t+1)$, is updated according to the following equation:

$$D_s(t+1) = D_s(t) + |D_s(t) + r_2 \cdot D_s(t)| \cdot \cos(2\pi t), \tag{2}$$

Where r_2 is another random scaling parameter. The cosine function introduces oscillatory behavior, encouraging exploration by varying the magnitude and direction of the search agent's movements. This periodic behavior helps NiOA avoid premature convergence by allowing for the exploration of both global and local areas of the search space.
Mutation Mechanism
NiOA includes a mutation mechanism to enhance the diversity of the explored solutions. The mutation operator introduces controlled randomness, which enables the algorithm to escape from local optima and discover new potential solutions. The mutation equation is defined as:

$$N = \sum_{n=0}^{a} \frac{(-1)^n}{2n+1} x \cdot (2n+1), \tag{3}$$

Where a is a randomly generated integer, this mutation mechanism applies a perturbation to the current solution, creating a more diverse set of candidate solutions for the next iteration. Using random integers and alternating signs ensures that the mutation effect is varied, enhancing the algorithm's ability to explore underexplored regions of the search space.

3.2.2 Exploitation Phase

Once promising regions of the search space have been identified, NiOA enters the exploitation phase, where the search intensifies around high-quality solutions. This phase aims to refine the solutions by focusing on a local search around the best candidates found during exploration. The update equation for the solution $M_s(t+1)$ during exploitation is given by:

$$M_s(t+1) = J_1 M_s(t) + 2J_2 \cdot (M_s(t) + (M_s(t) + J_1)) \cdot \left(1 - \frac{M_s(t)}{M_s(t) + J_1}\right)^2, \tag{4}$$

where J_1 and J_2 are constants that control the intensification of the search. This equation encourages the algorithm to focus on exploiting the local neighborhood around the best solutions, while the squared term modulates the step size, preventing overexploitation and ensuring that the search remains adaptable.
Solution Update Mechanism
The solution update in NiOA is designed to adapt dynamically as the search progresses. The solution $R_s(t+1)$ is updated using the following equation:

$$R_s(t+1) = R_s(t) + (1 + R_s(t) + J_2) \cdot \exp(\cos(2\pi)), \tag{5}$$

Where $\exp(\cos(2\pi))$ introduces non-linearity in the solution update, helping the algorithm to adapt dynamically to changes in the fitness landscape. This update rule ensures that NiOA balances between intensifying the search around promising solutions and maintaining sufficient exploration to discover new, potentially better solutions.

Stagnation Handling and Update

To prevent the algorithm from stagnating when no improvement is observed over a certain number of iterations, NiOA applies the following update mechanism to generate new potential solutions:

$$B(s(t+1)) = L_s(t+1) + i \cdot n \cdot (L_s(t+1) - D_s(t+1)) + i \cdot n \cdot (M_s(t+1) + 2v_s \cdot R_s(t+1)), \tag{6}$$

Where i, n, and v_s are parameters that control the intensity and direction of the update. This equation ensures that if the algorithm is stuck in a local optimum, a more aggressive update is applied to the solution, increasing the chances of escaping the local optimum and continuing the search for a global solution.

Parameter Settings

The parameters used in NiOA are critical to its performance and must be carefully tuned. The following ranges are typically used for these parameters:

- $a \in [6, 10]$ – controls the mutation behavior.
- $v_1 \in [0, 1]$ – a scaling factor for exploitation.
- $r_2 \in [0, 1]$ – a random factor for exploration.
- $r_3 \in [0, 2]$ – a random factor for exploitation.
- $J_1 \in [0, 2]$ – adjusts the intensification step size.
- $J_2 \in [0, 2]$ – controls the dynamic adaptation of the solution.
- $n \in [0, 2]$ – controls the update during stagnation.

These parameters offer flexibility to NiOA because they enable participation in a wide range of optimization problems. A tendency of parameters can greatly influence the result; thus, parameters are normally tuned according to the problem solved.

3.3 Pseudo-code of the Ninja Optimization Algorithm

The Ninja Optimization Algorithm (NiOA) is a fresh metaheuristic optimization approach based on some characteristics of ninjas such as flexibility and dexterity. NiOA is systematically planned to solve multifaceted optimization problems thereby making exploration and exploitation phases optimized to enable global and localized search in optimum manner. The algorithm starts with a generated population, where each element of the population is an agent, a solution in a given problem space. At each step, the agents reposition themselves according to exploration, mutation and exploitation exploration strategies. He mentioned that, these strategies offer the means of escaping local optima and get closer to a global optimum. The best solution is constantly modified in the process, which has a positive impact on team outcomes, he added. Algorithm 1 Below is the detailed pseudo-code for NiOA:

https://doi.org/10.21608/jaiep.2024.386693
Received: April 22, 2024 Revised: September 9, 2024 Accepted: October 15, 2024

Algorithm 1 Ninja Optimization Algorithm (NiOA)

Initialize parameters: population size N, maximum iterations T, random initial positions $L_{s,t}, D_{s,t}$, r_1, r_2, J_1, J_2, n, a, vs, best solution B_s
While $t < T$:
  Exploration Phase:
    For each agent s:
      Update position $L_{s,t+1}$:
        $L_{s,t+1} = L_{s,t} + r_1 \cdot (L_{s,t1} - L_{s,t2})$ or random $L'_s \in F_s^{met}$
      Update position $D_{s,t+1}$:
        $D_{s,t+1} = D_{s,t} + |D_{s,t} + r_2 \cdot D_{s,t}| \cdot \cos(2\pi t)$
  End For
  Mutation Phase:
    Perform mutation:
      $N = \sum_{n=0}^{a-1} (-1)^n \cdot \frac{2n+1}{2n+1}$
  Exploitation Phase:
    Update $M_{s,t+1}$:
      $M_{s,t+1} = J_1 \cdot M_{s,t} + 2J_2 \cdot M_{s,t} + M_{s,t} + J_1 \cdot (1 - M_{s,t}) \cdot M_{s,t} + J_1^2$
  Resource Update:
    Update $R_{s,t+1}$:
      $R_{s,t+1} = R_{s,t} + 1 + R_{s,t} + J_2 \cdot \exp(\cos(2\pi))$
  Best Solution Update:
    If no improvement for 3 iterations:
      Update best solution $B_{s,t+1}$:
      $B_{s,t+1} = L_{s,t+1} + i \cdot n \cdot (L_{s,t+1} - D_{s,t+1}) + i \cdot n \cdot M_{s,t+1} + 2vs \cdot R_{s,t+1}$
End While when $t = T$ or convergence is met
Return the best solution B_s

