Artificial Intelligence Review (2024) 57:194
https://doi.org/10.1007/s10462-024-10829-9

A comprehensive study on modern optimization techniques for engineering applications

Shitharth Selvarajan[1,2]

Accepted: 11 June 2024

Abstract
Rapid industrialization has fueled the need for effective optimization solutions, which has led to the widespread use of meta-heuristic algorithms. Among the repertoire of over 600, over 300 new methodologies have been developed in the last ten years. This increase highlights the need for a sophisticated grasp of these novel methods. The use of biological and natural phenomena to inform meta-heuristic optimization strategies has seen a paradigm shift in recent years. The observed trend indicates an increasing acknowledgement of the effectiveness of bio-inspired methodologies in tackling intricate engineering problems, providing solutions that exhibit rapid convergence rates and unmatched fitness scores. This study thoroughly examines the latest advancements in bio-inspired optimisation techniques. This work investigates each method's unique characteristics, optimization properties, and operational paradigms to determine how revolutionary these approaches could be for problem-solving paradigms. Additionally, extensive comparative analyses against conventional benchmarks, such as metrics such as search history, trajectory plots, and fitness functions, are conducted to elucidate the superiority of these new approaches. Our findings demonstrate the revolutionary potential of bio-inspired optimizers and provide new directions for future research to refine and expand upon these intriguing methodologies. Our survey could be a lighthouse, guiding scientists towards innovative solutions rooted in various natural mechanisms.

Keywords Bio-inspired meta-heuristic technique · Feature selection · Benchmark test problems · Optimization · Engineering problems

1 Introduction

These days, it's common to hear the term "meta," which means "deeper" or "notably greater level." The ongoing improvement of heuristic algorithms often gets referred to as "metaheuristics" despite the lack of a unified scientific definition. A heuristic

✉ Shitharth Selvarajan
ShitharthS@kdu.edu.et

[1] Department of Computer Science, Kebri Dehar University, 250 Kebri Dehar, Ethiopia

[2] School of Built Environment, Engineering and Computing, Leeds Beckett University, Leeds LS1 3HE, UK

Published online: 04 July 2024

algorithm is a technique that uses trial and error methods to generate feasible solutions for optimization problems with the environment (Feda et al. 2024; Karimzadeh Parizi et al. 2020). Not only can living things, organisms, and microscopic parts of the environment, including honeybees and ants, retain knowledge but so can humankind. Several meta-heuristics, often known as nature-inspired methods, draw motivation from nature. Intelligent algorithms have been developed to address real-world issues and are modelled after biological and natural laws. They possess the characteristics of a simple idea and practical execution and offer complex tasks for people to do in a group for collaborative execution. Intelligent algorithms (Parizi et al. 2021; Karimzadeh Parizi et al. 2021; Zhong et al. 2023) have grown to be a prominent topic for research because of their widespread use in computing, discovering data, communications networks, and time forecasting, as well as their propagation, brevity, adaptability, and endurance.

Optimization is essential for solving complex engineering problems (Osaba et al. 2020; Adegboye et al. 2024a; Karimzadeh Parizi and Keynia 2021) by providing suitable and intelligent solutions—optimization methodologies. The optimization methodologies (Ser et al. 2019) are extensively used in all application systems, offering the best solutions to specific issues. According to recent reviews, it is analyzed that more than 150 different types (Datta et al. 2019) of optimization techniques are used in real-time application systems. Typically, feature analysis is mainly performed to solve the given problem by determining the minimum and maximum values, and the obtained solution is termed the objective function (Yang and Shami 2020; Adegboye et al. 2024b; Adegboye and Deniz Ülker 2023). As shown in Fig. 1, the meta-heuristics optimization techniques are categorized into the following types:

- Evolutionary-based
- Physics-based
- Swarm-based
- Bio-inspired

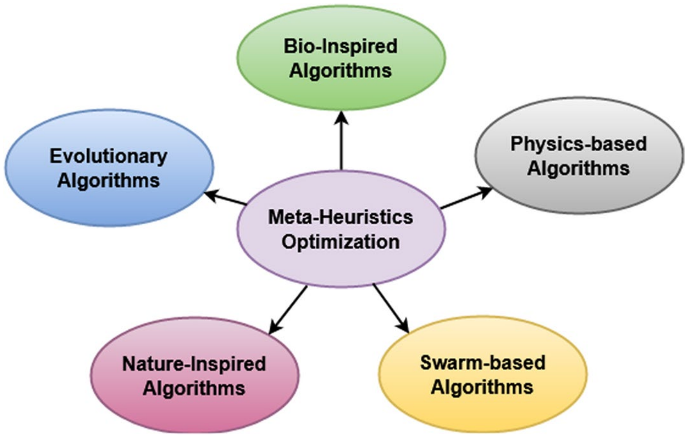


Fig. 1 Different types of optimization techniques

- Nature-inspired

The primary purpose of this paper is to investigate the different types of methodologies in bio-inspired optimization.

Due to the increased complexity of real-world problems, meta-heuristics (Abualigah et al. 2021) optimization techniques have gained more attraction recently. The optimization methodology is mainly used to obtain possible solutions for the given problems. Population-based meta-heuristics are developed from biological evolution operations such as recombination, selection, regeneration, etc. The most popular evolutionary algorithms (Slowik and Kwasnicka 2020) are Genetic Programming (GP) (Kumar et al. 2018), Evolutionary Programming (EP) (Banerjee and Mitra 2020), Differential Evolution (Song et al. 2023), Genetic Algorithms (GA) (Ramos-Figueroa et al. 2020), and Evolution Strategies (ES) (Ahrari and Essam 2022). Many engineering systems increasingly use bio-inspired meta-heuristic techniques to solve complex problems. They are easy to implement, have minimal computational complexity, and have an optimum function with reduced iterations. In physics-based algorithms, the communication between the searching agents and searching space is accomplished based on the physical characteristics (Pereira et al. 2021) of gravitational force, inertia weight, electromagnetic force (EMF), etc. For instance, the gravitational search and simulated annealing are physics-based algorithms. The optimal solution is obtained according to the law of gravity and mass interactions. Then, the swarm-based algorithms are developed based on the social behaviour analysis of natural colonies and herds. It includes the techniques of Particle Swarm Optimization (PSO) (Band et al. 2020), Bat Algorithm (Wang et al. 2019), Bee Colony Optimization (BCO) (Teodorović et al. 2021), Grasshopper Optimization (GO) (Aydogdu et al. 2022), Artificial Bee Colony (ABC) (Karaboga and Aslan 2019), Ant Colony Optimization (ACO) (Khan and Tiziano 2018), Firefly (FF) optimization (Hassan 2021), Cuckoo Search Optimization (CSO) (Yu et al. 2020), Salp-Swarm Optimization (SSO) (Castelli et al. 2022), etc. The main contribution of this paper is to examine the different types of bio-inspired optimization techniques with their working model, characteristics, and operating principles. The objectives behind this analysis are as follows:

- To evaluate the importance of applying optimization methods to tackle urgent engineering challenges.
- To examine the operational features, benefits, and drawbacks of the many bio-inspired meta-heuristic optimization approaches that have been employed recently.
- To thoroughly examine the optimization process, including an overview of its features and significance.
- This study uses standard benchmarking functions to assess the efficacy and outcomes of contemporary bio-inspired optimization strategies.

The remaining portions of this paper are segregated into the following: Section 2 presents the complete analysis of the list of bio-inspired optimization techniques with its separate algorithms, mathematical models, and workflow. Section 3 evaluates and compares the results of the recent optimization techniques using the standard benchmarking functions, which validates the performance in terms of the fitness function, objective space, searching and trajectory patterns. Finally, Section 4 summarises the overall review, including its obtainments and future scope.

2 Literature review

In bio-inspired optimization, many systematic reviews have been done to document the advancement and use of different algorithms. The critical reviews in the literature are as follows: Bio-inspired algorithms have been explored in numerous comprehensive reviews, which concentrate on their principles, uses, and performance measurements. Such overviews usually involve the examination of established techniques like Genetic Algorithms (GA), Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO). They give a wide-ranging understanding about how these methods work and where they can be applied. Some reviews concentrate on how bio-inspired algorithms are used in certain areas, like engineering optimization, network design, and scheduling problems. These kinds of reviews typically analyze the effectiveness of particular algorithms within specific surroundings and recognize challenges and solutions relevant to a particular domain. Specific reviews have compared bio-inspired algorithms, measuring them against usual benchmarks and performance measures. These kinds of analyses usually try to find out which algorithms work best in specific situations, offering helpful guidance for those who use them. Some of the latest reviews discuss more modern and less traditional bio-inspired methods. The usual focus is on algorithms from the previous ten years, looking at their newness, application possibilities, and initial outcomes in performance. Despite the extensive work done in previous systematic reviews, several gaps and limitations remain: Certain reviews may be old and not reflect the most recent progress in algorithms. This can mislead readers about the current state of affairs. Occasionally, a few reviews assume prior knowledge on the part of their reader. This can make it difficult for new people to understand certain aspects. Reviews with a narrow perspective: Some reviews might not consider various viewpoints or could be biased towards specific methods, thus limiting their usefulness. This can result in an incomplete assessment. Problematic Evaluation Procedures: Assessments could sometimes contain methodological flaws like imperfect trial setups or inappropriate data sets, which might influence final results unpredictably. Frequently, technical language is used excessively in evaluations, making it hard for readers without advanced understanding to grasp the main points being communicated. Sometimes, appraisals fail to provide practical illustrations regarding where and how particular algorithms are applied. This can prevent readers from fully comprehending their significance and usefulness outside academia or research settings.

Most of the time, evaluations concentrate on established algorithms and need to adequately cover the newest advancements and emerging trends within this field. Some reviews may cover only some domains or algorithm families, potentially missing out in the broader application and comparison of different bio-inspired techniques since there is little attention given to practical elements such as computational efficiency, robustness and issues related to actual implementation in the real world. These items are essential for people who work in this field. Comparative depth is about how far the comparison goes. Comparative analysis exists, but it might need to be more detailed and systematic across many new and old algorithms using the same standards. This review addresses gaps and goes deeper into analysis with recent updates and practical thinking. The goal is to offer a valuable resource for researchers and those in the bio-inspired optimization area. This part of the literature review is essential because it combines previous systematic reviews and emphasizes new things that this current study brings. By covering missing parts and analyzing recent advancements and practical aspects, this review wants to be a valuable resource for people doing research and work in the bio-inspired optimization field.

The most commonly used bio-inspired optimization techniques are Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Grey Wolf Optimizer (GWO) and Artificial Bee Colony (ABC). Each method has its benefits as well as some limitations. The Genetic Algorithm, for instance, is known for being strong and able to locate global optima in complicated, multivariable exploration spaces. It doesn't need gradient information, so it's good for non-differentiable or discontinuous functions. GAs ensure their solution population remains varied using crossover and mutation operators. This helps them explore more possibilities and decreases the chance of getting stuck in the local best solutions. However, GAs can take up a lot of computer resources because they require assessing many possible answers throughout multiple generations, where their effectiveness strongly depends on set parameters such as the size of the population, the rate for crossover and rate for mutation. PSO is preferred for its simplicity and efficiency in computations. The idea behind PSO is similar to how birds and fish act socially. It uses a group of particles that travel in the search space based on their own best-known position and the best-known positions of other particles nearby. This algorithm is simple to apply and has fast convergence towards top-notch solutions, which makes it suitable for instant-use situations. However, PSO might encounter the problem of early convergence and stagnation, particularly in intricate multimodal landscapes. The Grey Wolf Optimizer (GWO) imitates grey wolves' leadership structure and hunting actions. Wolves are divided into distinct positions (alpha, beta, delta and omega), controlling the search process.

GWO is good at managing a trade-off between exploration and exploitation. Because it is simple and has a solid local search ability, it has also been used successfully for different optimization problems. But, just like other meta-heuristic algorithms, GWO can also experience performance issues in spaces with high dimensions and needs fine adjustments of parameters to reach the best results. The Artificial Bee Colony (ABC) algorithm is inspired by how honey bees search for food through a division into employed bees, onlookers and scout bees. They work together to explore and exploit the search space. ABC is especially praised for its capability to tackle intricate optimization problems with fewer control parameters, giving more flexibility and robustness. It successfully handles the issue of getting stuck in local optima using its exploration–exploitation balance. However, ABC might converge slower than other algorithms, such as PSO and GWO. Also, its effectiveness can change significantly based on the optimization problem it is used for. Even with these limits, the ongoing development and mixing of these algorithms have created better versions that work to reduce their natural flaws. This makes them essential tools in optimization methods used across many different science and engineering areas.

3 Meta-heuristic models

Condensing all relevant data and meta-heuristic algorithms into one article becomes difficult. The survey compiles the majority of the current meta-heuristic models in engineering applications. The list of bio-inspired optimization strategies used to solve challenging real-time engineering challenges is fully explored in this section. It also covers the advantages and disadvantages of the most modern bio-inspired optimization techniques and the operational flow, assessment models, parameter descriptions, and advantages and disadvantages. The taxonomy of bio-inspired algorithms utilized in various application system types is shown in Table 1. The algorithms chosen for review in this study are a well-thought-out assortment that tries to cover the span and intricacy

Table 1 List of Bio-inspired algorithms

Year	References	Techniques
1989	Sivaram et al. (2019); Lin et al. (2019)	Tabu Search
1992	Deng et al. (2019); Dorigo and Stützle (2019); Uthayakumar et al. (2020)	Ant Colony Optimization (ACO)
1992	Kalita et al. (2020); Mahmoodpour et al. 2021)	Genetic Programming (GP)
1995	Sengupta et al. (2018); Jain et al. (2018); Chopard and Tomassini (2018)	Particle Swarm Optimization (PSO)
2001	Abualigah et al. (2020); Mousavi et al. (2021); Dubey et al. (2021)	Harmony Search Algorithm (HSA)
2003	Zanbouri and Jafari Navimipour (2020)	Queen Bee Evolution (QBE)
2004	Dokeroglu et al. (2019)	BeeHive Optimization (BHO)
2006	Massoudi et al. (2020)	Bees Algorithm (BA)
2008	Boughaci et al. (2020)	Roach Infestation Optimization (RIO)
2007	Kaveh et al. (2021a)	Imperialistic Competitive Algorithm (ICA)
2007	Rabanal et al. (2019)	River Formation Dynamics (RFD)
2008	Yan and Lu (2018)	Fast Bacterial Swarming Algorithm (FBSA)
2009	Rashedi et al. (2018)	Gravitational Search Algorithm (GSA)
2009	Nikolić et al. (2020)	Bee Colony Optimization (BCO)
2009	Xiuwu et al. (2019)	Glowworm Swarm Optimization (GSO)
2009	Shishavan and Gharehchopogh (2022)	Cuckoo Search Optimization (CSO)
2010	Sasikala (2019)	Firefly Optimization (FO)
2010	Ammal et al. (2020)	Termite Colony Optimization (TCO)
2010	Tzanetos and Dounias (2020)	Japanese Tree Frogs (JTF)
2010	Jayabarathi et al. (2018)	Bat Algorithm (BA)
2011	Rachappanavar et al. (2022)	Galaxy Based Search (GBS)
2011	Cruz-Duarte et al. (2021)	Spiral Optimization Algorithm (SOA)
2012	Kaveh et al. (2021b)	Big Bang-Big Church
2012	Abdel-Basset and Shawky (2019)	Flower Pollination Algorithm (FPA)
2012	Almufti (2019)	Great Salmon Run (GSR)
2013	Kaur and Sharma (2022)	Egyptian Vulture Optimization Algorithm (EVOA)
2014	Abdullahi et al. (2020)	Symbiotic Organisms Search (SOS)

Table 1 (continued)

Year	References	Techniques
2015	Assiri et al. (2020)	Ant Lion Optimization (ALO)
2015	Zhou et al. (2019)	Water Wave Optimization (WWO)
2016	Pitchipoo et al. (2021)	Dragon Fly (DF) Optimization
2017	Moazzeni and Khamehchi (2020)	Rainfall Optimization Algorithm (ROA)
2017	Chen et al. (2020)	Bacterial Foraging Inspired Algorithm (BFIA)
2017	Talatahari and Azizi (2021)	Fractal Based Algorithm (FBA)
2017	Meraihi et al. (2021)	Grasshopper Optimization Algorithm (GOA)
2019	Sharma et al. (2019)	Spider Monkey Optimization (SMO)
2019	Shadravan et al. (2019)	Sailfish Optimization (SO)
2019	Heidari et al. (2019)	Harris Hawks Optimization (HHO)
2020	Fu et al. (2022)	Black Window Optimization (BWO)
2020	Kaveh et al. (2021c)	Water Strider (WS) Algorithm
2020	Nikpour and Mohebbi (2022)	Newton Meta-Heuristic Algorithm (MHA)
2020	Martínez-Álvarez et al. (2020)	Coronavirus Optimization Algorithm (COA)
2020	Khishe and Mosavi (2020)	Chimp Optimization Algorithm
2020	Kadry et al. (2021)	Mayfly Optimization Algorithm (MOA)
2021	Basu et al. (2022)	Horse Herd Optimization (HHO) Algorithm

of advancements in bio-inspired optimization methods. Each algorithm's selection was influenced by several important factors, which are listed below:

- Relevance and Impact: Algorithms like Tabu Search, Ant Colony Optimization (ACO), Genetic Programming (GP), etc., were chosen because they are widely used in academic research and have a significant influence on optimization. They are also successfully applied across many fields.
- Inspirations from Different Areas: Bio-inspired phenomena from various fields inspire the algorithms. The inspiration from social insects such as bees (like Bee-Hive Optimization or Bees Algorithm) and ants (such as ACO or Ant Lion Optimization) helped to articulate this survey. Further, the natural dynamics like gravitational forces (for example, Gravitational Search Algorithm) plus river formations are shown in specific algorithms such as River Formation Dynamics, among others. Every algorithm gives its particular viewpoint on how to solve optimization problems.
- Novelty and Innovation: This survey selected algorithms like the Coronavirus Optimization Algorithm (COA), the Horse Herd Optimization (HHO) Algorithm, and the Spider Monkey Optimization (SMO) algorithm because they present a new way of thinking in the field, which has recently been shown in the literature. These modern methods signify the forefront of bio-inspired optimization research, introducing fresh ideas and techniques.
- Performance and Efficacy: The selection criteria also considered algorithms' performance and efficacy in solving difficult optimization problems. Particle Swarm Optimization (PSO), Firefly Optimization (FO), and Water Wave Optimization (WWO), for example, have proven robustness and scalability over a broad variety of applications, which makes them notable choices to be reviewed.
- Algorithm Diversity: The survey included most algorithms that imitate different natural events and living systems. Every algorithm, from animal actions like Harris Hawks Optimization or Horse Herd Optimization to ecological changes such as the Great Salmon Run and Rainfall Optimization Algorithm, offers a unique viewpoint on optimization inspired by nature.