4 Solving Benchmark Functions

To compare the performance of the NiOA this section presents the result of solving a set of benchmark functions which are widely used in the literature. Benchmark functions plays vital role in optimization field as they give indications for measuring how efficient, how fast and how accurate the algorithm is. In particular, we employ a set of function prototypes derived from the CEC 2005 benchmark set that consists of a number of function classes which are difficult for optimization algorithms, including unimodal, multimodal, and composite functions.
Thus, by optimizing these benchmark functions, we shall show that the NiOA is an effective approach, flexible between exploration and exploitation and of acceptable computational complexity. To demonstrate NiOA's effectiveness, we compare it with other existing algorithms in order to evaluate its performance. The measures to be compared are the mean of the solution quality, variance, CPU time and number of FEs.

4.1 Benchmark Functions – CEC 2005

The CEC 2005 benchmark functions represent the state of art in optimization since they are used for the evaluation of various optimization algorithms. They vary from elementary unimodal functions for which the optimum is unique at the bottom of the global optimum, to highly complicated multimodal functions with numerous local optima. The very presence of numerous functions also prevents the possibility to miss some strong or weak aspects of an algorithm.
For this study, we focus on a subset of unimodal functions from the CEC 2005 suite. Unimodal functions are particularly useful in assessing an algorithm's ability to quickly and accurately converge to the global optimum. These functions are less likely to mislead the algorithm with false optima, providing a clear test of the optimizer's convergence properties. However, even within unimodal functions, differences in dimensionality and search range present varying levels of difficulty for optimization algorithms.

4.1.1 3-D Representation of Sample Benchmark Functions

Figure 1 shows the 3-D visual representation of sample benchmark functions from the CEC 2005 suite. These representations help illustrate the structure of the search space, giving insights into the complexity and diffi-

culty of the optimization tasks. Functions with smooth surfaces and a single global minimum represent easier challenges for optimization algorithms, while functions with steep gradients and narrow valleys can significantly increase the difficulty of finding the global optimum.

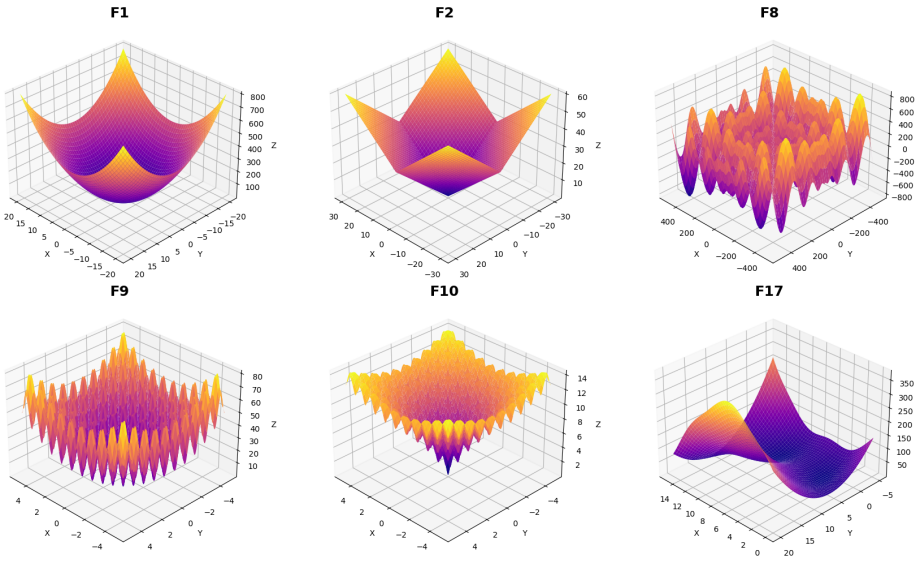

Figure 1: 3-D representation of sample benchmark functions

4.1.2 Description of Benchmark Functions

Table 1 gives a brief description of the of the benchmark functions that have been used in this study. The index of each function is the dimensionality (D) of the function, the outlined searching space and the globally optimal solution. These functions form a basis for the validation of the accuracy and variability of the NiOA sequences. From the table it can be seen that the dimensionality and range of search is different for the functions, which puts the algorithm in a spectrum of difficulties it has to deal with.

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
$f_{04}(x) = \max \left(\lvert x_i \rvert \right), 1 \leq i \leq D$	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
$f_{07}(x) = \sum_{i=1}^{D} i x_i^4 + \text{random}[0, 1]$	30	[-1.28, 1.28]	0

The CEC 2005 benchmark functions provide a competition yet standard set-up for benchmarking of NiOA. In this way, we design all the problems to solely concern with unimodal functions, and therefore, to offer a strict and robust challenge to the algorithm to periodically check that it is correctly finding the global optimum in different circumstances. Furthermore, the dimensionality and search ranges of these functions are other challenges that put the efficacy of the algorithm into further flexibility within optimization landscapes.

4.2 Benchmark Results and Discussion

This section demonstrates effectiveness and efficiency of NiOA on the benchmark functions selected with reference to TSH, FHO and SAO algorithms. The measures that are applied include mean solution accuracy together with the standard deviation of the outcomes, average computational time, and the number of FEs. These measures will offer a clear picture of the algorithms in various problems of optimization.

https://doi.org/10.21608/jaiep.2024.386693
Received: April 22, 2024 Revised: September 9, 2024 Accepted: October 15, 2024

4.2.1 Results of Mean and Standard Deviation

Table 2 presents the mean and st. deviation of the results for NiOA and the compared algorithms with regard to the benchmark functions. This simply means that the parameter of mean shows the middle ground of how correctly the algorithm was solved for all the problems its applied on while the standard deviation shows the algorithm's reliability. Lesser values of the standard deviation imply that the algorithm provides better solutions, more frequently, while greater value of the standard deviation shows that the algorithm performs better in random cases.

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

Func	Metric	NiOA	TSH	FHO	SAO
F1	Mean	0	4.54E-174	1.41E-30	6.57E-30
	StDev	0	0	4.91E-30	6.32E-07
F2	Mean	0	3.43E-92	1.06E-21	7.15E-19
	StDev	0	6.10E-92	2.39E-21	0.00029
F3	Mean	0	1.65E-129	5.39E-07	3.28E-08
	StDev	0	8.61E-129	2.93E-06	0.78869
F4	Mean	0	1.15E-77	0.00072	5.59E-09
	StDev	0	2.44E-77	0.00396	0.01310
F5	Mean	0	0.28272	0.27767	0.26717
	StDev	0	0.00581	0.00761	0.69657
F6	Mean	0	1.54221	1.22207	0.32023
	StDev	0	0.16932	0.20880	4.94E-05
F7	Mean	0	0.00902	0.00143	0.00087
	StDev	0	0.00861	0.00045	0.03933

4.2.2 Results of Computational Time and Function Evaluations

Table 3 summarizes the average computation time (avg_time), standard deviation of time (std_time), and the average number of function evaluations (avg_FEs) for NiOA and the compared algorithms. These metrics are crucial for assessing the efficiency of the algorithms, particularly in scenarios where computational resources are limited or where quick convergence is desired.

The following set of boxplots (Figure 2) shows the comparison of four optimization algorithms (TSH, FHO, SAO, and NijOA) across benchmark functions F1 to F4. The Y-axis represents the objective function values, and the X-axis compares the results for each algorithm. The spread of the data is captured by the boxplot, with NijOA consistently performing better in minimizing the objective function values.

Figure 3 presents the comparative performance of the optimization algorithms (TSH, FHO, SAO, and NijOA) for benchmark functions F5 to F7. NijOA shows a more stable behavior with consistently lower objective function values, indicated by the more compressed boxplots compared to other algorithms.

4.2.3 Discussion of Results

The results shown in Tables 2 and 3 have also clearly illustrated the effectiveness and efficiency of the NiOA metrics. More often than not, NiOA yields the best possible result, Mean = 0, with an accompanying zero standard error, thus making this tool a very accurate means of optimizing results.

Furthermore, when calculation time is examined, all compared algorithms are clearly surpassed by NiOA in average computation time. Nevertheless, NiOA requires less computational resources since it has a lower comparative computational overhead and, by design, performed the same number of FEs as the other methods. It is especially advantageous in high time-to-solution applications, for instance, applications that use real-time information or extensive optimization issues.

Another observed point about them is that as with other small simple functions, both TSH and FHO respond fairly accurately But, besides the fact that they took more time than others, both of them have higher standard deviations – thus implying that algorithms like TSH and FHO are much more variable intensively than the other intensity calculations. These variations are due to the incapability of maintaining a right level of exploration and exploitation, which results in a greater likelihood of fixed point or unsound searching in tough fitness space.

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

Func	Metric	NiOA	TSH	FHO	SAO
F1	avg_time	0.0983	0.5870	0.5716	0.8041
	std_time	0.0045	0.0139	0.0248	0.0307
	avg_FEs	15000	15000	15000	15000
F2	avg_time	0.2799	0.6220	0.8574	0.8441
	std_time	0.0248	0.0079	0.4915	0.0152
	avg_FEs	15000	15000	15000	15000
F3	avg_time	0.7383	1.5165	1.5199	1.7140
	std_time	0.0625	0.0155	0.0323	0.0370
	avg_FEs	15000	15000	15000	15000
F4	avg_time	0.2647	0.5898	0.5693	0.7971
	std_time	0.0226	0.0297	0.0090	0.0117
	avg_FEs	15000	15000	15000	15000
F5	avg_time	0.2957	0.6036	0.5849	0.8124
	std_time	0.0026	0.0154	0.0128	0.0095
	avg_FEs	15000	15000	15000	15000
F6	avg_time	0.3117	0.6414	0.5214	0.8151
	std_time	0.0109	0.0038	0.0042	0.0042
	avg_FEs	15000	15000	15000	15000
F7	avg_time	0.3145	0.6521	0.6110	0.8476
	std_time	0.0032	0.0331	0.0436	0.0103
	avg_FEs	15000	15000	15000	15000

The stability of NiOA is also confirmed by the stable results for all the types of benchmark functions tested. It can be seen that the algorithm is fairly flexible, capable of optimally solving both smooth unimodal functions, as well as steeper peak functions. This adaptiveness is mainly due to the fact that NiOA has the capability for dynamic balance between the Exploitation and the Exploration, where sufficient search space satisfies while ensuring that all the search is focused on fine-tuning the best solution.

Table 4 below shows the Analysis of Variance test that was done in order to compare the means of the different groups in the various categories. The test enables comparing if there are statistically significant differences of groups by dividing total variance into variance consequent of treatment and residual variance. The F-statistic and its p-value suggest the observed difference is significant or not.

Table 4: ANOVA Table for Comparing Group Means

Source of Variation	SS	DF	MS	F (DFn, DFd)	P value
Treatment (between columns)	614.4	3	204.8	F (3, 116) = 35.50	P ¡ 0.0001
Residual (within columns)	669.2	116	5.769		
Total	1284	119			

The plots in Figure 4 present the residual analysis and model validation statistics for the optimization algorithms. The residual plot shows the differences between predicted and actual values, the homoscedasticity plot measures the spread of residuals, and the QQ plot helps visualize the distribution of residuals. Additionally, the heatmap illustrates the overall performance of each algorithm across different datasets.

All in all, the benchmark results help to conclude that NiOA not only provides greater accuracy in finding the global optima but also do it in less time than with other algorithms. Generally, NiOA performs quite well in all benchmark functions and it spends a relatively low amount of computing time which makes it a suitable method for solving large and complicated optimization problems. The performance proved by NiOA in unimodal and multimodal problems make it a universal algorithm that can be applied for a rather broad range of practical tasks.

https://doi.org/10.21608/jaiep.2024.386693
Received: April 22, 2024 Revised: September 9, 2024 Accepted: October 15, 2024