This study includes a wide variety of algorithms, which aims to give an overall understanding of the latest methods in bio-inspired optimization techniques. It also intends to highlight their features, ways of measuring performance and real-life impact on optimization problems within various fields.

Table 2 presents the comparative analysis of various bio-inspired optimization techniques based on robustness, fast convergence speed, solution capability, stopping criteria, success rate, and computational efficiency.

- Robustness: An algorithm's stability refers to how steady it is in producing the same correct and dependable result, particularly when handling various input data sets. The survey used a range of test problems designed to reflect different complexities and traits to test each algorithm's robustness. Robustness was measured by checking whether the algorithm could consistently find high-quality solutions on different problem sets.

Table 2 Comparative analysis between various bio-inspired optimization techniques

Algorithms	Robustness	Fast convergence speed	Solution capability	Stopping criteria	Success rate	Computational efficiency
HHO	✓	✓	×	✓	✓	✓
MOA	✓	✓	✓	×	✓	✓
COA	✓	✓	✓	✓	×	×
MHA	×	×	✓	✓	×	×
WS	✓	×	✓	×	×	×
BWO	×	×	✓	✓	×	×
SO	✓	×	✓	✓	✓	×
SMO	×	✓	×	✓	✓	✓
GOA	×	✓	✓	✓	✓	✓
FBA	✓	✓	✓	×	×	×
BCO	×	✓	×	✓	✓	✓
ROA	✓	×	×	✓	×	×
DF	×	✓	✓	✓	✓	×
WWO	✓	✓	×	×	×	✓
ALO	×	×	✓	✓	×	✓
SOS	×	×	✓	✓	×	×
EVOA	✓	×	×	✓	×	×
GSR	✓	✓	×	✓	×	×
FPA	✓	✓	×	×	✓	✓
BA	×	✓	×	✓	✓	✓
FFO	×	×	✓	✓	✓	×

- Fast Convergence Speed: The term "quickly converging speed" refers to a situation where the algorithm can obtain solutions very close to optimal within a few iterations or computational time. Convergence speed was evaluated by measuring the algorithm's convergence rate on standard benchmark problems. Algorithms that quickly reached the best or nearly the best solutions were seen as those with fast convergence speeds.
- Solution Capability: Solution capability reflects an algorithm's ability to find high-quality solutions within a reasonable timeframe. Each algorithm was assigned a task to solve benchmark optimization problems. The prime aim for all of them is to discover solutions with superior fitness values.
- Capability of Solution: The quality of solutions obtained and their nearness to the global optimum were assessed.
- Stopping Criteria: Stopping criteria dictate when an optimization algorithm should terminate its search process. The survey evaluates the efficiency of stopping rules based on their ability to control computational resources and maintain solution quality. Algorithms with well-defined and effective stopping criteria were rated favourably in this category.
- Success Rate: The success rate measures how well the algorithm can continue to find better or equal solutions to different problem instances.

- Success Percentage: The success rate results from applying the algorithm to a typical set of problem examples. It shows how often this algorithm could find the best or close-to-best solutions, which is recorded as a percentage.
- Computational Efficiency: Efficiency in computation is about how well an algorithm utilizes computational resources to produce good solutions. Efficiency was evaluated by measuring the algorithm's runtime and memory consumption on benchmark problems. Algorithms capable of achieving similar solution quality but requiring less computational power were referred to as computationally efficient.

By employing standardized methods to assess each algorithm against these criteria thoroughly, the comparison shown in Table 2 provides a comprehension of their relative strengths and drawbacks. This assists researchers and practitioners in selecting an optimization technique that best meets their specific needs.

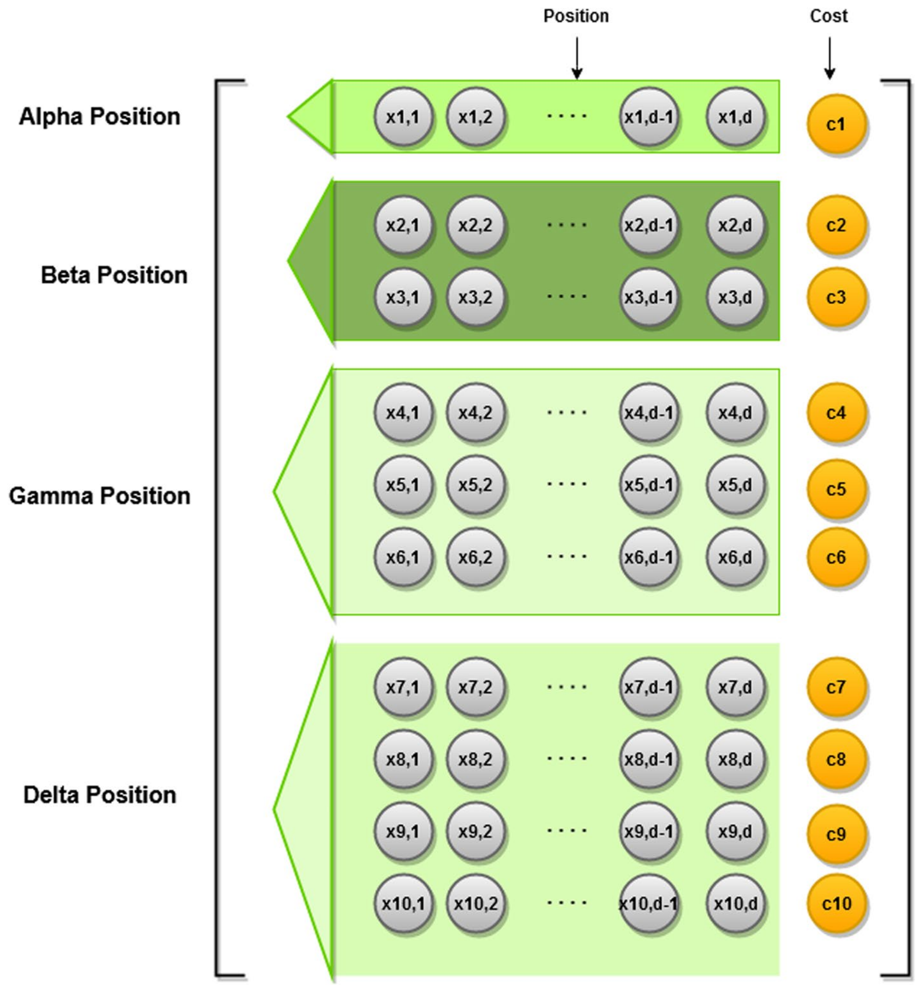

Position
Cost
Alpha Position
Beta Position
Gamma Position
Delta Position
x1,1 x1,2 .... x1,d-1 x1,d c1
x2,1 x2,2 .... x2,d-1 x2,d c2
x3,1 x3,2 .... x3,d-1 x3,d c3
x4,1 x4,2 .... x4,d-1 x4,d c4
x5,1 x5,2 .... x5,d-1 x5,d c5
x6,1 x6,2 .... x6,d-1 x6,d c6
x7,1 x7,2 .... x7,d-1 x7,d c7
x8,1 x8,2 .... x8,d-1 x8,d c8
x9,1 x9,2 .... x9,d-1 x9,d c9
x10,1 x10,2 .... x10,d-1 x10,d c10

Fig. 2 The sorting process of the HHO technique

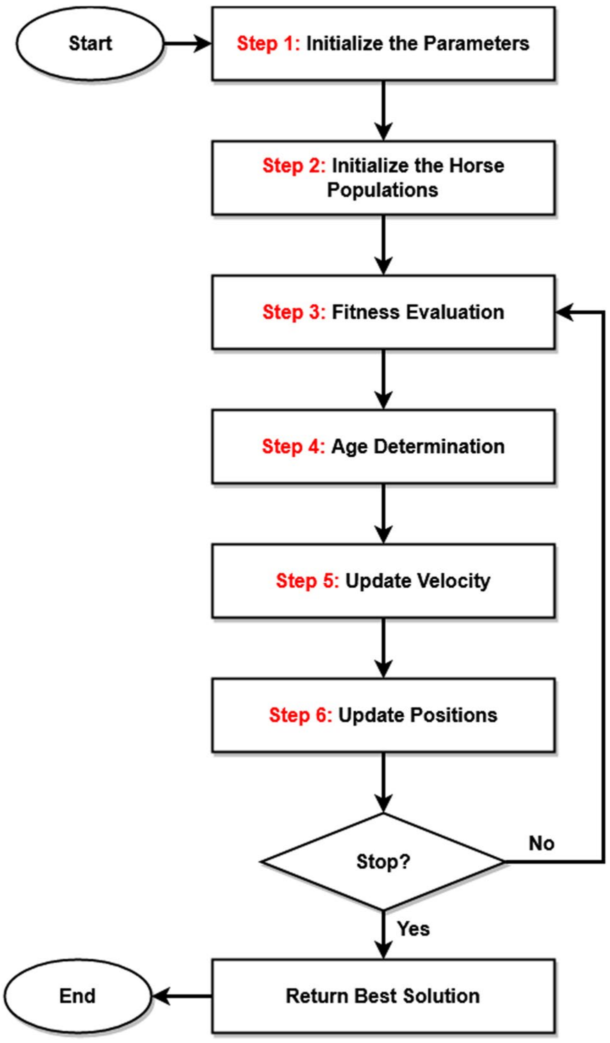

Start
Step 1: Initialize the Parameters
Step 2: Initialize the Horse Populations
Step 3: Fitness Evaluation
Step 4: Age Determination
Step 5: Update Velocity
Step 6: Update Positions
Stop?
No
Yes
Return Best Solution
End

Fig. 3 Flow of HHO technique

3.1 Horse Herd Optimization (HHO)

The Horse Herd Optimization (HHO) (Awadallah et al. 2022) is a modern optimisation algorithm developed recently for solving complex and multi-objective problems. It works based on horse behaviour, like hierarchy, sociability, grazing, imitation, roaming, and defence. MiarNaeimi et al. (2021) *suggested an HHO technique to provide suitable solutions to* big dimensional optimization problems. Here, the computational complexity of this optimization is validated and examined by computing the cost of operations. The position update is estimated based on sorting the global matrix, as shown in Fig. 2, where the sorting of computational cost determines the best and worst states. The key benefits of using this technique are increased processing speed, reduced computational cost, and better recognition capability. In addition to that, the Uni-modal benchmark functions are utilized in this work for parameter sensitivity analysis. Figure 2 shows the sorting process of the HHO technique, and its working model is depicted in Fig. 3.

Algorithm 1 Horse Herd Optimization (HHO)

```
Initialize population of horses $P$ with random positions $X_i$
Set number of horses $N, maxIter$, and other parameters
Evaluate fitness of each horse and determine $X_{best}$
For $t = 1$ to $maxIter$ do
  For each horse $i\ in\ P$ do
    If the exploration phase then
      $r1 = random(0,1)$
       $X_i(t+1) = X_i(t) + r1 * |X_{best}(t) - X_i(t)|$
    Else if the exploitation phase, then
      $r2 = random(0,1)$
      $X_i(t+1) = X_{best}(t) + r2 * (X_{best}(t) - X_i(t))$
    End if
    Optionally: perform random walk
    $r3 = random(0,1)$
   $X_i(t+1) = X_i(t) + r3 * (X_{rand}(t) - X_i(t))$
   Ensure $X_i(t+1)$ is within search space boundaries
   Evaluate fitness of $X_i(t+1)$
   If fitness($X_i(t+1)$) > $fitness(X_{best}(t))$ then
      $X\_best(t+1) = X\_i(t+1)$
    End if;
  End for;
End for;
Output the best solution $X_{best}$ and its fitness;
```

Elmanakhly et al. (2022) introduced a Binary Horse Herd Optimization (BinHOA) technique to perform efficient feature learning in large, medium, and small-scale datasets. Here, the Levy flight operator has been utilized to enhance the exploring behaviour and strengthen optimization performance. The local search algorithm was incorporated with this model to obtain the best optimal solution after each iteration. The Roulette Wheel Selection (RWS) algorithm combined with Levy flight improves the overall optimization performance. In this study, some other recent optimization techniques are compared with the BinHOA technique to evaluate its performance and effectiveness, which includes the following existing models:

- Flow Directional Algorithm (FDA)
- Whale Optimization Algorithm (WOA)
- Equilibrium Optimization (EO)
- Dragonfly Optimization (DO)

Figure 2 shows the working model of the HHO technique, which holds the operations of parameter initialization, fitness evaluation, age determination, velocity, and position updation.

The HHO technique also starts with the other techniques with the parameter setup (i.e. initialization of controlling parameters, maximum number of iterations, and population size). Let's consider the horses can move at each iteration as represented below:

$$H_k^{t,A} = \overrightarrow{Ve}_k^{t,A} + H_k^{(t-1),A} \tag{1}$$

where, A indicates the age of horse that is $A = \alpha, \beta, \gamma, \delta$, k is the position of horse, t indicates the iterations, and $\overrightarrow{Ve}_k^{t,A}$ represents the velocity of kth horse. The age of horse is determined as follows:

- $\alpha =$ More than 15 years.
- $\beta =$ Age between 10 and 15 years.
- $\gamma =$ Age between 5 and 10 years.
- $\delta =$ Age between 0 and 5 years.

Based on this factor, the six horse social skills—grazing, hierarchy, sociability, imitation, defense, and roam—can be estimated.

3.2 Black Widow Optimization (BWO)

The Black Widow Optimization (BWO) technique is increasingly utilized in many application systems to obtain the best optimal solution to solve the given problem with increased convergence speed. This technique could eliminate the inappropriate fitness of the initial populations and avoid the local minima. It begins with the process of population initialization, where each spider indicates some possible solutions. When breeding, the female BW can eat the male BW. The flow of the BWO technique is shown in Fig. 4, which includes the operations of population initialization, fitness computation, procreate (i.e. new generation production), cannibalism, and mutation. Kumar et al. (2021) *suggested the BWO technique for* energy-efficient scheduling in cloud systems. This technique primarily aimed to obtain an improved QoS by allocating the resources based on the optimal solution.

3.3 Mayfly optimization algorithm (MOA)

The mayfly optimization (Zhao and Gao 2020) techniques are mainly used to solve feature selection problems, and this method is developed based on the mating process of mayflies (i.e. insects). The essential characteristic of this technique is that it incorporates the benefits of conventional GA, PSO, and FA. Hence, it is more suitable for handling large dimensional datasets. Furthermore, it includes the following processes:

- Movement of male mayflies
- Movement of female mayflies
- Crossover between mayflies
- Mutation of mayflies

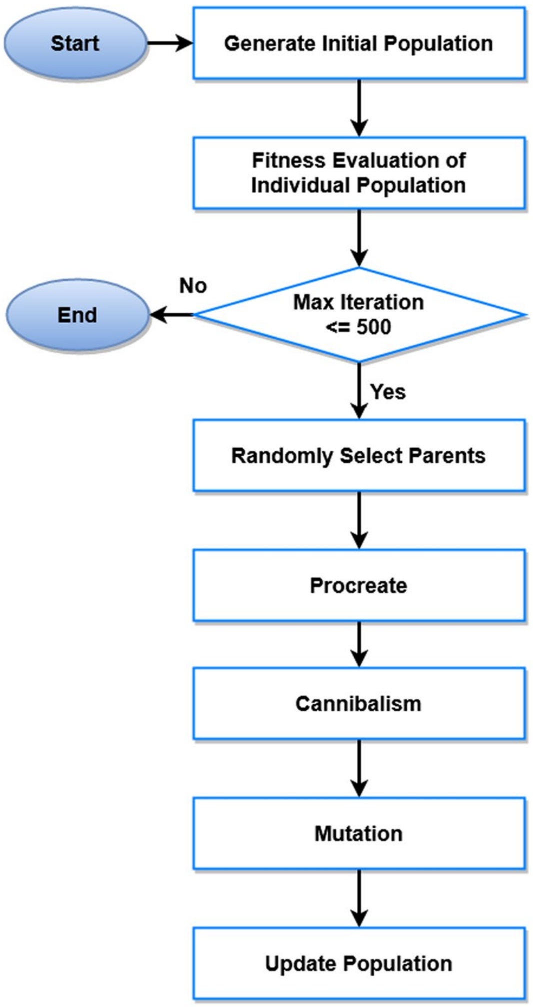


Fig. 4 Working flow of BWO technique

Initially, the position of male mayflies is initialized as follows:

$$p_i^{k+1} = p_i^k + e_i^{k+1} \tag{2}$$

where, p_i^{k+1} and e_i^{k+1} indicates the present position and velocity of male mayflies, respectively. Then, its vector is updated as follows:

$$e_{tj}^{k+1} = q_c \times e_{tj}^k + s_1 \times e^{-\delta r_p^2} \times \left(pb_{tj} - p_{tj}^k\right) + s_2 \times e^{-\delta r_p^2} \times \left(gb_{tj} - p_{tj}^k\right) \tag{3}$$

where, e_{tj}^k indicates the velocity of mayfly t with dimension j and time k, p_{tj}^k is the position of mayfly, s_1 and s_2 are the positive attraction constants, q_c indicates the gravitational coefficient, δ denotes the visibility coefficient, pb_{tj} indicates the optimal position mayfly t, and gb_j is the position of best mayfly. Consequently, the fitness value of mayfly with its optimal position is computed as follows:

$$pb_t = \begin{cases} p_i^{k+1} \\ if\ fitness\left(p_i^{k+1}\right) < fitness(pb_t) \end{cases} \tag{4}$$

Similarly, the movement of female mayflies is computed as follows:

$$r_i^{k+1} = r_i^k + e_i^{k+1} \tag{5}$$

In this model, the quality of the current solution highly depends on the attraction between the male and female mayflies, as represented below:

$$e_{tj}^{k+1} = \begin{cases} if fitness\left(r_t\right) > fitness(p_t) \\ q_c \times e_{tj}^{k+1} + s_2 \times e^{-\delta d_{mf}^2} \tau \times \left(p_{tj}^k - r_{tj}^k\right) \\ else if fitness\left(r_t\right) \le fitness(p_t) \\ q_c \times e_{tj}^k + wc \times d \end{cases} \tag{6}$$

where, e_{tj}^k and r_{tj}^k indicates the velocity and position of female mayfly respectively, d_{mf} represents the Cartesian distance of male and female mayflies, wc is the random walk coefficient, and d indicates the random value. After that, the crossover between the mayflies is estimated as shown below:

$$O_s 1 = a_{os} \times male + (1 - a_{os}) \times female \tag{7}$$

$$O_s 2 = a_{os} \times male + (1 - a_{os}) \times male \tag{8}$$

where, $O_s 1$ and $O_s 2$ are the offspring 1 and 2, and a_{os} indicates the stipulated value lies between the range of 0 to 1. Then, the mutation is computed by using the following model:

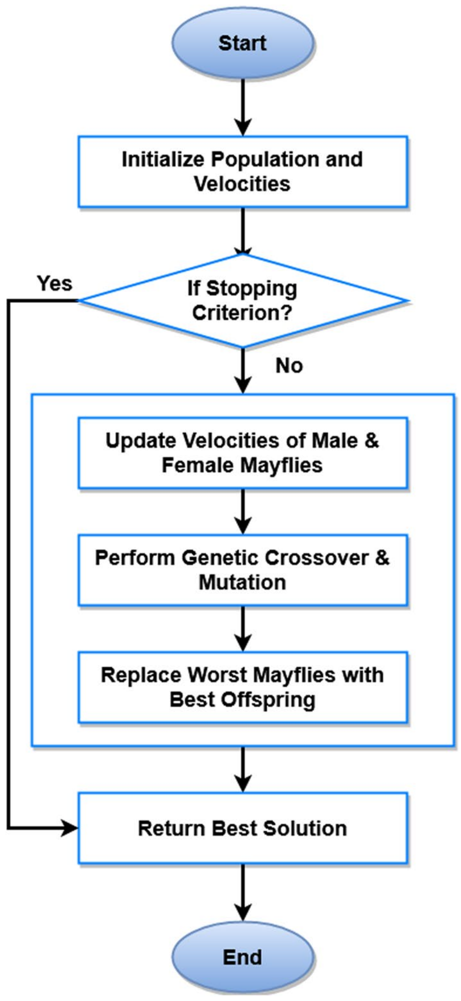
$$O\prime_s = O_s + u \tag{9}$$

where, u indicates the distributed random value. The working flow of the MOA is shown in Fig. 5.