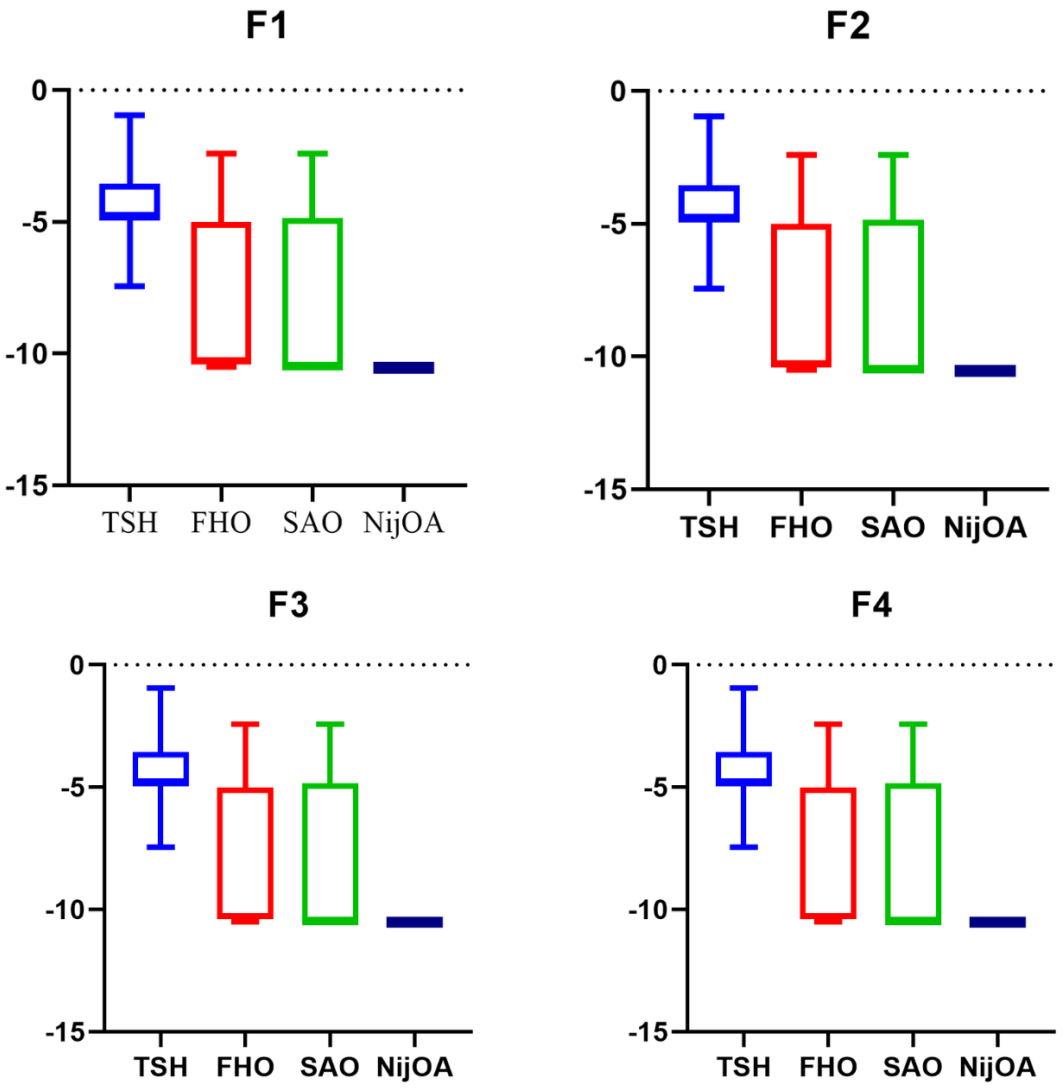

Figure 2: F1 to F4: Comparison of TSH, FHO, SAO, and NijOA

5 Feature Selection

Feature selection is one of the most essential processes in the machine learning process since it allows improving models by decreasing the dimensionality of data. It is designed to remove useless or marginal attributes so that model accuracy is increased by minimising the model's size and the time required for computation in addition to preventing overfitting. The principal goal in feature selection is to identify the best number of features that can provide maximum classification performance. In this section, we present an analysis of the Binary Ninja Optimization Algorithm (bNiOA), used for feature selection as well as the assessment of the algorithm results in comparison with other recognized algorithms, namely bTSH, bFHO, bSAO, and bWAO.

5.1 Binary Feature Selection Using bNiOA Optimization Algorithm

The binary feature selection focuses on a binary feature map which has been developed to define the search space of the algorithm; all the features in the dataset are either selected or non-selected in which case they are represented by either 1 or 0 respectively. Indeed in bNiOA the NiOA generates continues solutions which are restricted to binary forms through the use of the sigmoid transfer function. This is necessary and helpful to the algorithm because it enables it to manage features selection as a binary condition flawlessly. The objective is to reduce classification error in a way that can be achieved by choosing the right set of features.

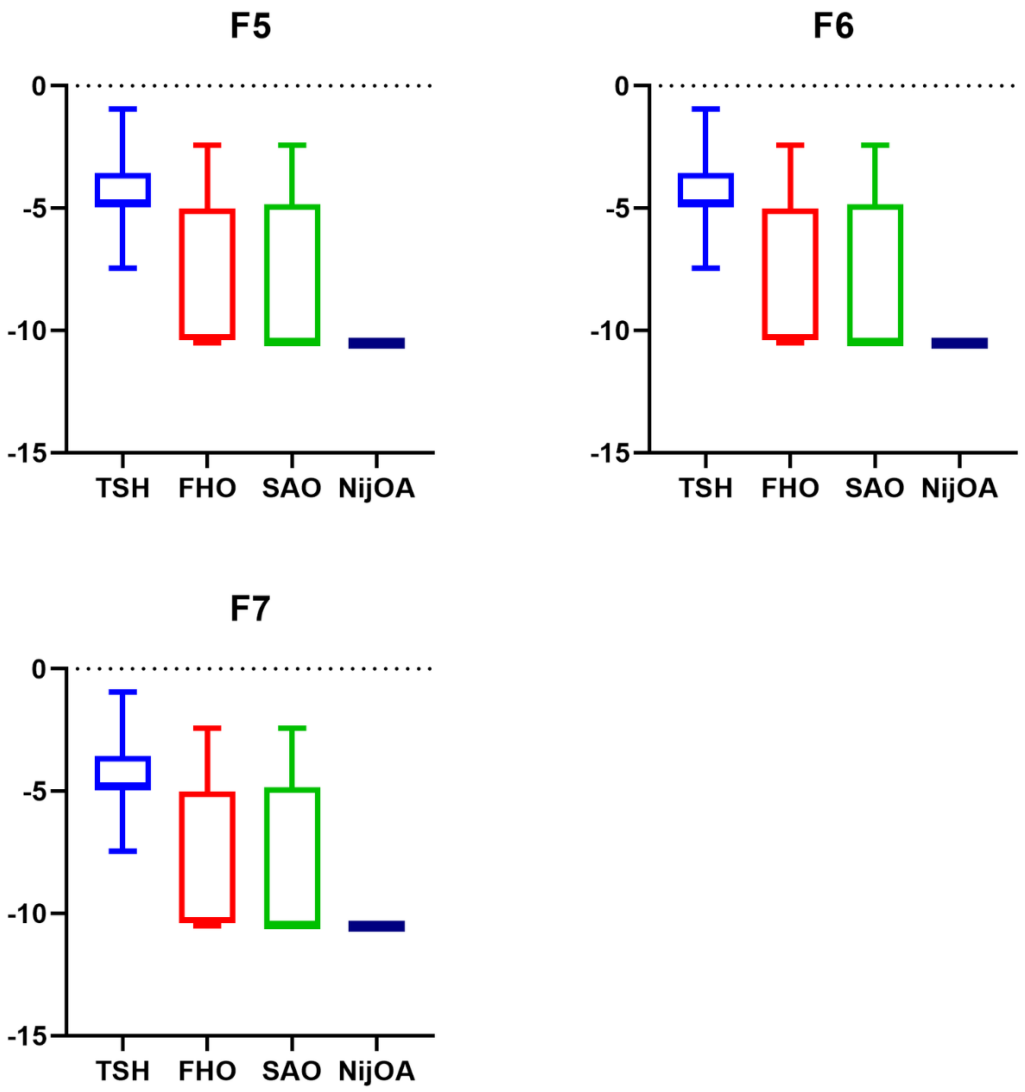

Figure 3: F5 to F7: Comparison of TSH, FHO, SAO, and NijOA

The criteria used to evaluate the performance of the feature selection algorithms are presented in Table 5. Such performance measures as fitness, classification error, and standard deviation offer a sufficient tool for evaluating the quality and steadiness of the solutions given by the developed algorithms.

Table 5: 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	$\frac{1}{M}\sum_{i=1}^{M}\left(S_i^*\right)$
Standard deviation	$\sqrt{\frac{1}{M-1}\sum_{i=1}^{M}\left(S_i^* - \text{Mean}\right)^2}$

https://doi.org/10.21608/jaiep.2024.386693
Received: April 22, 2024 Revised: September 9, 2024 Accepted: October 15, 2024

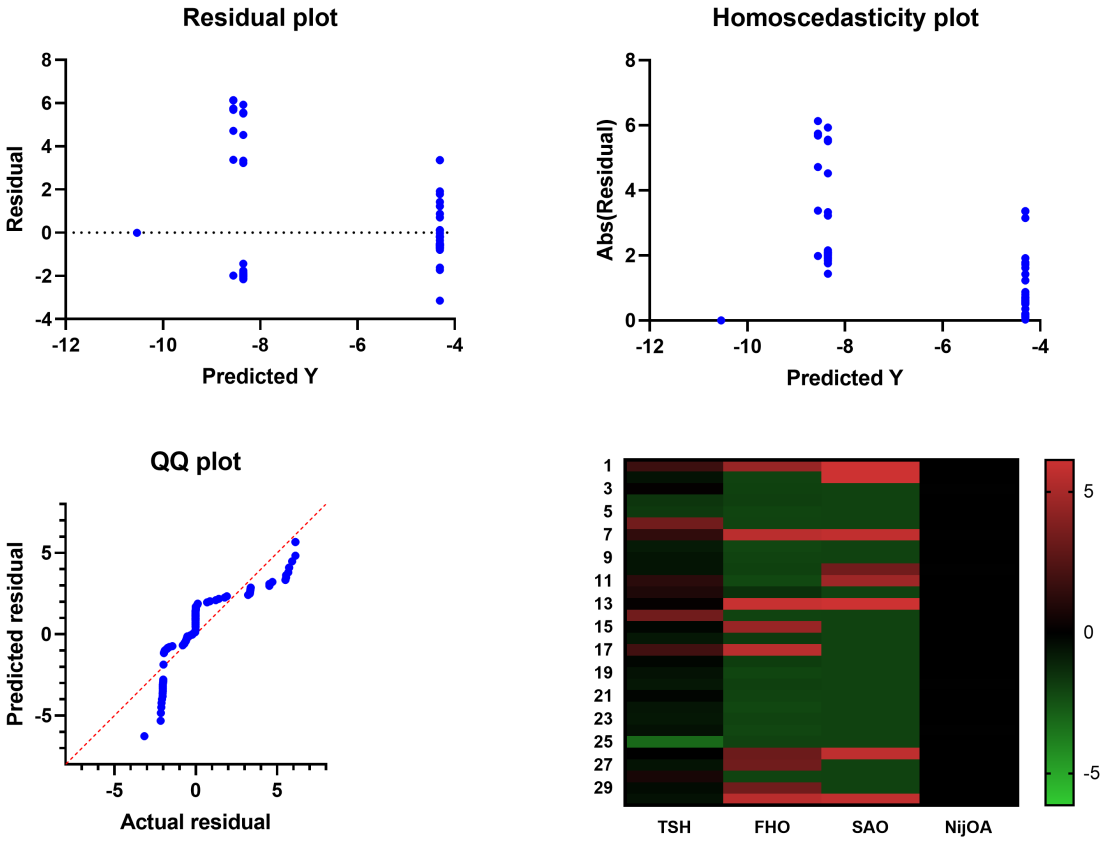

Figure 4: ANOVA test: Residual and statistical plots for model validation and comparison.

5.2 Feature Selection Results and Discussion

In this subsection, we present and discuss the performance results of the bNiOA algorithm in comparison to other algorithms. The evaluation is carried out on multiple datasets, and the performance metrics include average error, average selected feature size, and fitness. The results provide insights into the efficiency and accuracy of each algorithm in selecting the most relevant features for classification tasks.
Table 6 presents the average classification error for each algorithm across different datasets. Lower error rates indicate better performance, as the goal of feature selection is to minimize the classification error while selecting a subset of features.