Algorithm 2 Mayfly Optimization

```
Inputs:  Population size, maximum iterations;
Output:  Best agent $P = (p_1, p_2 \dots p_d)$;
Step 1:  Initialize the set of population and velocity of both male and female mayflies;
Step 2:  Estimate the gbest value for the given population;
         For $itration \leftarrow 1$ to maximum iterations do
             For $i \leftarrow 1$ to population size do
                 Update the value of pbest;
                 Estimate and update the velocities of both male and female mayflies;
             End for;
Step 3:  Sort and rank the mayflies according to the velocity and position;
Step 4:  Perform crossover and mutation operations for both male and female mayflies;
Step 5:  Replace the worst mayflies with the best offspring value;
Step 6:  Update gbest
Step 7:  End for;
```

Fig. 5 Working flow of MOA


Start
Initialize Population and Velocities
If Stopping Criterion?
Yes
No
Update Velocities of Male & Female Mayflies
Perform Genetic Crossover & Mutation
Replace Worst Mayflies with Best Offspring
Return Best Solution
End

Bhattacharya et al. (2020) suggested a hybrid Mayfly-Harmony Search (MA-HS) optimization methodology for solving the combinatorial optimization problem. Here, the primary purpose of developing an integrated method is to improve the exploration ability of optimization by tuning the parameters. Zheng et al. (2020) *investigated the performance and effectiveness of using a Heterogeneous Mayfly Optimization (HMO) technique for* multi-objective problems. Due to its multiple-choice updating behaviour, it provides the optimal global solution with increased convergence speed. Hence, it is suitable for handling different optimization problems with better performance outcomes.

3.4 Water Strider Algorithm (WSA)

The Water Strider Algorithm (WSA) (Kaveh et al. 2021c) is a population-based optimization technique that mimics the behaviour of water striders. In this model, the social behaviour patterns of WS (Syah et al. 2021) are considered for estimating the fitness function, which includes the processes of birth, territory establishment, mating process, feeding, death & succession, and termination criteria. At first, all striders are randomly initialized in the searching space as illustrated as follows:

$$X_i^0 = L_b + r_i.\left(U_b - L_b\right), i = 1,2 \ldots N \tag{10}$$

where, X_i^0 indicates the initial position of *i*th WS, U_b and L_b are the upper and lower bounds respectively, N is the total number of WS, and r_i is the random value with the uniform distribution of 0 to 1. Then, the objective function is computed for the given optimization problem based on the position of WS. Based on the estimated fitness function, the total number of WS is sorted and placed in the groups; then, the WS is split into territories based on their ranks. During the mating process, the keystone position has been updated in each case, as illustrated below:

$$\begin{cases} X_i^{k+1} = X_i^k + V \cdot r_i ifmatinghappens \\ S_i^{k+1} = S_i^k + V \cdot \left(A_v + r_i\right) Otherwise \end{cases} \tag{11}$$

where, X_i^{k+1} indicates the position of *i*th WS in *k*th cycle, r_i is the random number lies in the range of 0 to 1, V is the vector, and A_v denotes the array vector. The target female is selected for mating based on the highest and lowest probability values. The keystone could evoke the consumed energy during the feeding stage, and its new position is updated during the mating process. It is illustrated as follows:

$$X_i^{k+1} = X_i^k + 2 \times r_i(X_{Bc}^k - X_i^k) \tag{12}$$

here, the best cost value X_{Bc}^k of WS is computed according to its current position. If it does not identify the food source, the keystone is killed to increase its energy level, and a new WS can be replaced by using the following model:

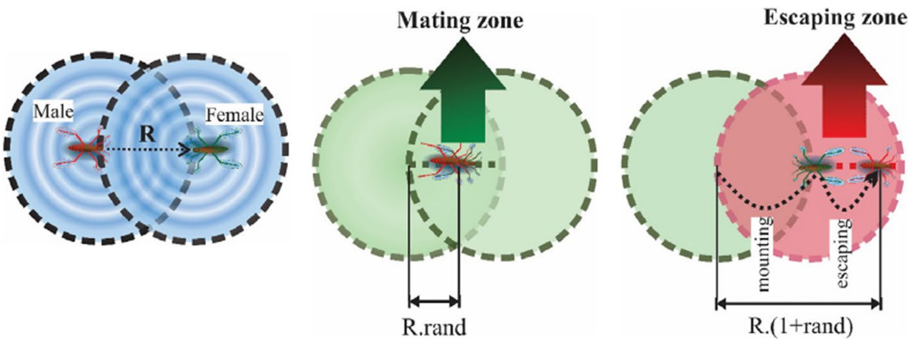


Fig. 6 WS mating behavior and position updation

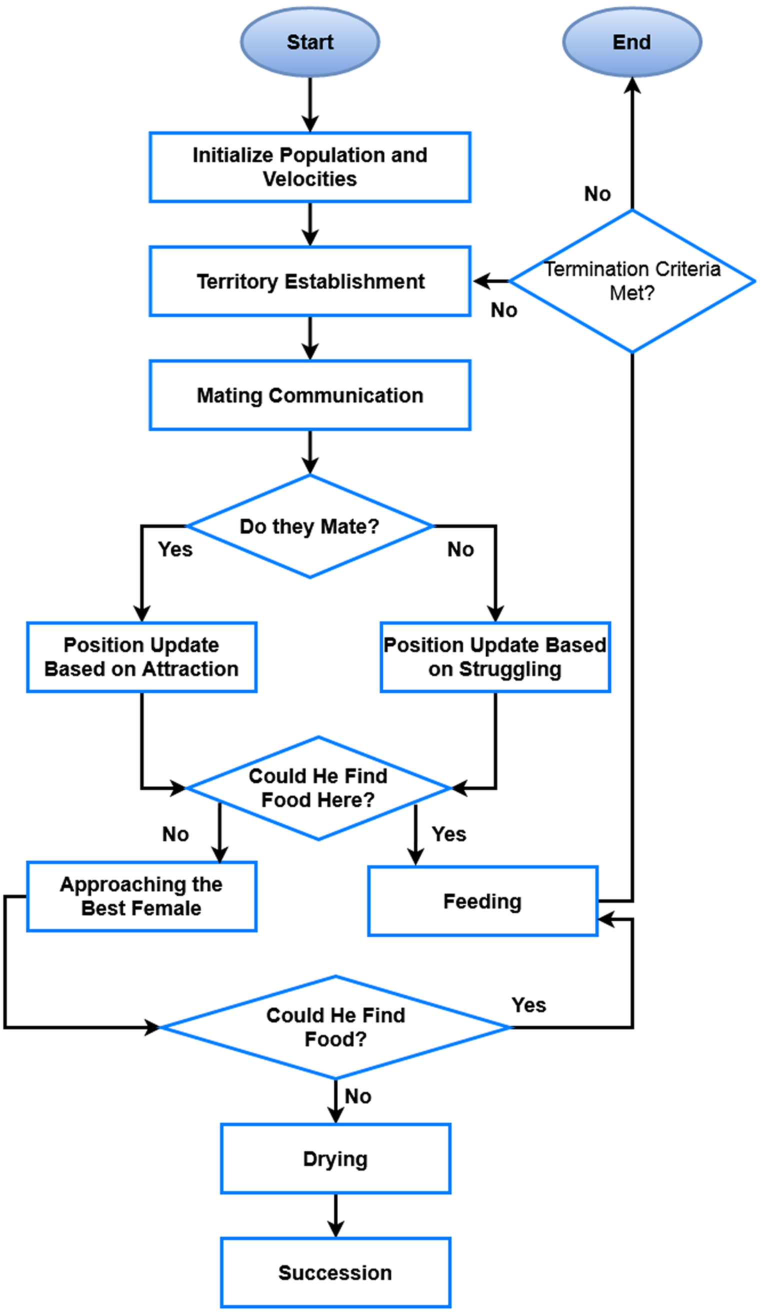

Start
End
Initialize Population and Velocities
Territory Establishment
Termination Criteria Met?
No
No
Mating Communication
Do they Mate?
Yes
No
Position Update Based on Attraction
Position Update Based on Struggling
Could He Find Food Here?
No
Yes
Approaching the Best Female
Feeding
Could He Find Food?
Yes
No
Drying
Succession

Fig. 7 Flow of WSA

$$X_i^{k+1} = LB_j^k + r \times (UB_j^k - LB_j^k) \tag{13}$$

where, X_i^{k+1} denotes the new position of WS, r is the random value, UB_j^k and LB_j^k are the upper and lower bounds of WS, respectively. The mating process of WS is illustrated with its position update in Fig. 6, and the overall flow of the WSA is represented in Fig. 7.

Algorithm 3 WSA

```
Inputs:  Population size, maximum number of iterations, and number of territories;
Outputs:        WS position and objective function;
Step 1:  Randomly initialize the set of populations and, compute the fitness value of all WS;
Step 2:  while (if the termination is not met) do
           The $N$ number of territories are established and allocated to the WSs;
Step 3:    for (each territory)
                The keystone sends the mating ripples for selecting the female;
                Then, the position of the keystone is updated according to the response provided by the female;
                The keystone finds the food source for managing the consumed energy at the time of mating;
Step 4:         If (Keystone does not have a food source) then
                  The hungry keystone can be killed, and the new territory can be replaced as the successor;
                End if;
         End for;
Step 5:  Return the best position;
```

Kaveh et al. (2020) *utilized a WSA to solve complex optimization problems, discussing the significant purposes of using this optimization technique* with its original characteristics. Also, some other benchmark functions have been utilised to validate the

Table 3 Comparative analysis based on cost

Techniques	Optimal variables				Optimal cost
	×*1*	×*2*	×*3*	×*4*	
WSA	0.205	3.470	9.036	0.205	1.724
GA	0.220	3.282	8.470	0.219	1.77
PSO	0.219	3.434	8.433	0.236	1.852
ICA	0.205	3.466	9.034	0.205	1.725
MFO	0.205	3.470	9.036	0.205	1.724
SCA	0.205	3.467	9.048	0.205	1.725
BBO	0.185	4.312	8.439	0.235	1.91
CBO	0.205	3.470	9.036	0.205	1.724
NNA	0.205	3.470	9.036	0.205	1.724
GWO	0.205	3.470	9.038	0.205	1.725
TEO	0.205	3.472	9.035	0.205	1.725
WOA	0.204	3.519	8.959	0.213	1.777
GSA	1.182	3.856	10	0.202	7.879
HHO	0.204	3.531	9.024	0.206	1.731
HS	0.244	6.223	8.291	0.244	0.238
ES	0.199	3.612	9.0375	0.206	1.737
CDE	0.203	3.542	9.033	0.206	1.733
CPSO	0.202	3.544	9.048	0.205	1.728
FGA	0.205	3.471	9.020	0.206	1.72

objective functions of this technique. The key benefit of using this technique is that it efficiently reduces the computational complexity of processing with increased convergence speed. Moreover, the optimal cost of various optimization techniques is validated and compared based on the variables' optimal values, as shown in Table 3.

3.5 Coronavirus Optimization Algorithm (COA)

The Coronavirus Optimization Algorithm (COA) is a recent bio-inspired mechanism developed based on coronavirus spreading (Shadkam 2021), where factors such as reinfection probability, social distancing measure, super spreading rate, and travelling rate have been considered. In this technique, the input parameters are already defined according to the disease statistics, so initialising the parameters with arbitrary values is unnecessary. Also, it can reach up to the end after executing various iterations without fixing the value. It also includes the following processes: generation of initial populations, disease propagation, population updation, and stopping criterion. Here, the initial population is generated based on the random solution, and the number of iterations is executed according to the duration of the preset value.

Algorithm 4 Coronavirus Optimization

```
Inputs: Infected population, Individuals (newly infected persons), dead $D$, recovered list $Re$, patient zero $Pa_Z$, best
        individual $B_I$, current best individual $CB_I$, and time (T);
Output: Best fitness solution, and current best fitness value;
        Set the parameters as follows:
        $T \leftarrow 0$;
        $Pa_Z \leftarrow Infect\ Patient\ Zero$ ();
        Infected population $\leftarrow Pa_Z$;
        Best Individual $\leftarrow Pa_Z$;
        While ($T < P_D$ && $size\ of\ (I_P) > 0$) do
        //$P_D$ - Pandemic duration, and $I_P$ – Infected population;
            $D \leftarrow die\ (I_p)$;
            For all $i \in I_P$ do
                $aux \leftarrow infect\ (i, Re, D)$;
                If notnull (aux) then
                  $NI_P \leftarrow aux$;
                End if;
            End for;
            $CB_I \leftarrow Select\ B_I(NI_P)$;
            If $fitness\ (CB_I) > B_I$ then
                $B_I \leftarrow CB_I$;
            End if;
                $R_e \leftarrow I_P$;
                $T \leftarrow T + 1$;
        End while;
        Return $B_I$;
```

Majdoubi et al. (2021) utilized a COA to solve the general combinatorial problem with the suitable solutions. The primary purpose of using this technique is to provide possible explanations for handling large datasets with a reduced computational process. The exact optimum corresponding to the Travelling Salesman Problem (TSP) with suitable resolutions is obtained. AI-Betar et al. (2021) investigated the performance and

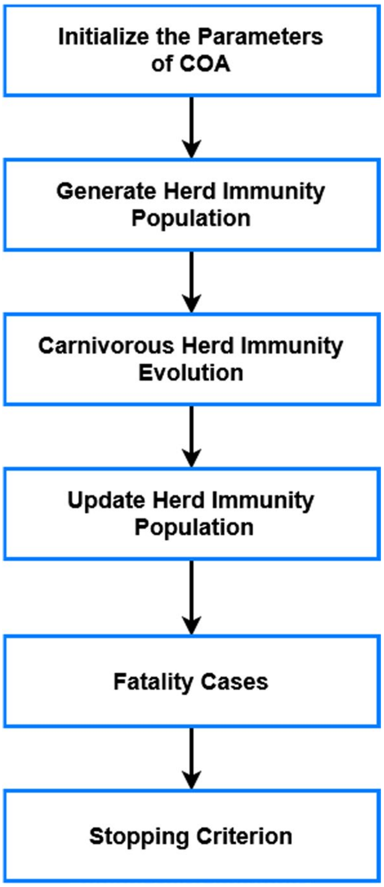

Initialize the Parameters of COA
Generate Herd Immunity Population
Carnivorous Herd Immunity Evolution
Update Herd Immunity Population
Fatality Cases
Stopping Criterion

Fig. 8 Working flow of COA technique

efficiency of using Coronavirus Herd Immunity Optimizer (CHIO) to solve different optimization problems. This work stated that the CHIO is one potent and efficient optimization technique suitable for various application systems. The operational flow of the CHIO technique is shown in Fig. 8, which comprises the operations of inspiration, herd immunity, population hierarchy, social distancing, population update, and optimal solution identification.

3.6 Fractal Based Algorithm (FBA)

The Fractal Based Algorithm (FBA) (Aromolaran et al. 2021) is a continuous optimization technique that is mainly used to solve complex optimization problems. In this technique, the state-space model is partitioned into the fractals, where the constant optimization problems are solved iteratively based on the self-similar and fractal shaped structure. It provides the best optimal solution by predicting the degree of promise in the state space.