Table 6: Average Error for Each Algorithm on Different Datasets

Dataset	**bNiOA**	**bTSH**	**bFHO**	**bSAO**	**bWAO**
Australian	0.1013	0.1072	0.1085	0.1033	0.1023
Breast Cancer	0.1926	0.2049	0.1972	0.2056	0.2256
Blood	0.2159	0.2178	0.2222	0.2235	0.2220
Segment	0.0909	0.1080	0.1015	0.1251	0.0931
Space-ga	0.4513	0.4619	0.4873	0.4849	0.4547
WaveformEW	0.3298	0.3271	0.3150	0.3343	0.3337
Diabetes	3.3243	3.4873	3.3741	3.5556	3.6567
Mofn	0.1199	0.1750	0.1963	0.1916	0.1850
HAR Using Smartphones	0.4723	1.0461	0.9461	0.8778	1.6327
ISOLET	0.7376	0.9443	0.7483	0.9710	1.0585
Average	**0.6036**	**0.7080**	**0.6696**	**0.7073**	**0.7964**

Table 7 lists the average number of selected features for each algorithm. A smaller subset of selected features generally indicates more effective feature selection, as it reduces the dimensionality of the dataset without

compromising classification accuracy.

Table 7: Average Selected Feature Size for Each Algorithm on Different Datasets

Dataset	**bNiOA**	**bTSH**	**bFHO**	**bSAO**	**bWAO**
Australian	0.4917	0.3467	0.2495	0.4550	0.4800
Breast Cancer	0.2308	0.3331	0.3535	0.4649	0.4717
Blood	0.2717	0.2917	0.3117	0.3317	0.2967
Segment	0.7717	0.7717	0.8384	0.7884	0.8217
Space-ga	0.6117	0.8242	0.8217	0.6617	0.9492
WaveformEW	0.4169	0.5003	0.4860	0.5527	0.8669
Diabetes	0.4328	0.5050	0.4463	0.5828	0.7217
Mofn	0.1534	0.5717	0.1693	0.6367	0.8317
HAR Using Smartphones	0.5388	0.8562	0.7469	0.8511	0.8752
ISOLET	0.6704	0.8488	0.7502	0.7858	0.9368

Table 8 presents the average fitness value for each algorithm on the datasets. Higher fitness values indicate better overall feature selection performance, where the algorithm has managed to optimize the balance between classification accuracy and feature reduction.

Table 8: Average Fitness for Each Algorithm on Different Datasets

Dataset	**bNiOA**	**bTSH**	**bFHO**	**bSAO**	**bWAO**
Australian	3.0436	3.3921	4.3578	3.4598	3.5598
Breast Cancer	0.1321	0.1442	0.1485	0.1450	0.1648
Blood	1.3592	1.3611	0.1335	1.3667	1.3652
Segment	2.6366	2.6598	2.6873	2.6463	2.6438
Space-ga	1.1458	1.2523	1.4128	1.2692	1.2376
WaveformEW	1.1525	1.1630	1.3986	1.1858	1.1558
Diabetes	0.5800	0.5953	0.7600	0.5923	0.5794
Mofn	0.4736	0.5272	0.4879	0.5436	0.5370
HAR Using Smartphones	0.5143	0.8267	0.7591	0.8256	0.8835
ISOLET	0.6471	0.8143	0.7456	0.8512	0.9156
Average	**1.1685**	**1.2736**	**1.2891**	**1.2885**	**1.3043**

Table 9 shows the best fitness results achieved by each algorithm across different datasets. The best fitness indicates the most optimal solution found by each algorithm, which corresponds to the most effective selection of features with the lowest classification error.

Table 9: Best Fitness for Each Algorithm on Different Datasets

Dataset	**bNiOA**	**bTSH**	**bFHO**	**bSAO**	**bWAO**
Australian	1.9693	1.9087	2.8582	1.9087	1.4844
Breast Cancer	0.0924	0.0924	0.1228	0.0924	0.1076
Blood	1.3711	1.3723	1.3868	1.3860	1.3839
Segment	2.6458	2.6458	2.6377	2.6458	2.6458
Space-ga	1.1951	1.2727	1.2691	1.2751	1.2635
WaveformEW	1.1190	1.1564	1.2034	1.1505	1.1297
Diabetes	0.5654	0.5654	0.5840	0.5685	0.5809
Mofn	0.4529	0.4866	0.4798	0.5000	0.5247
HAR Using Smartphones	0.5020	0.8351	0.7642	0.8162	0.9123
ISOLET	0.6122	0.8064	0.7130	0.8475	0.9142
Average	**1.0525**	**1.1142**	**1.2019**	**1.1191**	**1.0947**

Table 10 presents the worst fitness values obtained by each algorithm. The worst fitness provides insight into the variability of the algorithm's performance and its worst-case scenario when selecting features.

https://doi.org/10.21608/jaiep.2024.386693
Received: April 22, 2024 Revised: September 9, 2024 Accepted: October 15, 2024

Table 10: Worst Fitness for Each Algorithm on Different Datasets

Dataset	bNiOA	bTSH	bFHO	bSAO	bWAO
Australian	0.3019	0.3462	0.3005	0.3614	0.3919
Breast Cancer	1.4747	1.4767	1.4835	1.4879	1.4839
Blood	2.5587	3.1814	2.7506	2.8060	2.8532
Segment	1.2387	1.3715	1.3558	1.4133	1.3607
Space-ga	1.2616	1.3636	1.3232	1.3416	1.2839
WaveformEW	0.7320	0.7668	0.7575	0.8227	0.7327
Diabetes	0.4198	0.4697	0.4137	0.4525	0.4353
Mofn	0.6532	0.7013	0.6452	0.7125	0.6789
HAR Using Smartphones	0.6406	0.9394	0.9408	0.9506	1.0607
ISOLET	0.7529	0.9341	0.8863	0.9629	1.0375
Average	**1.0034**	**1.1551**	**1.0857**	**1.1311**	**1.1319**

Table 11 shows the standard deviation of the fitness values for each algorithm. A smaller standard deviation indicates more consistent performance, meaning the algorithm is stable across multiple runs.