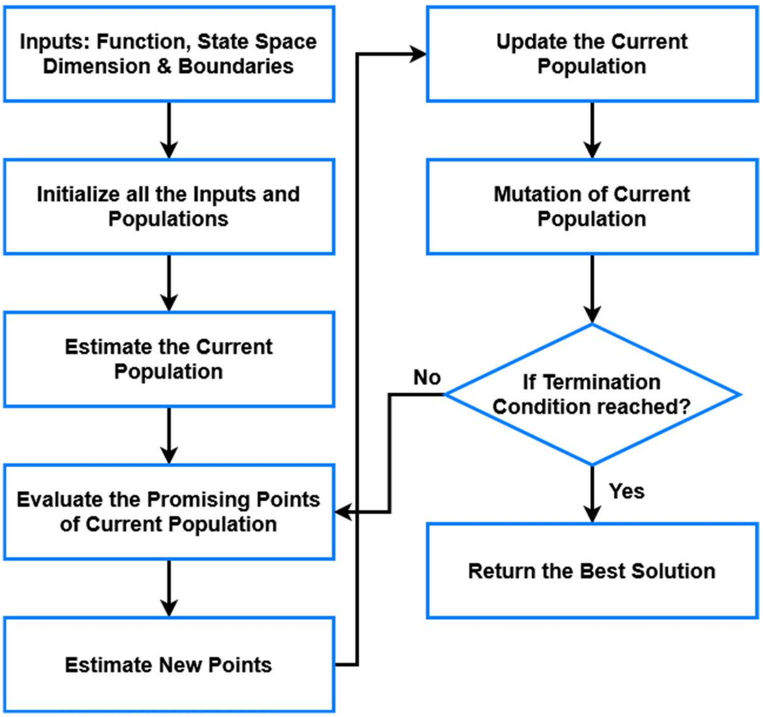

Inputs: Function, State Space Dimension & Boundaries
Initialize all the Inputs and Populations
Estimate the Current Population
Evaluate the Promising Points of Current Population
Estimate New Points
Update the Current Population
Mutation of Current Population
If Termination Condition reached?
No
Yes
Return the Best Solution

Fig. 9 Flow of FBA

Also, it identifies the optimal function without loss of generality, where the minor changes are applied to determine the maximum of continuous operation. Consequently, the optimal point is identified by generating the random points, where the minimum of a continuous function is obtained in the n-dimensional space. Then, the entire space is split into various sequences, and the initial population is randomly generated according to the uniform distribution function. Next, the current population is estimated with the lowest values, where the number of good points from the subspace are extracted. Similarly, the value function is also estimated according to the random points, and the fittest point of the new population and current population is integrated based on the truncation operation. Finally, the best solution is obtained and returned as the output, which is used to select the parameters optimally. Figure 9 shows the general working flow of the FBA based optimization technique.

3.7 Spider Monkey Optimization (SMO)

The Spider Monkey Optimization (SMO) (Sharma et al. 2019) is a recent swarm intelligence-based methodology mainly used to solve complex optimization problems. This algorithm is developed based on the organization and social behaviour of the spider monkeys, which works based on their intelligent foraging activities (i.e. fission–fusion) (Elmanakhly et al. 2022). According to the position and postures, the monkeys can share their information or observations with other monkeys. It includes the following modules (Singh et al. 2018). According to the position and postures, the monkeys can share their information or observations with other monkeys. It includes the following modules (Sharma et al. 2020): initialization, local leader phase, global leader phase, global leader learning, local leader learning, and decision phase. Based on these processes, the optimal best fitness function is generated that helps to solve the given optimization problems with balanced convergence speed. The SMO technique's main advantages are balanced exploration and exploitation, increased processing speed, and optimal performance rate with a reduced number of iterations.

During initialization, the N number of spider monkeys are initialized with the upper and lower bound values as shown below:

$$K_{ij} = K_{mnj} + U(0{,}1) \times (K_{mxj} - K_{mnj}) \tag{14}$$

where, K_i indicates the spider monkey, K_{mnj} and K_{mxj} are the lower and upper bounds of the searching space, and $U(0{,}1)$ indicates the uniformly distributed function ranging from 0 to 1. Then, the position of monkeys are updated based on the experiences of the local leader and group members as represented below;

$$K_{newij} = K_{ij} + U(0,1) \times \left(Loc_{tj} - K_{ij}\right) + U(-1{,}1) \times (K_{rj} - K_{ij}) \tag{15}$$

where, K_{ij} indicates the jth position of spider monkey i, Loc_{tj} is the local leader of tth group, and $U(-1{,}1)$ denotes the uniformly distributed random number. After this phase, the global leader phase is executed based on the selection probability, which helps to estimate the fitness function as shown below:

$$FF = f_i = \begin{cases} \frac{1}{1+o_j} & if\ o_i \geq 0 \\ 1 + abs(o_j) & if\ o_i < 0 \end{cases} \tag{16}$$

where, f_i is the fitness value and, o_i indicates the objective function. Then, the roulette wheel selection method is used to estimate the selection probability as shown in below:

$$Pb_i = \frac{f_i}{\sum_{i=1}^{N} f_i} \tag{17}$$

Consequently, the position of global leader is updated based on the experience and knowledge of group members as shown below:

$$K_{newij} = K_{ij} + U(0{,}1) \times \left(Glo_j - K_{ij}\right) + U(-1{,}1) \times (K_{rj} - K_{ij}) \tag{18}$$

During the decision phase, all the members in the group have updated their positions based on the experience of the global leader as represented below:

$$K_{newij} = K_{ij} + U(0{,}1) \times \left(Glo_{ij} - K_{ij}\right) + U(0{,}1) \times (K_{rj} - Loc_{tj}) \tag{19}$$

Finally, the best optimal solution is estimated according to the position of the global leader. The foraging behaviour of SMO technique is illustrated in Fig. 10.

Algorithm 5 Spider Monkey Optimization (SMO)

```
Step 1: Initialize the set of parameters as population, local leader limit Loc, global leader limit Glo, and perturbation rate;
Step 2: At first, the local and global leaders are identified;
Step 3: Update the position of local leader as shown in below:
        For each member $K_{ij} \in t^{th}$ group do
          For each $j \in \{1, 2, ... D) do$
              If $U(0,1) \geq per_r$ then
                $K_{newij} = K_{ij} + U(0,1) \times (Loc_{tj} - K_{ij}) + U(-1,1) \times (K_{kj} - K_{ij})$
              Else
                $K_{newij} = K_{ij}$
              End if;
          End for;
        End for;
Step 4: Update the position of global leader as shown below:
        Initialize count $Cnt = 0$;
        While $Cnt < size\ of\ group$ do
          For each $K_i \in$ group do
            If $U(0,1) > Pb_i$
              $Cnt = Cnt + 1$;
              Select the integer randomly from
              $j \in \{1,2, ... D\}$;
              Select the monkey $K_r \in$ group
              $K_{newij} = K_{ij} +$
                    $U(0,1) \times (Glo_j - K_{ij}) + U(-1,1) \times (K_{rj} - K_{ij})$
            End if;
          End for;
        End while;
Step 5: Perform learning through local and global leaders;
Step 6: The positions of local leader and global leader are updated in the decision phase;
        //Local leader
        If $LLCnt > LLL$ then
          Initialize local limiter count $LLCnt$ =0;
            For each $j \in \{1,2 ... D\}$ group do
            If $U(0,1) \geq Pb_i$ then
              $K_{newij} = K_{mnj} + U(0,1) \times (K_{mxj} - K_{mnj})$
            Else
              $K_{newij} = K_{mnj} + U(0,1) \times (Glo_j - K_{ij}) + U(0,1) \times (I_{rj} - Loc_{tj})$
            End if;
          End for;
        End if;
        //Global leader
        If $GLCnt > GLL$ then
          Initialize global limiter count $GLCnt$ =0;
            If $No\ of\ groups < \max\_group$ then
              The swarms are split into groups;
            Else
              Single group can be created by integrating all the groups;
            End if;
            Update the position of all local leaders;
          End if;
Step 7: Based on the decision of global leader, the decision of fusion-fission is obtained;
Step 8: If (termination is satisfied)
          Stop;
        Else
          The position of global leader is updated as the optimal solution;
```

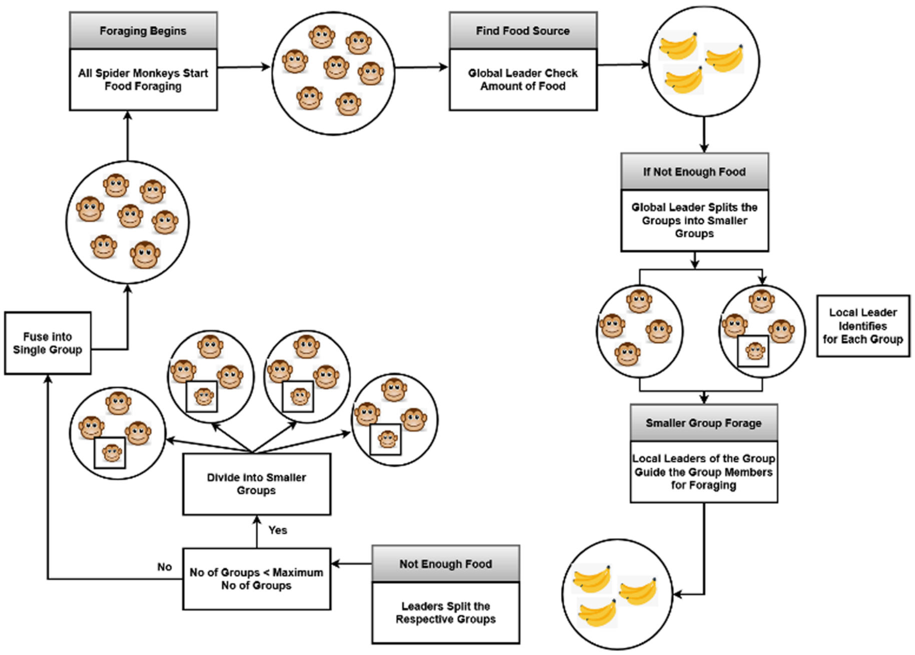

Foraging Begins
All Spider Monkeys Start Food Foraging
Find Food Source
Global Leader Check Amount of Food
If Not Enough Food
Global Leader Splits the Groups into Smaller Groups
Local Leader Identifies for Each Group
Smaller Group Forage
Local Leaders of the Group Guide the Group Members for Foraging
Not Enough Food
Leaders Split the Respective Groups
No of Groups < Maximum No of Groups
Yes
No
Divide Into Smaller Groups
Fuse into Single Group

Fig. 10 SM foraging behavior analysis

3.8 Newton Meta-Heuristic Algorithm (NMHA)

The Newton Meta-Heuristic Algorithm (NMHA) (Qin et al. 2018) is a familiar and population-based optimization technique mainly used to solve complex discrete optimization problems. Also, it provides an effective solution for handling the different types of issues based on the updating rule. For example, it finds the optimum value of the fitness function according to the first and second-order derivatives. Here, the iteration is constructed as follows:

$$p_i^{k+1} = p_i^k - \frac{f\prime(p_i^k)}{f\prime\prime(p_i^k)} \tag{20}$$

where, p_i^{k+1} and p_i^k indicates the iterations, $f\prime(p_i^k)$ and $f\prime\prime(p_i^k)$ are the first-order and second-order derivative functions of the point. Here, the following numerical assumptions are taken:

$$p_i^k - p_{i-1}^k = t(p_{i+1}^k - p_{i-1}^k) \tag{21}$$

$$p_{i+1}^k - p_i^k = (1-t)(p_{i+1}^k - p_{i-1}^k) \tag{22}$$

where t indicates the positive parameter. Similarly, the derivatives are also illustrated as follows:

$$f(x) = f\left(p_i^k\right) + \left(p - p_i^k\right)f\prime\left(p_i^k\right) + \frac{\left(p - p_i^k\right)^2}{2}f\prime\prime\left(p_i^k\right) \tag{23}$$

here, it is assumed as $\tau = p_{i+1}^k - p_i^k$, and its functions $f(p_{i-1}^k)$ and $f(p_{i+1}^k)$ are estimated as follows:

$$f(p_{i+1}^k) = f\left(p_i^k\right) + \tau(1-t)f\prime\left(p_i^k\right) + \frac{\tau^2(1-t)^2}{2}f\prime\prime\left(p_i^k\right) \tag{24}$$

Moreover, it includes the populations of best weight value, average weight value, worst weight, standard deviation, and several analyses. After initializing the set of populations, the objective values are estimated to sort the particles in ascending order. Based on this, the position of particles is updated to evaluate the best optimum solution. The key benefits of using this technique are increased convergence speed, providing the optimal solution with a minimum number of iterations, and handling complex discrete problems. The operational flow of the NMHA technique is represented in Fig. 11.

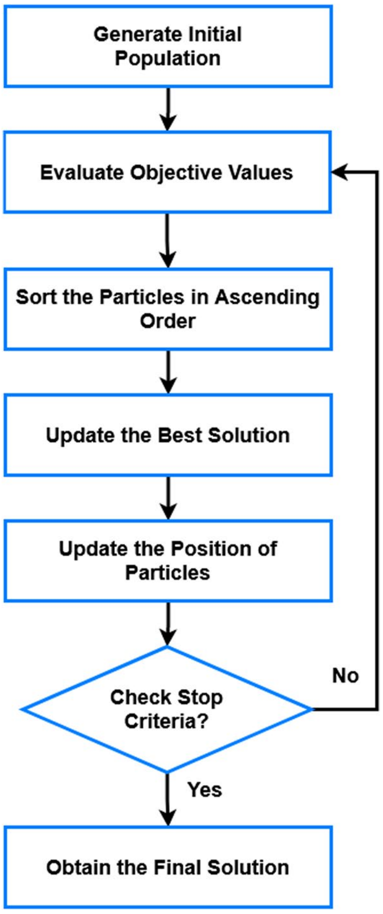

Generate Initial Population
Evaluate Objective Values
Sort the Particles in Ascending Order
Update the Best Solution
Update the Position of Particles
Check Stop Criteria?
No
Yes
Obtain the Final Solution

Fig. 11 Flow of NMHA

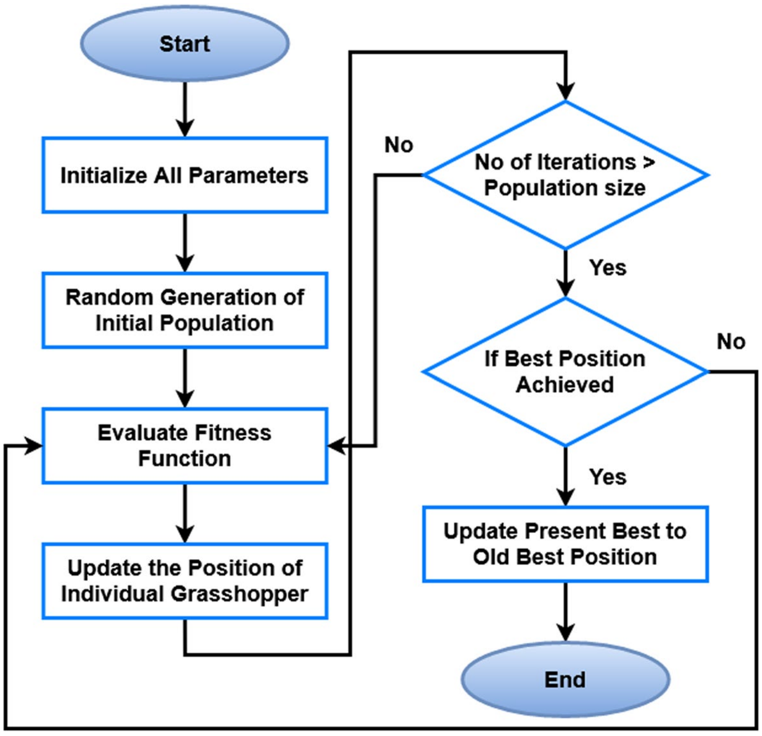

Start
Initialize All Parameters
Random Generation of Initial Population
Evaluate Fitness Function
Update the Position of Individual Grasshopper
No of Iterations > Population size
No
Yes
If Best Position Achieved
No
Yes
Update Present Best to Old Best Position
End

Fig. 12 Working flow of GSO

3.9 Grasshopper Optimization Algorithm (GOA)

Grasshopper Optimization (GO) (Alsammarraie and Hussein 2020) is also one of the bio-inspired methodologies that imitate the foraging behaviour of grasshopper insects. In this technique (Bhukya and Nandiraju 2020), the searching problem is split into the categories of exploitation and exploration, in which the search agents can abruptly move during exploitation. The flow of the GOA technique is graphically illustrated in Fig. 12, where the swarming behaviour of grasshoppers is estimated by using the following model:

$$P_i = D_i + G_i + W_i \tag{25}$$

where, P_i indicates the position of grasshopper, D_i denotes the social interaction, G_i is the gravitational force, and W_i represents the advection of wind. Based on this model, the random behavior of populations is estimated as follows:

$$P_i = r1D_i \times r2G_i \times r3W_i \tag{26}$$

where, $r1$, $r2$, and $r3$ are the random numbers [0,1]. Consequently, the social interaction is updated as follows:

$$D_i = \sum_{j=1;i\neq 1}^{n} f(c_{ij})\hat{c}_{ij} \tag{27}$$

where, c_{ij} indicates the distance across the grasshoppers of i and j, and f is the function of social force. The distance is estimated as follows:

$$\hat{c}_{ij} = \frac{(P_j - P_i)}{\text{c}} \tag{28}$$

Moreover, the function of social forces is updated as illustrated below:

$$f(r) = ae^{\frac{-r}{h}} - e^{-r} \tag{29}$$

where, a is the attraction intensity, and h is the scale length. By using the unit vector, the interaction level among the grasshoppers is estimated as follows:

$$P_i = \sum_{j=1;i\neq j}^{n} f\left(\left|P_j - P_i\right|\right)\frac{P_j - P_i}{c_{ij}} - k\hat{e}_c + q\hat{e}_w \tag{30}$$

where, k indicates the gravitational constant, $\hat{e}_c$ is the unit vector near the midpoint of earth, q denotes the constant drift, and $\hat{e}_w$ is the unity vector in the direction of the wind. Typically, the grasshoppers can reach their comfort zone fast, and it does not converge to the optimum point. Also, the above mathematical models are not directly used to solve optimization problems. The following updated equation form is used to get the optimal solution for solving the given problems.

$$P_i^c = s\left(\sum_{j=1,j\neq i}^{N} s\frac{u_c - l_c}{2}f(|P_j^c - P_i^c|)\frac{P_j - P_i}{c_{ij}}\right) + \widehat{G_c} \tag{31}$$

where, u_c and l_c are the upper and lower bounds at dimension c respectively, $\widehat{G}_c$ is the target value at dimension c, and s indicates the decreasing coefficient constant as estimated below:

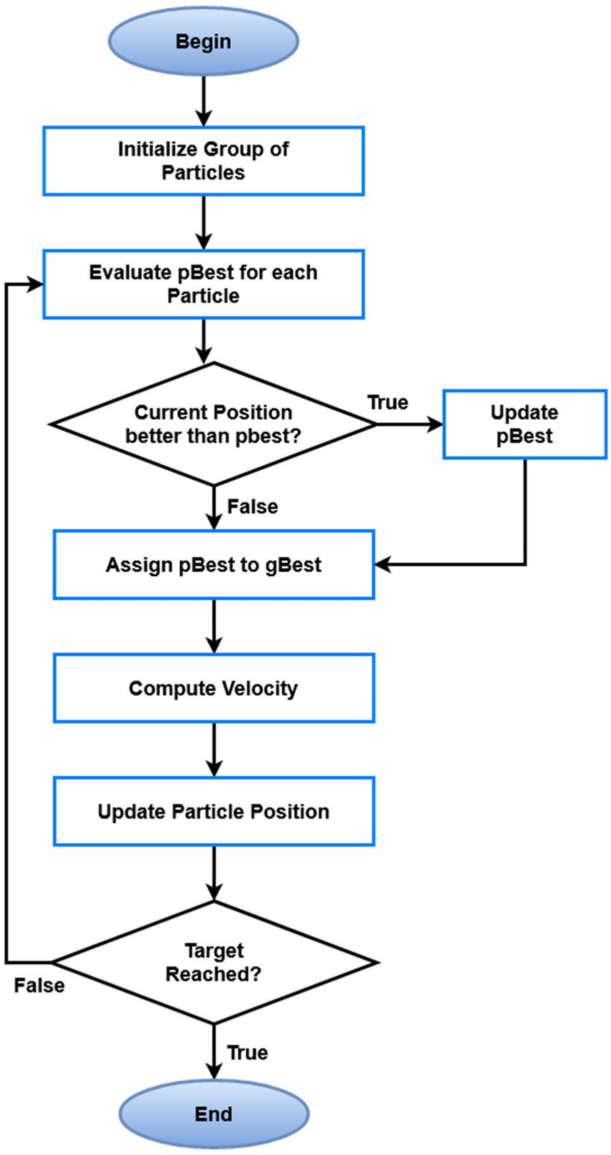
$$s = s_{mx} - m\frac{s_{mx} - s_{mn}}{m} \tag{32}$$

where, s_{mx} and s_{mn} denotes the maximum and minimum values of the current iteration, and m indicates the maximum number of iterations.

Algorithm 6 Grasshopper Optimization (GO)

```
Begin
1. Initialize the particles, maximum number of iterations $ct_{mx}$, constant minimum $h_{mn}$ and maximum values $h_{mx}$;
2. Then, the grasshopper population is randomly generated GS;
3. The fitness value is computed for each agent;
4. Y = bestagents;
5. while($ct < ct_{mx}$) do
   Update the constant value h;
      For i = 1ton
           The distance between grasshoppers GS are normalized at first;
           The step vector value is estimated and updated;
           The position of the current agent is updated;
    End for;
    ct = ct + 1;
   End while;
   Return the value of Y;
```

Fig. 13 Flow of PSO technique



3.10 Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) (Gopal et al. 2020) is a stochastic optimization technique increasingly used in different application systems to solve feature selection problems. Due to its increased convergence speed and reduced complexity (Aje and Josephat 2020), researchers highly prefer it for solving optimization problems. In this technique, the position and velocity of particles are estimated based on the particles' social and behavioural characteristics. Here, the current best particles in the searching space are obtained by updating the position of the particles. Other parameters are also required to be adjusted to get the optimal solution. The general flow of the PSO technique is shown in Fig. 13.

Algorithm 7 Particle Swarm Optimization (PSO)

```
Step 1:  At first, initialize the set of populations with random positions, velocities, on the $D$ dimensional searching
         space;
Step 2:  The optimized fitness function is estimated for each particle in D variables;
Step 3:  Estimate the particle fitness function (pbest);
         If (Current Value > Particle Fitness Value)
                 Set $pbest = Current\_Value$;
         End If;
Step 4:  Find out the best particles in the neighborhood, and assign the index variable;
Step 5:  Update the velocity and position of particles;
Step 9:  If (Criteria is satisfied)
                 Exit the loop;
Step 10:         End loop;
Step 11:         Return the best position;
```

3.11 Cuckoo Search Optimization (CSO)

Mareli et al. (2018) suggested an adaptive Cuckoo Search Optimization (CSO) technique for solving the multi-objective problems. Due to its global optimization capability, the method has been increasingly utilized in many application domains to obtain the best global solution. This work has validated and compared other bio-inspired algorithms like PSO, BAT, and SA according to their convergence speed and computational efficiency. As shown in Fig. 14, the local and global random walk parameters have been estimated using the scaling factor, heavy side function, random permutation, and levy distribution function (Yu et al. 2020). The probability distribution function is computed with the switching parameter and exponential function. However, the algorithm's performance was highly dependent on the dynamic value update of the switching parameter. Shehab et al. (2017) investigated the variants and applications of using the CSO techniques, which include the following approaches:

- Dynamic Cuckoo Search Algorithm (DCSA)
- Modified Cuckoo Search Algorithm (MCSA)
- Quantum Inspired Cuckoo Search Algorithm (QICSA)
- Adaptive Cuckoo Search Algorithm (ACSA)
- Enhanced Cuckoo Search Algorithm (ECSA)

Moreover, the performance and utilization of these algorithms are assessed in different domains of applications such as medical, clustering, engineering design, mining, and benchmarking. Based on this investigation (Joshi et al. 2017), it is analyzed that the CSCBOA has been increasingly utilized in many prediction systems compared to other optimization techniques like PSO, EA, and DE.

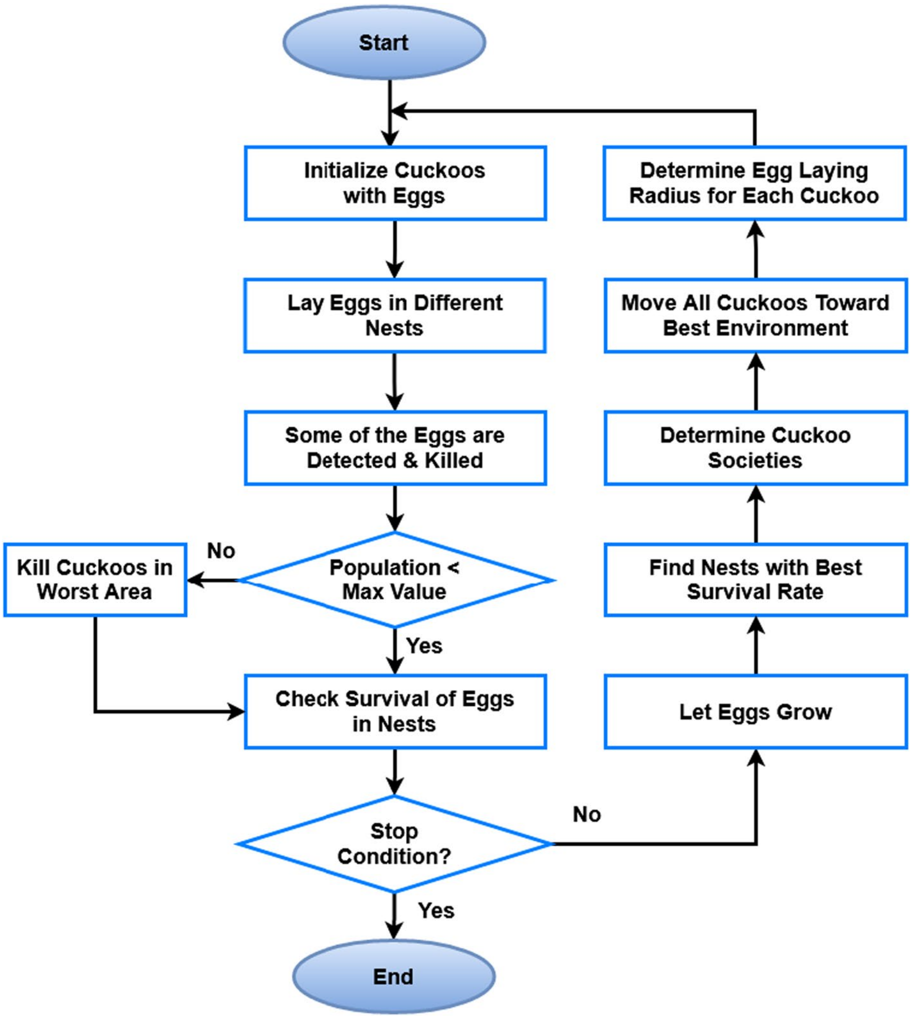

Start
Initialize Cuckoos with Eggs
Lay Eggs in Different Nests
Some of the Eggs are Detected & Killed
Population < Max Value
No
Kill Cuckoos in Worst Area
Yes
Check Survival of Eggs in Nests
Stop Condition?
No
Yes
End
Let Eggs Grow
Find Nests with Best Survival Rate
Determine Cuckoo Societies
Move All Cuckoos Toward Best Environment
Determine Egg Laying Radius for Each Cuckoo

Fig. 14 Flow of CSO technique

Rakhshani et al. (2017) suggested a snap-drift cuckoo search optimization technique for increased convergence speed and robustness. The learning strategy has been utilized to validate the online trade-off between the global and local search operators. Moreover, the state-of-the-art optimization techniques are validated and compared in this study based on the measures of success ratio, best/worst fitness value, run time, and mean and median values.

Algorithm 8 Cuckoo Search Optimization (CSO)

```
Step 1: Compute the objective function as follows:
        $O(f), f = (f_1, f_2 \dots f_d)^K$
Step 2: Initialize the set of population with $N$ host nests as represented below:
        $fi(i = 1, 2, 3 \dots N)$
Step 3: While $(m < M_G)$ // $M_G$ – Maximum generation
          Randomly select the cuckoo $(i)$ based on the Levy Distribution function;
          Estimate the fitness function $F_{F1}$;
          Randomly select a nest among $n$ (say $j$)
          Estimate the fitness function $F_{F2}$;
          $If\ (F_{F1} > F_{F2})$
              Replace the value $F2$ based on the new solution;
          End;
          A fraction of $(W_a)$ of worse nests are abandoned and new nests are constructed in new locations based on
          Levy flights;
          Obtain the best solution;
          Rank the solution with best fitness value;
        End while;
Step 4: Return the best optimal solution.
```

Harris Hawks Optimization (HHO):

- Exploitation: HHO imitates the surprise pounce method of Harris hawks for exploitation by concentrating on moving towards the prey (best solution) through multiple phases of cooperative hunting.
- Exploration: The exploration part is improved by the varied tactics used by hawks, such as soft besiege and hard besiege. These help to prevent the algorithm from becoming trapped in local optima.
- Diversity: HHO maintains diversity by introducing random changes in how the hawk acts, making sure that a significant area for search is covered.

Mayfly Optimization Algorithm (MOA):

- Exploitation: MOA blends genetic algorithm principles with particle swarm dynamics. This improves the local search and aims for convergence to find the best solutions.
- Exploration: The dynamics of the swarm and genetic operations (mutation and crossover) help us thoroughly explore the search space.
- Diversity: The inclusion of GA guarantees genetic diversity, and mayflies' group behavior sustains varied population dynamics.

Coronavirus Optimization Algorithm (COA):

- Exploration: In COA, the survey considers the exploitation of the search space through virus-like spread and mutation, concentrating on intensifying the search in hopeful regions.
- Exploration: The infection and mutation processes help the algorithm explore diverse search space areas.
- Diversity: The viral mutation mechanism ensures that the population remains diverse, preventing premature convergence.

Water Strider (WS) Algorithm:

- Exploitation: The WS algorithm exploits the search space by mimicking how water striders search for food, emphasizing searching close to water surfaces and intensifying the process.
- Exploration: It explores through random walks on water, guaranteeing a thorough search area coverage.
- Diversity: The random walks and interactions among water striders help maintain population diversity.

Black Widow Optimization (BWO):

- Exploitation: BWO imitates how black widow spiders act as cannibals, intending to remove weaker solutions from the search space and move towards stronger ones.
- Exploration: It can explore various parts by imitating mating and the subsequent offspring, bringing fresh genetic content to the population.
- Diversity: Mating and generating offspring guarantees genetic diversity and coverage in a broad search space.

Sailfish Optimization (SO):

- Exploitation: SO is using the search space by mimicking the predatory actions of sailfish and concentrating on organized hunting techniques.
- Exploration: The algorithm is exploring, much like the sailfish, with its lively and quick motion allowing it to search different areas effectively.
- Diversity: The dynamic behavior and interaction among sailfish maintain a diverse population.

Newton Meta-Heuristic Algorithm (MHA):

- Exploitation: MHA exploits the search space through Newton's laws of motion, improving solutions according to their velocities and positions.
- Exploration: Exploring the search space is carried out by adjusting the velocities of particles, which helps maintain a wide-ranging search capacity.
- Diversity: The constant adjustments in particle velocities help maintain diversity within the population.

Spider Monkey Optimization (SMO):

- Exploitation of Search Space: Here, SMO mimics spider monkeys' social foraging tactic by exploiting recognized food resources.
- Exploration: It simulates the movement and communication among different monkey groups.
- Diversity: The social structure and group interaction ensure a diverse search strategy.

Grasshopper Optimization Algorithm (GOA):

- Exploitation: GOA exploits the search space by imitating grasshoppers' swarming behavior. This enhances exploration near promising areas.

- Exploration: The algorithm explores by simulating grasshoppers' random and dynamic movement.
- Diversity: The random interactions and swarming behavior maintain population diversity.

Bee Colony Optimization (BCO):

- Exploitation: BCO exploits the search space through simulating a bee-like foraging behavior, concentrating its efforts on intensifying the search around potential nectar sources.
- Exploration: The algorithm explores by simulating the scout bees' search for new nectar sources.
- Diversity: The balance between scout and forager bees maintains diversity in the search process.

Flower Pollination Algorithm (FPA):

- Exploitation: FPA exploits the search area by copying how flowers are pollinated, focusing on local pollination, to better converge towards optimal solutions.
- Exploratory: This is similar to how pollination happens worldwide, with pollen being dispersed across the search space.
- Diversity: The combination of local and global pollination ensures diverse solution exploration.

Bat Algorithm (BA):

- Exploitation: BA exploits the search space by mimicking bats' echolocation, focusing on better sound wave solutions.
- Exploration: The algorithm explores through a process that mimics bats' random movement and sound wave emission.
- Diversity: The echolocation and movement strategies maintain diversity within the population.

Firefly Optimization (FFO):

- Exploitation: In this step, FFO imitates the attraction behavior of fireflies by focusing on moving towards the brightest (best) solutions found in the search space.
- Exploration: It explores by simulating fireflies' random movement and light intensity variations.
- Diversity: The varying light intensities and attraction mechanisms ensure diverse search space coverage.

Figure 15 shows the number of novel or newly developed optimization algorithms with their appropriate category from 2020 to 2021. Most newly created meta-heuristics fall into evolution/swarm-based algorithms and math/physics-based algorithms, as depicted in this chart. In addition, as illustrated in Fig. 16, around 47 original meta-heuristic models have been accounted for in 2022 alone. The trend line in this figure, with a coefficient of determination (R^2) = 0.926, is significantly rising (Rajwar et al. 2023b). The statistic used to assess a model's fit quality is called R^2. When the trend line's R^2 value is at or close to 1,

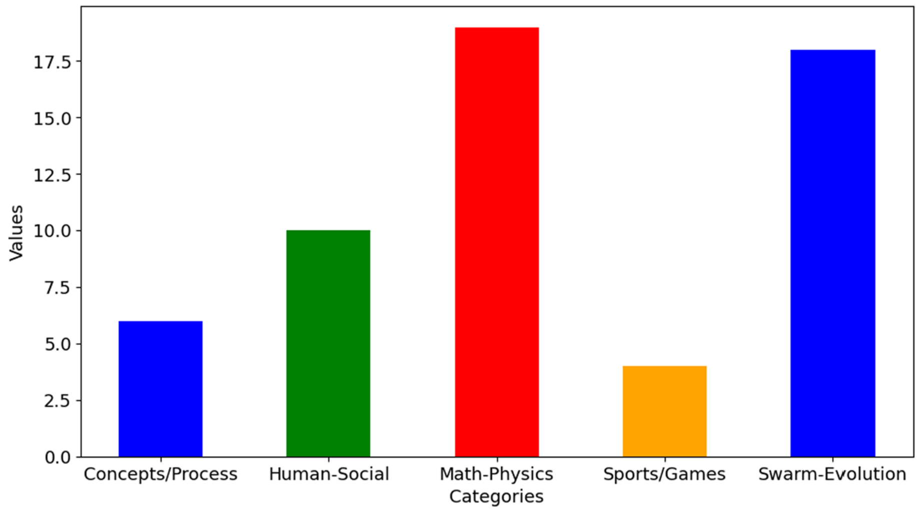


Fig. 15 Number of heuristics based on category

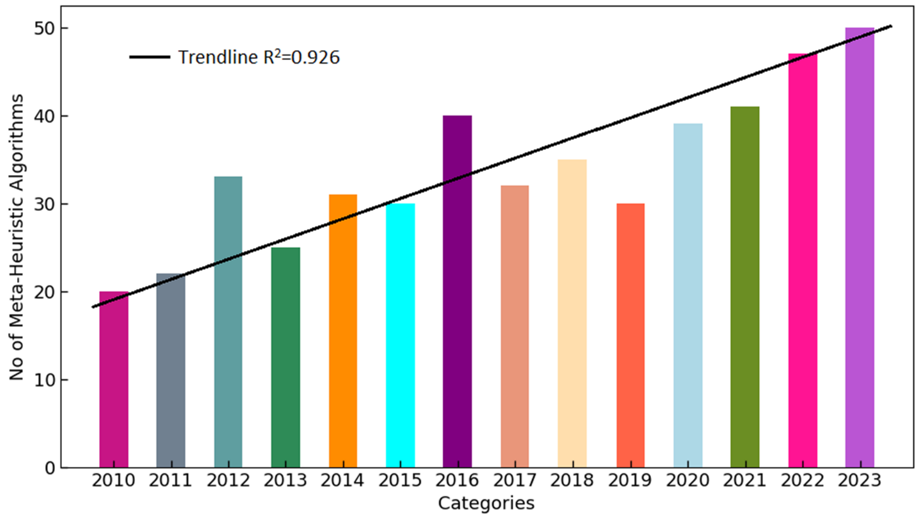


Fig. 16 The cumulative number of meta-heuristic algorithms developed in the past few decades

it is deemed most trustworthy. The substantial R^2 valuation indicates that the number of "unique" meta-heuristic models being developed is rising quickly. The distribution of new optimization algorithms in various categories from 2020 to 2021 is shown in Fig. 15. This figure mainly divides these fresh meta-heuristics into evolution/swarm-based algorithms and math/physics-based algorithms. Evolution/Swarm-Based Algorithms: These algorithms are inspired by how evolution naturally happens in biology and the collective behavior of swarms or groups in nature. For example, Genetic Algorithms (GA), Particle Swarm

Optimization (PSO), Ant Colony Optimization (ACO) and similar strategies copy biological evolution and mimic social bird behavior together with fish' movements as well as ants among other living organisms. According to Fig. 15, many newly created algorithms fall under this category during this period. This shows us how much interest and innovation there remains in techniques using natural occurrences for optimizing processes.