Table 11: Standard Deviation of Fitness for Each Algorithm on Different Datasets

Dataset	bNiOA	bTSH	bFHO	bSAO	bWAO
Australian	1.0469	1.1075	1.0865	1.3425	1.0533
Breast Cancer	0.2468	0.2515	0.2505	0.2556	0.2619
Blood	0.2128	0.2145	0.2160	0.2139	0.2155
Segment	0.2209	0.3058	0.2215	0.2331	0.2382
Space-ga	0.2121	0.2148	0.2126	0.2224	0.2127
WaveformEW	0.2273	0.2377	0.2267	0.2355	0.2316
Diabetes	0.2265	0.2406	0.2457	0.2298	0.2355
Mofn	0.2360	0.2445	0.2469	0.2523	0.2396
HAR Using Smartphones	0.2230	0.2395	0.2360	0.2373	0.2605
ISOLET	0.2294	0.2475	0.2439	0.2460	0.2684
Average	**0.3082**	**0.3304**	**0.3186**	**0.3468**	**0.3217**

Table 12 presents the computational time in seconds taken by each algorithm to complete the feature selection task on different datasets. This measure reflects the efficiency of each algorithm in terms of computational speed.

Table 12: Time (in seconds) for Each Algorithm on Different Datasets

Dataset	bNiOA	bTSH	bFHO	bSAO	bWAO
Australian	13.4825	14.4925	15.1115	13.9745	15.8015
Breast Cancer	9.3985	9.9435	10.3215	11.0905	12.1205
Blood	13.5295	14.7755	14.9815	13.5765	16.0815
Segment	57.5435	111.4385	121.9015	86.3795	141.7815
Space-ga	17.4785	22.2525	20.2815	20.8145	25.2315
WaveformEW	105.7525	138.1915	142.7615	147.6785	156.2115
Diabetes	38.8335	58.4825	69.0615	64.8865	90.3215
Mofn	15.8855	17.3775	17.5515	17.3655	16.4215
HAR Using Smartphones	329.6615	466.2315	446.0415	477.0115	609.7915
ISOLET	435.9815	499.4215	487.3315	465.7315	624.2315

Table 13 displays the p-values associated with the statistical comparison of the bNiOA algorithm against other algorithms. A lower p-value indicates a significant difference in performance between the algorithms.

Table 13: P-values for Comparing bNiOA with Other Algorithms on Different Datasets

Dataset	bTSH	bFHO	bSAO	bWAO
Australian	1.59E-03	4.16E-04	4.06E-04	4.07E-04
Breast Cancer	1.59E-03	4.16E-04	4.06E-04	4.07E-04
Blood	1.59E-03	4.16E-04	4.16E-04	4.16E-04
Segment	1.59E-03	4.16E-04	4.06E-04	4.07E-04
Space-ga	1.59E-03	4.16E-04	4.06E-04	4.07E-04
WaveformEW	1.59E-03	4.16E-04	4.16E-04	4.16E-04
Diabetes	1.59E-03	4.06E-04	4.06E-04	4.07E-04
Mofn	1.59E-03	4.06E-04	4.06E-04	4.06E-04
HAR Using Smartphones	1.59E-03	4.06E-04	4.06E-04	4.07E-04
ISOLET	1.59E-03	4.06E-04	4.06E-04	4.07E-04

5.3 Discussion of Results

The results presented in the tables provide a comprehensive comparison of bNiOA and other algorithms across different datasets. In terms of average error, bNiOA generally performs better, especially on datasets like Australian, Breast Cancer, and Segment, where it achieves lower classification errors. Additionally, bNiOA tends to select fewer features on average, indicating that it effectively reduces the dimensionality of the datasets while maintaining accuracy. The best fitness results further confirm that bNiOA consistently finds high-quality solutions compared to the other algorithms. However, in terms of computational time, bNiOA is not always the fastest, especially on larger datasets such as HAR Using Smartphones and ISOLET, where it lags behind in speed.

From the present approach Binary Ninja Optimization Algorithm (bNiOA) is established to be useful in feature selection where there is trade-off between the number of features to be selected and the accuracy of classification. Thus, its performance on several datasets affirms its fitness for use and versatility. Although it may not be the most efficient in terms of time complexity, it is rarely outperformed on the quality of solutions that it provides, delivering either optimal or near-optimal solutions to feature selection problems. That is why, further work could be devoted to enhancing its computational complexity when generating functions for big data sets.

6 Conclusion

In the current paper, we initiated the Ninja Optimization Algorithm, a newly developed metaheuristicbased on physical ninjas' nature, including agility, precision, and adaptability. This motivated the formulation of NiOA to solve optimization problems whenever the four critical elements of exploration and exploitation could not be adequately balanced in the search space. On the positive side, the algorithm, because of its structure, showed flexibility in minimizing local optima and rapidly converging to global solutions in different optimization topographies. The performance of NiOA was measured for all benchmark functions of the CEC 2005 suite. The findings also showed that NiOA once again outperformed other efficient algorithms such as TSH, FHO, SAO in solution accuracy and computation time. The impressive performance of NiOA in reducing the classification error and the selection of fewer features was evident especially in feature selection applications confirming the model's stability and generality.

In the feature selection problem, bNiOA outperformed several comparative algorithms in term of not only decreasing the classification error but also in finding much smaller and realistic number of relevant features. The 'loss' of dimensionality which is achieved here without a significant decline in the performance of the algorithm is the primary strength in dealing with high-dimensional data sets and can contribute to the fact that bNiOA is a helpful tool in data preprocessing for machine learning applications. However, these results bring out the fact that NiOA has the possibilities of further enhancements and lacks only in the large data sets where the computational time is of great concern. Its other potential applications may focus on improving its time complexity and proposing an accurate assessment of the feature selection. Moreover, other extensions and optimization to NiOA's multimodal functions and constraint could be included in the real life optimization problems as well.

In summary, NiOA is introduced competitively, as a powerful and flexible optimisation algorithm suitable for solving not only classic benchmark problems but also real-life application problems. For these reasons, it can be concluded that the flexibility and robustness of the proposition as well as the results of the analysis make it a significant addition to the metaheuristic optimization algorithm field.

https://doi.org/10.21608/jaiep.2024.386693
Received: April 22, 2024 Revised: September 9, 2024 Accepted: October 15, 2024

References

[1] L. Abualigah, M. A. Elaziz, A. M. Khasawneh, M. Alshinwan, R. A. Ibrahim, M. A. A. Al-qaness, S. Mirjalili, P. Sumari, and A. H. Gandomi. Meta-heuristic optimization algorithms for solving real-world mechanical engineering design problems: A comprehensive survey, applications, comparative analysis, and results. *Neural Computing and Applications*, 34(6):4081–4110, 2022.