Math/Physics-Based Algorithms: This group contains algorithms from mathematical theories, physics principles or physical occurrences. Simulated Annealing (SA), Quantum-inspired algorithms, and algorithms based on concepts such as gravity, electromagnetism, and thermodynamics are a few examples of this category. The illustration depicts how numerous novel meta-heuristics also fit under this kind of algorithm, showing solid research work and the use of mathematical plus physical ideas for optimization problems.

The chart shows that these two kinds are most commonly used for creating new meta-heuristics. This demonstrates the wide-ranging usefulness and success of using methods inspired by biology or based on mathematical/physical rules to handle complex optimization problems. The illustration in Fig. 15 reflects a constant and increasing fascination with applying both biological models and mathematical/physical principles for inventing and enhancing optimization methods. To conclude, Fig. 15 shows that between 2020 and 2021, most new optimization algorithms were put in groups like evolution/swarm-based and math/physics-based algorithms. This demonstrates ongoing progress and the frequent use of these methods within the meta-heuristics area.

Figure 16 is a chart that displays the total number of meta-heuristic algorithms made during these past decades, showing how much growth has been in creating these optimization techniques. This figure shows data about 2022, and it suggests around 47 new meta-heuristic models were made in just that year. The line in Fig. 16 that represents the trend is going up a lot, shown by its coefficient of determination (R^2) value being 0.926. The R^2 value is a statistical measure for assessing how well fit a model has been made with data points. An R^2 value near to 1 means there's very high confidence in the reliability of this trend line to represent these data points accurately. Context, R^2 value of 0.926 shows that the model employed to describe a rising trend in meta-heuristic algorithm development is trustworthy and precisely matches real data. The R^2 value is big, showing that many different meta-heuristic models are being made. The graph's trend shows a growing interest and constant innovation in meta-heuristics. This shows how people who study or practice this field keep trying to find new, good ways to optimize techniques. It is important because it shows the active and changing character of meta-heuristic research with more and more new algorithms used to solve difficult optimization problems in many areas of study or work. Moreover, it shows the gradual rise in meta-heuristic algorithm development during these last few decades. There is a significant jump in 2022 that stands out from other years. The strong R^2 value of 0.926 shows a dependable and important upward pattern, indicating an increasing rate of development in meta-heuristic optimization.

4 Results analysis

This section evaluates and compares the performance and efficiency of different bio-inspired optimization algorithms using the standard benchmarking functions. Recent optimization techniques include HHO, BWO, MOA, WSA, COA, FBA, SO, SMO, NMA, and GOA. Figure 17 shows the comparative analysis of these optimization techniques based on their functions, search history, trajectory plots, and average fitness value. Figure 17a shows

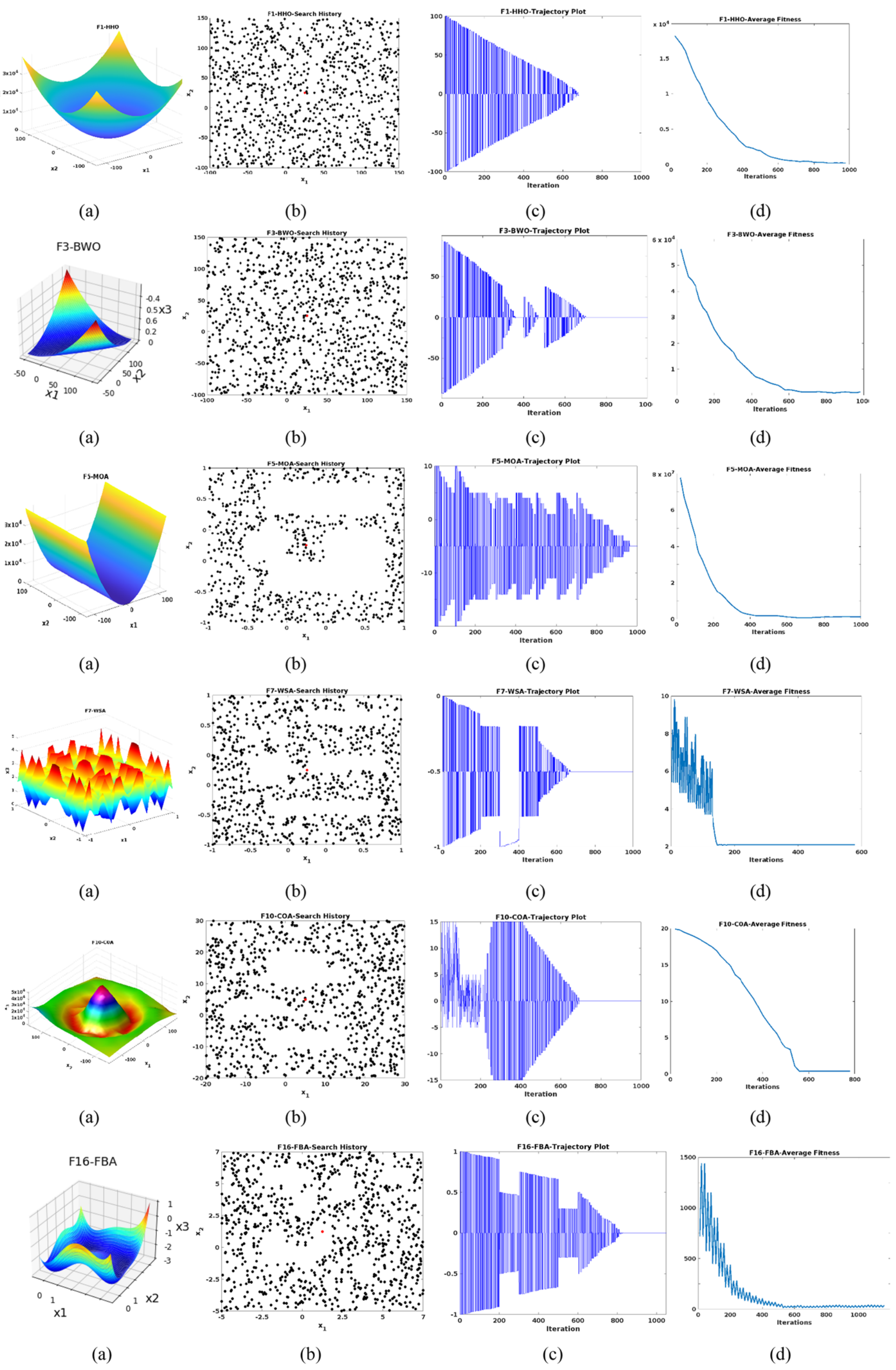


Fig. 17 Performance analysis of various optimization techniques (**a**). Benchmark testing functions (**b**). Search history (**c**). Trajectory plot, and (**d**). Average fitness

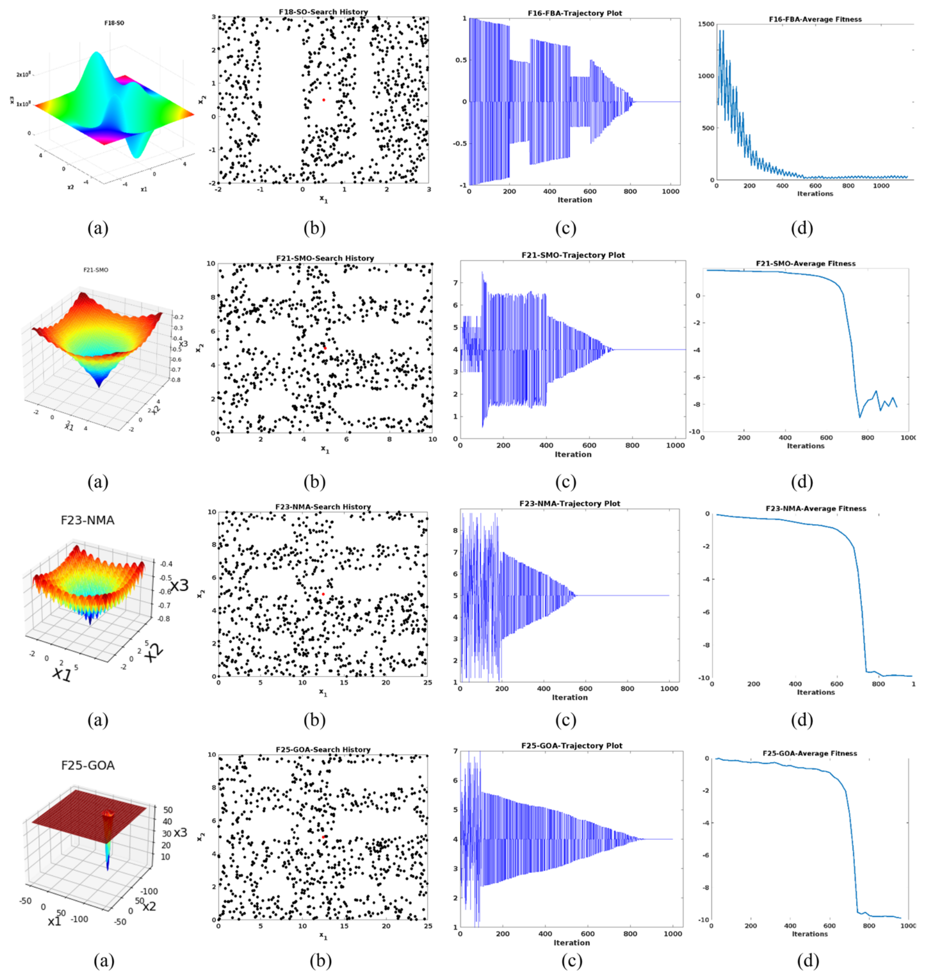


Fig. 17 (continued)

the outcomes of optimization methods on a basic group of benchmark functions. The results are displayed as final solution quality (for example: minimum value discovered) for every benchmark function, emphasizing the capability of each method to navigate through various optimization environments effectively. The subfigure of search history shows us how the optimization process changes with time, presenting the best solution for every iteration and algorithm. This visual aid aids in comprehending algorithms' convergence behavior: it demonstrates how fast they enhance solutions. A sharp decline shows quick progress, but a gentler slope might suggest slower coming together or trouble getting out of small, best solutions. The trajectory plot subfigure shows the tracks that algorithms follow in the search area. When you choose one benchmark function, this plot shows where different optimization methods start and how they move to reach their final solution. The trajectory graph helps us see how algorithms explore and use up resources, showing us their path through the solution space for finding optimum solutions. The average sub-fitness is the typical fitness value every optimisation technique reaches across many attempts. It gives a statistical explanation of performance, displaying the average and scatter of fitness values. This measure assists in determining how dependable and strong these algorithms are; higher consistency demonstrates their capability to discover superior solutions consistently during various runs.

Typically, the performance of the optimization technique is determined based on its convergence rate and average fitness value. Hence, this analysis examines and compares the interpretation of these techniques according to their average fitness value. Moreover,

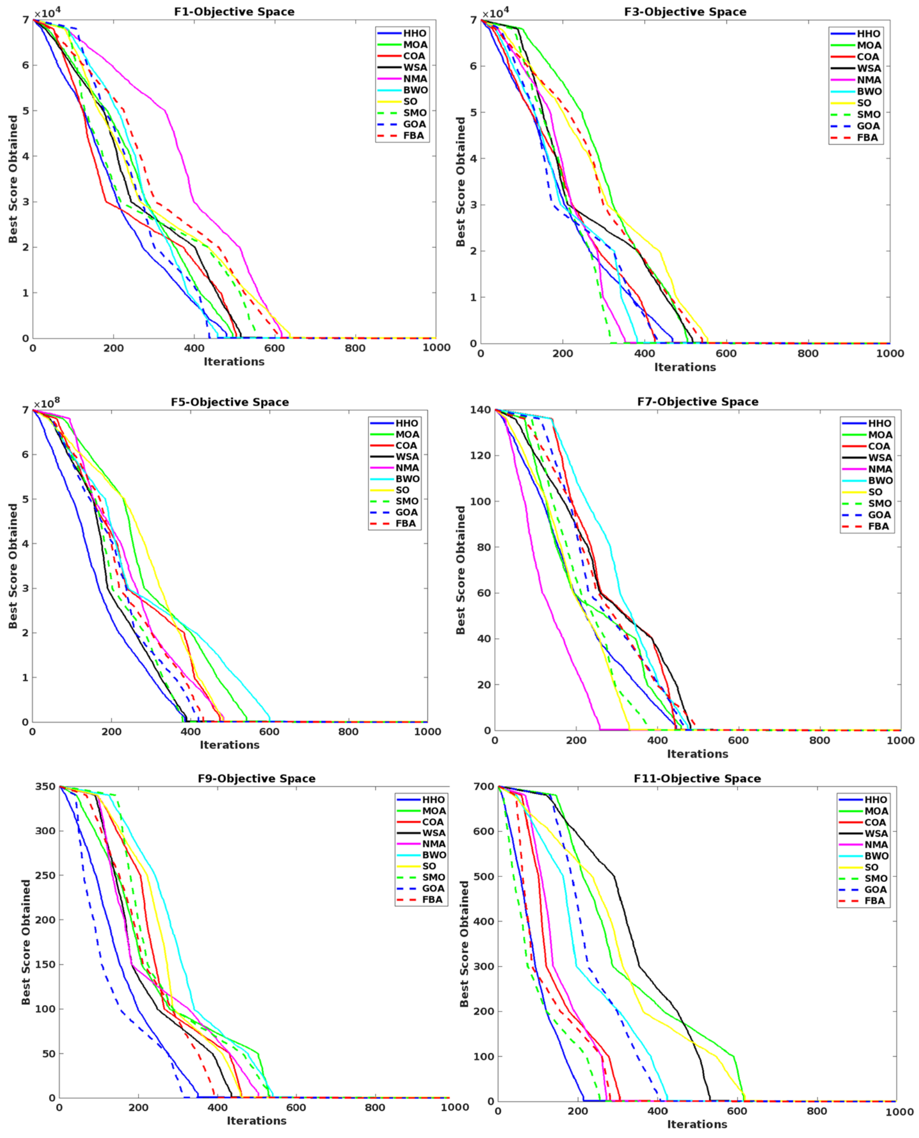


Fig. 18 Convergence analysis of various optimization techniques with different benchmark testing functions

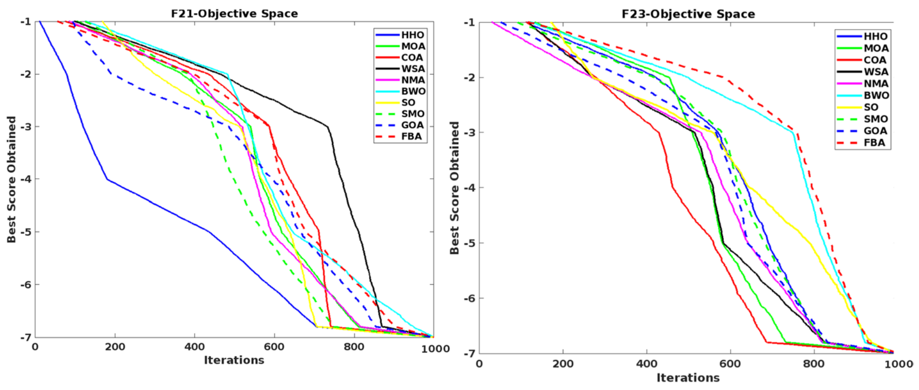


Fig. 18 (continued)

the convergence analysis is demonstrated and represented in a 2D environment, as shown in Fig. 18. Next, the search history is estimated to analyze the location history of optimization, where the unimodal test functions are computed to distribute the samples in the searching space sparsely. Based on this illustration, the exploitation and exploration capabilities of the optimization technique are analyzed. Then, the trajectory is estimated to demonstrate the abrupt changes in the optimization. The results show that the swarm-based methodologies could eventually converge the points in the searching space. Then, the average fitness is estimated to analyze the average objective value of the optimization at each iteration. Using this analysis, the descending behavior of the optimization techniques is validated, which also helps evaluate the accuracy of optimum function generation.

Figure 18 shows the convergence analysis of all optimization techniques based on the testing functions, including F1, F3, F5, F7, F9, F11, F21, and F23. The algorithms' convergence speed and better performance are validated and analyzed based on this analysis. Figure 18 shows the convergence behavior of different bio-inspired optimization techniques applied to several benchmark testing functions. It demonstrates how fast and well each algorithm approaches the best solution after some iterations. Each subplot's convergence curve represents the top fitness value reached by algorithms during time, and steeper curves point to quicker convergence rates. For example, in the beginning, algorithms that quickly decrease their fitness values show high performance at early stages—they can easily escape from local optimum points while also focusing on finding global ones. On the other hand, algorithms that have gentler slopes might show slower coming together. This could be because it is not easy to balance exploring new options and exploiting already discovered ones. The comparison of these methods shows us their good parts and weak points, helping us understand how effective and dependable they are in solving difficult optimization problems on different types of landforms. Similar to that, the scalability of optimization techniques is validated to determine whether the methods could handle large-scale optimization problems. Then, the obtained results are graphically shown in Fig. 19. Based on this evaluation, the increased scalability and robustness of the optimization techniques are assessed. Finally, as shown in Fig. 20, the CEC 2015 benchmark test functions have been utilized to validate the performance of the optimization techniques. Using this box plot, the best optimizer of all testing functions is identified, and the method could satisfy all these measures by having the exploration and exploitation capabilities. According

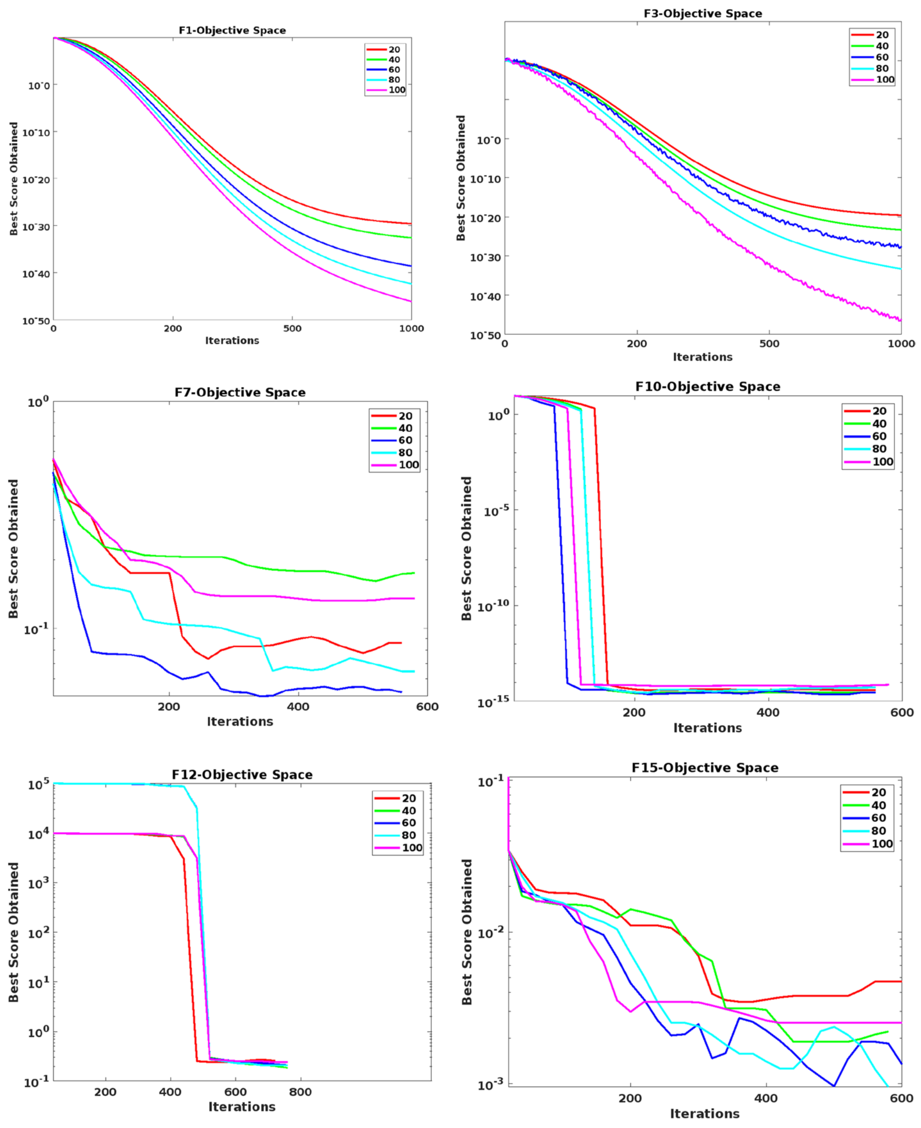


Fig. 19 Objective score estimation of various optimization techniques with different benchmark testing functions

to the analysis, it is observed that the HHO technique provides improved performance results over the other optimization techniques. Also, it has an increased ability to handle large and complex optimization problems with increased convergence speed and best fitness function.

Figure 20 compares different bio-inspired optimization methods using the CEC (Congress on Evolutionary Computation) benchmark functions. These functions are famous for their strict and thorough evaluation standards. Every subplot in the figure shows a distinct CEC benchmark function, covering many kinds of optimization difficulties such as unimodal and multimodal landscapes, separable or non-separable functions etcetera. These functions are made especially to test how strong, efficient and adjustable an optimization algorithm is when faced with various complicated situations. The performance of every optimization technique is shown visually, using measurements like the best fitness value reached, convergence speed and reliability in numerous runs. This comparison clarifies

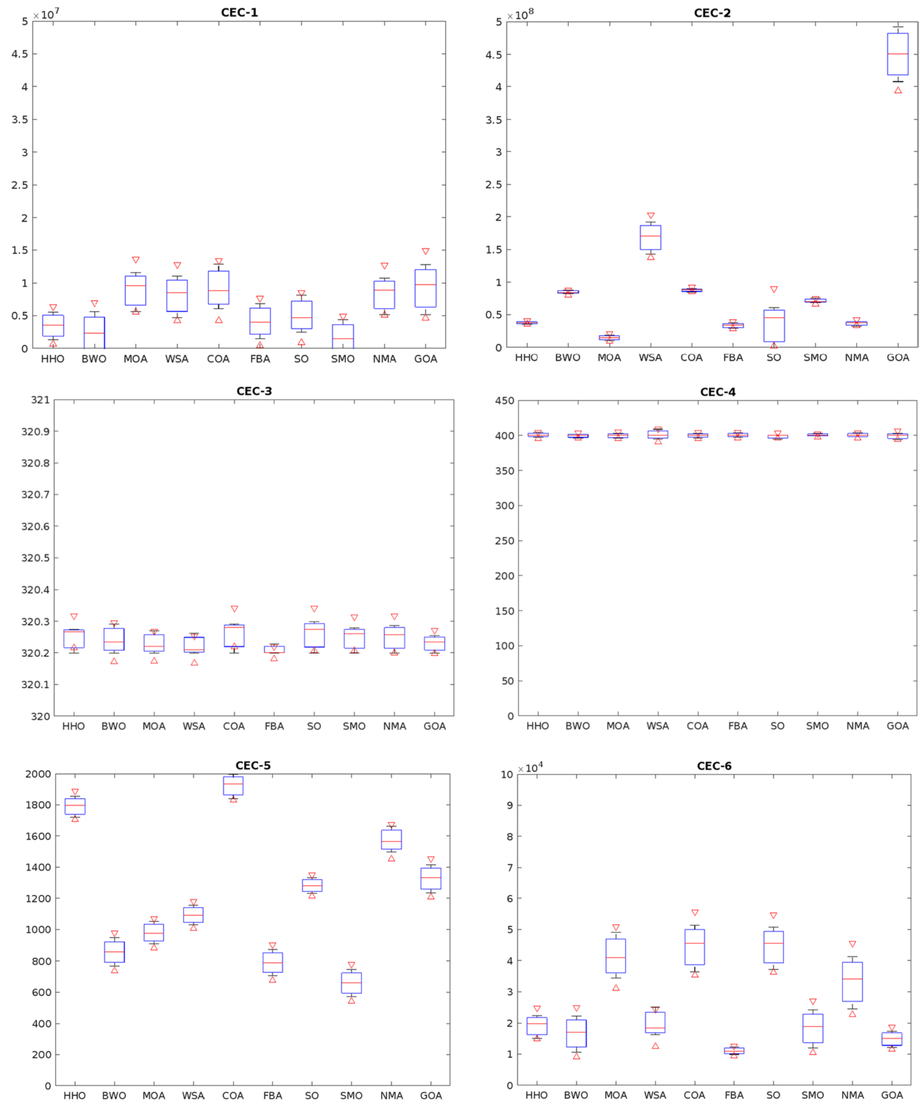


Fig. 20 Comparative analysis among the optimization approaches based on CEC benchmark

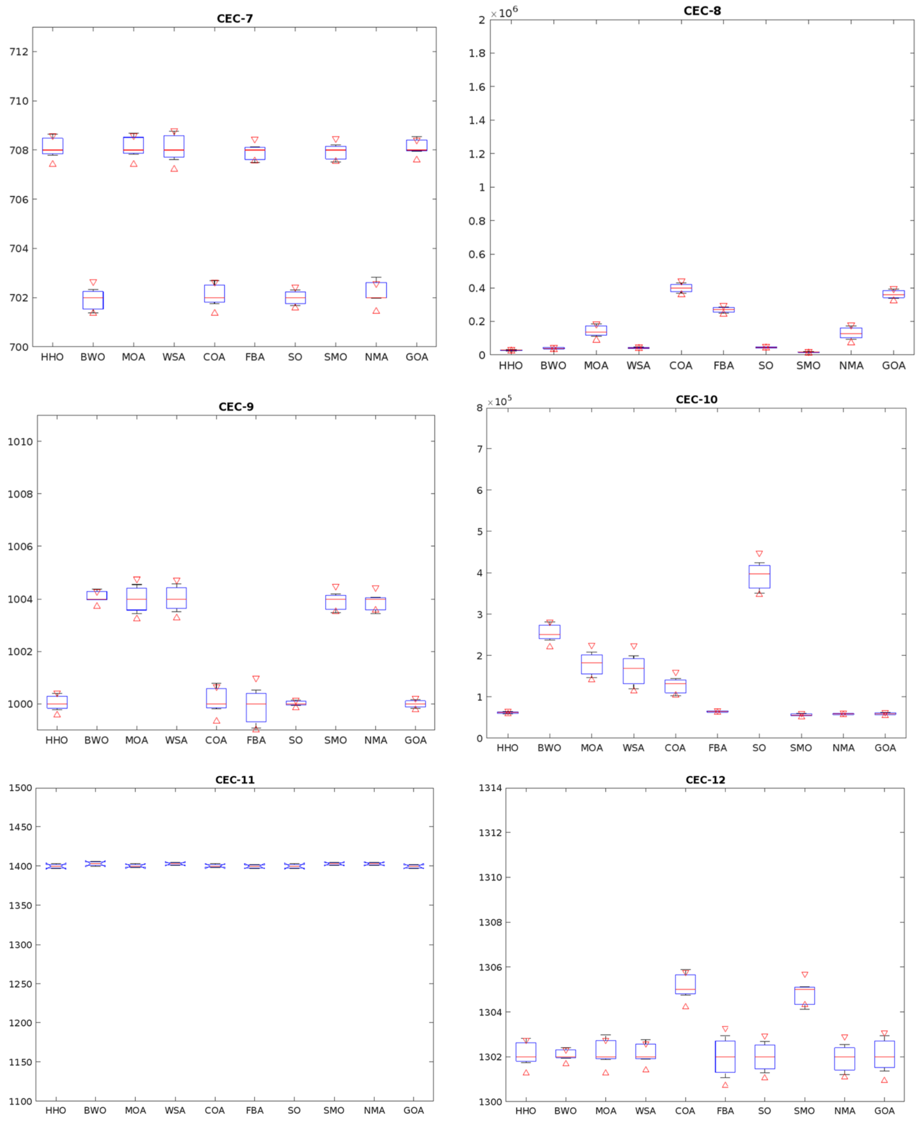


Fig. 20 (continued)

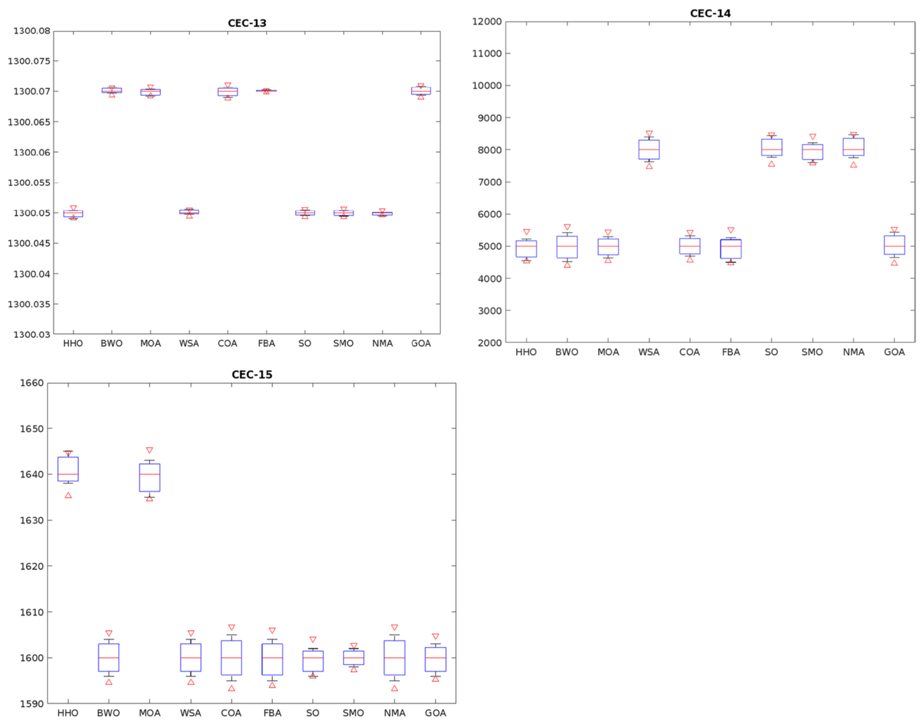


Fig. 20 (continued)

which algorithms are best suited for different optimization problems. It also helps us understand what areas they shine in and where their limitations lie. The results are systematically compared to each other, as shown in Fig. 20. This type of analysis helps us understand how various algorithms perform when dealing with tough CEC benchmarks—making it easier to see their effectiveness on complex and diverse optimization tasks.

5 Discussion

This section explores the experiments carried out in this paper, thoroughly examining the different optimization strategies used. The survey contrasts these techniques with current methods, emphasizing the benefits and drawbacks of each and talking about any particular issues found. Harris Hawks Optimization (HHO), Mayfly Optimization Algorithm(MOA), Coronavirus Optimization Algorithm (COA), Newton Meta-Heuristic Algorithm (MHA), Water Strider Algorithm(WS), Black Widow Optimization (BWO), Sailfish Optimization (SO), Spider Monkey Optimization (SMO), Grasshopper Optimization Algorithm (GOA), Fractal Based Algorithm (FBA), Bee Colony Optimization (BCO), Rainfall Optimization Algorithm (ROA), Dragon Fly Optimization (DF), Water Wave Optimization (WWO), Ant Lion Optimization (ALO), Symbiotic Organisms Search (SOS), and Egyptian Vulture Optimization Algorithm. The great adaptability and strong global search abilities of these bio-inspired algorithms come from using processes and behaviors found in nature. Their natural mechanisms keep population variety, lowering the chance of early coming together or convergence, making them good at dealing with complicated optimization areas. But there are also some downsides to these benefits: they have high computational complexity and can be quite sensitive to how you set their starting parameters—this may impact their performance and speed of convergence. Many of these algorithms, especially the ones inspired by nature, need big computational power and time because they could stop too early in certain situations like complex or multimodal search spaces. Also, problems with scaling can happen, which means it is not easy to use these algorithms on extremely large or high-dimension problems. But even when there are difficulties working with them due to their demanding nature computationally and possible issues of scalability for very large size problems- the survey still recognizes that bio-inspired algorithms have strong characteristics like flexibility and consistent performance over a variety of landscapes of problem areas which emphasize their importance as effective optimization instruments. In the future, more studies could concentrate on enhancing computational efficiency. This might include finding new ways to make algorithms quicker and using parallel computing methods. Also, developing adaptive parameter mechanisms that adjust their qualities automatically based on problem complexity or data characteristics would likely improve current approaches significantly. Another possible direction for future exploration is to create hybrid methods by combining multiple optimization algorithms. These hybrids could be designed to use different techniques in a complementary way, with each one compensating for the limitations of others. They might also incorporate machine learning elements that dynamically adapt based on progress made during iterations or available data patterns. Such studies can help overcome some of the limitations and further improve these methods' efficacy in handling complex optimization tasks across different application areas.

This paper stands out because it concentrates on the most recent developments in bio-inspired optimization techniques among a wide range of meta-heuristic algorithms. While many reviews could evaluate current methods, this study dives deep into forefront research by examining each method's unique characteristics, optimization properties and operational paradigms. The goal is to find groundbreaking ways to solve problems. One important aspect is the focus on the growing recognition of how well bio-inspired methods can handle complicated engineering issues. The supporting proof from fast convergence rates and unmatched fitness scores points to a change in optimization strategies. The paper seems to give a lot of comparative analysis. It compares the performance of these new methods with normal benchmarks, such as search history, trajectory plots and fitness functions. The aim is to show that these bio-inspired approaches are better than traditional ones through this analysis. The decision to concentrate on bio-inspired optimization strategies is especially important considering the current situation of fast industrialization, which creates a strong need for efficient optimization solutions. Using natural happenings to guide optimization techniques provides hopeful routes for dealing with increasingly complex engineering issues. In general, the paper wants to be like a lighthouse for scientists who are looking for fresh answers that come from many natural ways. It shows not only the big changes bio-inspired optimizers can make but also points out possible new areas of study in coming times to perfect and increase these methods more. Horse Herd Optimization (HHO) might be better than other meta-heuristic algorithms because it combines biology features and various search methods. Taking ideas from the social activities and hunting habits of horse herds, HHO uses the natural abilities shown in group thinking and adjusting tactics. By imitating how decisions are made non-centralised, and communication happens within horse herds, HHO promotes strong investigation and uses search areas to maximum advantage. HHO also uses different types of search operators and mechanisms. These include random exploration, local exploitation, and sharing global information which helps it to move around complicated optimization landscapes that are constantly changing. HHO has a complex approach that helps find a balance between exploring new solutions and taking full advantage of promising regions. This assists with quick coming together towards optimal or almost perfect solutions. The built-in ability to scale up and the flexible nature of HHO make it easy to use in various problem areas and solution spaces. This characteristic improves its performance compared to other meta-heuristic algorithms, too. In general, because of the good combination of biology's motivation and flexible search methods, HHO is strong at handling optimization problems. This makes it a powerful tool for dealing with real-life issues in different areas.

The study has illustrated the wonderful potential and application of bio-inspired optimization algorithms, yet several open research questions (ORQs) remain that require further investigation. By exploring these matters, the survey enhances understanding of these algorithms and their functioning while dealing with complex optimization problems.