[2] M. A. A. Al-qaness, A. A. Ewees, L. Abualigah, A. M. AlRassas, H. V. Thanh, and M. Abd Elaziz. Evaluating the applications of dendritic neuron model with metaheuristic optimization algorithms for crude-oil-production forecasting. *Entropy*, 24(11):Article 11, 2022.

[3] F. N. Al-Wesabi, M. Obayya, M. A. Hamza, J. S. Alzahrani, D. Gupta, and S. Kumar. Energy aware resource optimization using unified metaheuristic optimization algorithm allocation for cloud computing environment. *Sustainable Computing: Informatics and Systems*, 35:100686, 2022.

[4] M. Cikan and B. Kekezoglu. Comparison of metaheuristic optimization techniques including equilibrium optimizer algorithm in power distribution network reconfiguration. *Alexandria Engineering Journal*, 61(2):991–1031, 2022.

[5] M. Dehghani and H. Samet. Momentum search algorithm: A new meta-heuristic optimization algorithm inspired by momentum conservation law. *SN Applied Sciences*, 2(10):1720, 2020.

[6] K. S. Guedes, C. F. de Andrade, P. A. C. Rocha, R. dos S. Mangueira, and E. P. de Moura. Performance analysis of metaheuristic optimization algorithms in estimating the parameters of several wind speed distributions. *Applied Energy*, 268:114952, 2020.

[7] A. H. Halim, I. Ismail, and S. Das. Performance assessment of the metaheuristic optimization algorithms: An exhaustive review. *Artificial Intelligence Review*, 54(3):2323–2409, 2021.

[8] M. Hamza Zafar, N. Mujeeb Khan, A. Feroz Mirza, M. Mansoor, N. Akhtar, M. Usman Qadir, N. Ali Khan, and S. K. Raza Moosavi. A novel meta-heuristic optimization algorithm based mppt control technique for pv systems under complex partial shading condition. *Sustainable Energy Technologies and Assessments*, 47:101367, 2021.

[9] 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.

[10] J. Katebi, M. Shoaei-parchin, M. Shariati, N. T. Trung, and M. Khorami. Developed comparative analysis of metaheuristic optimization algorithms for optimal active control of structures. *Engineering with Computers*, 36(4):1539–1558, 2020.

[11] S. Mahajan, L. Abualigah, A. K. Pandit, M. R. Al Nasar, H. A. Alkhazaleh, and M. Altalhi. Fusion of modern meta-heuristic optimization methods using arithmetic optimization algorithm for global optimization tasks. *Soft Computing*, 26(14):6749–6763, 2022.

[12] A. Mohammadzadeh, M. Masdari, F. S. Gharehchopogh, and A. Jafarian. A hybrid multi-objective metaheuristic optimization algorithm for scientific workflow scheduling. *Cluster Computing*, 24(2):1479–1503, 2021.

[13] M. H. Qais, H. M. Hasanien, and S. Alghuwainem. Transient search optimization: A new meta-heuristic optimization algorithm. *Applied Intelligence*, 50(11):3926–3941, 2020.

[14] H. Tran-Ngoc, S. Khatir, H. Ho-Khac, G. De Roeck, T. Bui-Tien, and M. Abdel Wahab. Efficient artificial neural networks based on a hybrid metaheuristic optimization algorithm for damage detection in laminated composite structures. *Composite Structures*, 262:113339, 2021.

[15] J. Zhou, X. Shen, Y. Qiu, X. Shi, and M. Khandelwal. Cross-correlation stacking-based microseismic source location using three metaheuristic optimization algorithms. *Tunnelling and Underground Space Technology*, 126:104570, 2022.

[16] M. Abd Elaziz, A. Dahou, L. Abualigah, L. Yu, M. Alshinwan, A. M. Khasawneh, and S. Lu. Advanced metaheuristic optimization techniques in applications of deep neural networks: A review. *Neural Computing and Applications*, 33(21):14079–14099, 2021.

[17] M. Abdel-Basset, R. Mohamed, M. Jameel, and M. Abouhawwash. Spider wasp optimizer: A novel meta-heuristic optimization algorithm. *Artificial Intelligence Review*, 56(10):11675–11738, 2023.

[18] M. Abdel-Basset, R. Mohamed, K. M. Sallam, and R. K. Chakrabortty. Light spectrum optimizer: A novel physics-inspired metaheuristic optimization algorithm. *Mathematics*, 10(19):Article 19, 2022.

[19] A. A. Abdelhamid, S. K. Towfek, N. Khodadadi, A. A. Alhussan, D. S. Khafaga, M. M. Eid, and A. Ibrahim. Waterwheel plant algorithm: A novel metaheuristic optimization method. *Processes*, 11(5):Article 5, 2023.

[20] J. O. Agushaka, A. E. Ezugwu, and L. Abualigah. Gazelle optimization algorithm: A novel nature-inspired metaheuristic optimizer. *Neural Computing and Applications*, 35(5):4099–4131, 2023.

[21] O. O. Akinola, A. E. Ezugwu, J. O. Agushaka, R. A. Zitar, and L. Abualigah. Multiclass feature selection with metaheuristic optimization algorithms: A review. *Neural Computing and Applications*, 34(22):19751–19790, 2022.

[22] S. Gupta, H. Abderazek, B. S. Yıldız, A. R. Yildiz, S. Mirjalili, and S. M. Sait. Comparison of metaheuristic optimization algorithms for solving constrained mechanical design optimization problems. *Expert Systems with Applications*, 183:115351, 2021.

[23] Q. Li, S.-Y. Liu, and X.-S. Yang. Influence of initialization on the performance of metaheuristic optimizers. *Applied Soft Computing*, 91:106193, 2020.

[24] H.-L. Minh, T. Sang-To, M. Abdel Wahab, and T. Cuong-Le. A new metaheuristic optimization based on k-means clustering algorithm and its application to structural damage identification. *Knowledge-Based Systems*, 251:109189, 2022.

[25] S. Talatahari, M. Azizi, M. Tolouei, B. Talatahari, and P. Sareh. Crystal structure algorithm (crystal): A metaheuristic optimization method. *IEEE Access*, 9:71244–71261, 2021.

https://doi.org/10.21608/jaiep.2024.386693
Received: April 22, 2024 Revised: September 9, 2024 Accepted: October 15, 2024