- Scalability and Efficiency: The main difficulty is the large computational expense and scalability issues related to many bio-inspired algorithms. Future research should focus on developing more efficient versions of these algorithms that can handle big-scale and high-dimensional problems without significant performance loss. This may involve parallel methods, complicated data structures, or mixed techniques combining various algorithm strengths.
- Sensitivity of Parameters and Adaptation: Many algorithms show high sensitivity towards their starting parameter setups. This characteristic greatly impacts the performance of these methods. For upcoming times, the survey needs to study adaptable parameter mechanisms that allow these algorithms to modify their parameters during optimization procedure. This can help keep a good balance between exploration and exploitation, leading to better convergence rates.
- Premature Convergence: Bio-inspired algorithms are typically robust in problem landscapes, but they often deal with premature convergence difficulty. This problem is more frequent when dealing with complex or multimodal search spaces. In the future, research should focus on producing new ways to cease early convergence by employing methods such as incorporating mechanisms that boost variability and fresh escape tactics that assist the algorithms in steering away from local optima.
- Benchmarking and Comparative Studies: The lack of standardized benchmarking for fresh optimization formulas makes it difficult to evaluate their relative performance in a broader manner. In the future, research needs to concentrate on establishing a set of standard benchmarks and effectiveness measurements. This will create an environment where comparison becomes simpler when new algorithms are introduced with the already existing ones.
- Real-Life Uses: Theoretical progress is very important, but it's also difficult to use these algorithms in real-life situations. Next, studies must concentrate on testing and confirming the value of these algorithms in practical scenarios across different fields, such as healthcare, logistics, finance, or engineering. This would show how they can be applied practically and determine what changes are needed for particular areas.
- Machine Learning Collaboration: A hopeful path for research is linking bio-inspired optimization algorithms with machine learning methods. It could be very beneficial to study how these algorithms might enhance machine learning models, especially in

areas such as adjusting hyperparameters, selecting features, and training the model. This could lead to significant improvements in performance and effectiveness.

- Theoretical Foundations: More theory work is needed to understand bio-inspired algorithms' inner mechanisms and characteristics. A better theoretical understanding can help improve algorithms and predict their functioning in different types of optimization problems.
- In conclusion, responding to these open research questions will advance bio-inspired optimization algorithms. It will enhance their applicability and capacity to resolve practical issues. Focusing on these parts can assist future studies in addressing present shortcomings and paving improved ways for more robust, quicker, and flexible optimization methods.

6 Research gap

Even though there have been many improvements, some research gaps still exist in the area of bio-inspired optimization techniques. One big gap is the absence of standard methods for benchmarking. This makes it difficult to compare how well different algorithms perform correctly. If there are no universal standards for benchmarking that include a variety of test functions, problem situations and performance measurements, it becomes hard to evaluate these methods' effectiveness accurately. Moreover, even if bio-inspired algorithms show good results on small to medium-sized issues, they frequently have problems with scalability and computational effectiveness when used for huge or real-time optimization problems. This limitation highlights the need for more studies on how to improve these algorithms' scalability and efficiency. It could be done by combining them with other optimization methods, enhancing parallel processing abilities, or creating new algorithmic structures. Another important gap is the flexibility of bio-inspired algorithms in changing environments because many current methods are made for fixed problem situations and might not work well in scenarios where optimization conditions alter over time. As the survey explores further into bio-inspired optimization, it becomes evident that there are several areas where improvements could be made. These include creating more adaptable, scalable and effectively benchmarked techniques. This kind of work is important for developing and using these methods in complex real-life problems.

7 Conclusion

The primary purpose of this paper is to present a comprehensive analysis for validating and examining the performance and effectiveness of using various bio-inspired optimization techniques. Optimization methodologies are extensively used in all application systems, offering the best solutions to specific problems. Here, the standard benchmarking test functions are used to evaluate and compare the results of recent bio-inspired optimization techniques, including HHO, BWO, MOA, WSA, COA, FBA, SO, SMO, NMA, and GOA. Furthermore, the exploration and exploitation capabilities of these optimisation techniques are assessed based on the analysis. In this analysis, the working model, operating characteristics, advantages, and disadvantages of using different bio-inspired meta-heuristic optimization techniques have been discussed recently. Also, it presents a complete analysis of the optimization process and its importance and features. This analysis shows that the optimization performance depends on the increased convergence rate and optimal best fitness score. The results show that the evolutionary and swarm-intelligence-based bio-inspired optimization techniques are more suitable for solving complex problems, providing intelligent and appropriate solutions for the given problem with an increased convergence rate.

Author contributions All work done by first author and reviewed the manuscript.

Data availability This manuscript does not report data generation or analysis.

Declarations

Competing interests The authors declare that they have no competing interests.

Human/Animal Involvement We declare that no human or animal involvement is associated with this project.

References

Abdel-Basset M, Shawky LA (2019) Flower pollination algorithm: a comprehensive review. Artif Intell Rev 52:2533–2557

Abdullahi M, Ngadi MA, Dishing SI, Abdulhamid SIM, Usman MJ (2020) A survey of symbiotic organisms search algorithms and applications. Neural Comput Appl 32:547–566

Abualigah L, Diabat A, Geem ZW (2020) A comprehensive survey of the harmony search algorithm in clustering applications. Appl Sci 10:3827

Abualigah L, Gandomi AH, Elaziz MA, Hamad HA, Omari M, Alshinwan M et al (2021) Advances in meta-heuristic optimization algorithms in big data text clustering. Electronics 10:101

Adegboye OR, Deniz Ülker E (2023) Hybrid artificial electric field employing cuckoo search algorithm with refraction learning for engineering optimization problems. Sci Rep 13:4098

Adegboye OR, Feda AK, Ojekemi OS, Agyekum EB, Hussien AG, Kamel S (2024a) Chaotic opposition learning with mirror reflection and worst individual disturbance grey wolf optimizer for continuous global numerical optimization. Sci Rep 14:4660

Adegboye OR, Feda AK, Ojekemi OR, Agyekum EB, Khan B, Kamel S (2024b) DGS-SCSO: enhancing Sand Cat Swarm Optimization with Dynamic Pinhole Imaging and Golden Sine Algorithm for improved numerical optimization performance. Sci Rep 14:1491

Ahrari A, Essam D (2022) An introduction to evolutionary and memetic algorithms for parameter optimization. in Evolutionary and Memetic Computing for Project Portfolio Selection and Scheduling, ed: Springer, pp. 37–63

Aje OF, Josephat AA (2020) The particle swarm optimization (PSO) algorithm application–a review. Global J Eng Technol Adv 3:001–006

Al-Betar MA, Alyasseri ZAA, Awadallah MA, Abu Doush I (2021) Coronavirus herd immunity optimizer (CHIO). Neural Comput Appl 33:5011–5042

Almufti SM (2019) Historical survey on metaheuristics algorithms. Int J Sci World 7:1

Alsammarraie S, Hussein NK (2020) A new hybrid grasshopper optimization-backpropagation for feedforward neural network training. Tikrit J Pure Sci 25:118–127

Ammal RA, Sajimon P, Vinodchandra S (2020) Termite inspired algorithm for traffic engineering in hybrid software defined networks. PeerJ Comput Sci 6:e283

Aromolaran O, Oyelade J, Adebiyi E (2021) Performance evaluation of features for gene essentiality prediction. in IOP Conference Series: Earth and Environmental Science, p. 012019

Assiri AS, Hussien AG, Amin M (2020) Ant lion optimization: variants, hybrids, and applications. IEEE Access 8:77746–77764

Awadallah MA, Hammouri AI, Al-Betar MA, Braik MS, Abd Elaziz M (2022) Binary Horse herd optimization algorithm with crossover operators for feature selection. Comput Biol Med 141:105152

Aydogdu I, Ormecioglu TO, Tunca O, Carbas S (2022) Design of large-scale real-size steel structures using various modified grasshopper optimization algorithms. Neural Comput Appl 34:13825

Band SS, Janizadeh S, Chandra Pal S, Saha A, Chakrabortty R, Shokri M et al (2020) Novel ensemble approach of deep learning neural network (DLNN) model and particle swarm optimization (PSO) algorithm for prediction of gully erosion susceptibility. Sensors 20:5609

Banerjee S, Mitra S (2020) Evolving Optimal Convolutional Neural Networks. in 2020 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 2677–2683

Basu S, Kumar S, Basu M (2022) Horse herd optimization algorithm for economic dispatch problems. Eng Opt 55:806

Bhattacharyya T, Chatterjee B, Singh PK, Yoon JH, Geem ZW, Sarkar R (2020) Mayfly in harmony: a new hybrid meta-heuristic feature selection algorithm. IEEE Access 8:195929–195945

Bhukya L, Nandiraju S (2020) A novel photovoltaic maximum power point tracking technique based on grasshopper optimized fuzzy logic approach. Int J Hydrogen Energy 45:9416–9427

Boughaci D, Belaidi F, Kerkouche I (2020) A novel feature selection technique based on Roach Infestation Optimization for Internet Traffic Classification. in 2020 2nd International Conference on Computer and Information Sciences (ICCIS), pp. 1–5

Castelli M, Manzoni L, Mariot L, Nobile MS, Tangherloni A (2022) Salp Swarm Optimization: a critical review. Expert Syst Appl 189:116029

Chen H, Zhang Q, Luo J, Xu Y, Zhang X (2020) An enhanced bacterial foraging optimization and its application for training kernel extreme learning machine. Appl Soft Comput 86:105884

Chopard B, Tomassini M (2018) Particle swarm optimization. in An Introduction to Metaheuristics for Optimization, ed: Springer, pp. 97–102

Cruz-Duarte JM, Amaya I, Ortíz-Bayliss JC, Correa R (2021) Solving microelectronic thermal management problems using a generalized spiral optimization algorithm. Appl Intell 51:5622–5643

Datta S, Roy S, Davim JP (2019) Optimization techniques: an overview. Optimization in Industry, pp. 1–11

Del Ser J, Osaba E, Molina D, Yang X-S, Salcedo-Sanz S, Camacho D et al (2019) Bio-inspired computation: where we stand and what's next. Swarm Evol Comput 48:220–250

Deng W, Xu J, Zhao H (2019) An improved ant colony optimization algorithm based on hybrid strategies for scheduling problem. IEEE Access 7:20281–20292

Dokeroglu T, Sevinc E, Cosar A (2019) Artificial bee colony optimization for the quadratic assignment problem. Appl Soft Comput 76:595–606

Dorigo M, Stützle T (2019) Ant colony optimization: overview and recent advances. Handbook of metaheuristics, pp. 311–351

Dubey M, Kumar V, Kaur M, Dao T-P (2021) A systematic review on harmony search algorithm: theory, literature, and applications. Math Probl Eng 2021:1

Elmanakhly DA, Saleh M, Rashed EA, Abdel-Basset M (2022) BinHOA: efficient binary horse herd optimization method for feature selection: analysis and validations. IEEE Access 10:26795–26816

Feda AK, Adegboye M, Adegboye OR, Agyekum EB, Mbasso WF, Kamel S (2024) S-shaped grey wolf optimizer-based FOX algorithm for feature selection. Heliyon 10:e24192

Fu Y, Hou Y, Chen Z, Pu X, Gao K, Sadollah A (2022) Modelling and scheduling integration of distributed production and distribution problems via black widow optimization. Swarm Evol Comput 68:101015

Gopal A, Sultani MM, Bansal JC (2020) On stability analysis of particle swarm optimization algorithm. Arab J Sci Eng 45:2385–2394

Hassan BA (2021) CSCF: a chaotic sine cosine firefly algorithm for practical application problems. Neural Comput Appl 33:7011–7030

Heidari AA, Mirjalili S, Faris H, Aljarah I, Mafarja M, Chen H (2019) Harris hawks optimization: algorithm and applications. Futur Gener Comput Syst 97:849–872

Jain N, Nangia U, Jain J (2018) A review of particle swarm optimization. J Inst Eng India Ser B 99:407–411

Jayabarathi T, Raghunathan T, Gandomi A (2018) The bat algorithm, variants and some practical engineering applications: a review. Nature-inspired algorithms and applied optimization, pp. 313–330

Joshi AS, Kulkarni O, Kakandikar GM, Nandedkar VM (2017) Cuckoo search optimization-a review. Mater Today Proc 4:7262–7269

Kadry S, Rajinikanth V, Koo J, Kang B-G (2021) Image multi-level-thresholding with Mayfly optimization. Int J Electric Comput Eng 11:5420

Kalita K, Mukhopadhyay T, Dey P, Haldar S (2020) Genetic programming-assisted multi-scale optimization for multi-objective dynamic performance of laminated composites: the advantage of more elementary-level analyses. Neural Comput Appl 32:7969–7993

Karaboga D, Aslan S (2019) Discovery of conserved regions in DNA sequences by Artificial Bee Colony (ABC) algorithm based methods. Nat Comput 18:333–350

Karimzadeh Parizi M, Keynia F (2021) OWMA: an improved self-regulatory woodpecker mating algorithm using opposition-based learning and allocation of local memory for solving optimization problems. J Intell Fuzzy Syst 40:919–946

Karimzadeh Parizi M, Keynia F, Khatibi Bardsiri A (2020) Woodpecker Mating Algorithm (WMA): a nature-inspired algorithm for solving optimization problems. Int J Nonlinear Anal Appl 11:137–157

Karimzadeh Parizi M, Keynia F, Khatibi Bardsiri A (2021) Woodpecker mating algorithm for optimal economic load dispatch in a power system with conventional generators. Int J Ind Electron Control Opt 4:221–234

Kaur G, Sharma A (2022) Electrical devices scheduling in home energy management system using Egyptian vulture optimization algorithm. J Optoelectronics Laser 41:86–95

Kaveh A, Eslamlou AD (2020) Water strider algorithm: a new metaheuristic and applications. Structures 25:520–541

Kaveh A, Rahmani P, Eslamlou AD (2021a) An efficient hybrid approach based on Harris Hawks optimization and imperialist competitive algorithm for structural optimization. Eng Comput 38:1555

Kaveh A, Khodadadi N, Azar BF, Talatahari S (2021b) Optimal design of large-scale frames with an advanced charged system search algorithm using box-shaped sections. Eng Comput 37:2521–2541

Kaveh A, Amirsoleimani P, Eslamlou AD, Rahmani P (2021c) Frequency-constrained optimization of large-scale dome-shaped trusses using chaotic water strider algorithm. Structures 32:1604–1618

Khan S, Tiziano B (2018) Ant colony optimization (aco) based data hiding in image complex region. Int J Electric Comput Eng (IJECE) 8:379–389

Khishe M, Mosavi MR (2020) Chimp optimization algorithm. Expert Syst Appl 149:113338

Kumar M (2021) Energy efficient scheduling in cloud computing using black widow optimization. J Phys Conf Ser 1950:012063

Kumar S, Jain S, Sharma H (2018) Genetic algorithms. Advances in swarm intelligence for optimizing problems in computer science, pp. 27–52

Lin G, Guan J, Li Z, Feng H (2019) A hybrid binary particle swarm optimization with tabu search for the set-union knapsack problem. Expert Syst Appl 135:201–211

Mahmoodpour S, Kamari E, Esfahani MR, Mehr AK (2021) Prediction of cementation factor for low-permeability Iranian carbonate reservoirs using particle swarm optimization-artificial neural network model and genetic programming algorithm. J Petrol Sci Eng 197:108102

EL Majdoubi O, Abdoun F, Abdoun O (2021) A new optimized approach to resolve a combinatorial problem: CoronaVirus Optimization Algorithm and Self-organizing Maps. in International Conference on Digital Technologies and Applications, pp. 947–957

Mareli M, Twala B (2018) An adaptive Cuckoo search algorithm for optimisation. Appl Comput Inf 14:107–115

Martínez-Álvarez F, Asencio-Cortés G, Torres JF, Gutiérrez-Avilés D, Melgar-García L, Pérez-Chacón R et al (2020) Coronavirus optimization algorithm: a bioinspired metaheuristic based on the COVID-19 propagation model. Big Data 8:308–322

Massoudi MS, Sarjamei S, Esfandi Sarafraz M (2020) Smell Bees optimization algorithm for continuous engineering problem. Asian J Civil Eng 21:925–946

Meraihi Y, Gabis AB, Mirjalili S, Ramdane-Cherif A (2021) Grasshopper optimization algorithm: theory, variants, and applications. IEEE Access 9:50001–50024

MiarNaeimi F, Azizyan G, Rashki M (2021) Horse herd optimization algorithm: a nature-inspired algorithm for high-dimensional optimization problems. Knowl-Based Syst 213:106711

Moazzeni AR, Khamehchi E (2020) Rain optimization algorithm (ROA): a new metaheuristic method for drilling optimization solutions. J Petrol Sci Eng 195:107512

Mousavi SM, Abdullah S, Niaki STA, Banihashemi S (2021) An intelligent hybrid classification algorithm integrating fuzzy rule-based extraction and harmony search optimization: medical diagnosis applications. Knowl-Based Syst 220:106943

Nikolić M, Šelmić M, Macura D, Ćalić J (2020) Bee colony optimization metaheuristic for fuzzy membership functions tuning. Expert Syst Appl 158:113601

Nikpour M, Mohebbi A (2022) Optimization of micromixer with different baffle shapes using CFD, DOE, meta-heuristic algorithms and multi-criteria decision making. Chem Eng Proc-Process Intensification 170:108713

Osaba E, Del Ser J, Camacho D, Bilbao MN, Yang X-S (2020) Community detection in networks using bio-inspired optimization: latest developments, new results and perspectives with a selection of recent meta-heuristics. Appl Soft Comput 87:106010

Parizi MK, Keynia F, Bardsiri AK (2021) HSCWMA: a new hybrid SCA-WMA algorithm for solving optimization problems. Int J Inf Technol Decis Mak 20:775–808

Pereira JLJ, Francisco MB, Diniz CA, Oliver GA, Cunha SS Jr, Gomes GF (2021) Lichtenberg algorithm: a novel hybrid physics-based meta-heuristic for global optimization. Expert Syst Appl 170:114522

Pitchipoo P, Muthiah A, Jeyakumar K, Manikandan A (2021) Friction stir welding parameter optimization using novel multi objective dragonfly algorithm. Int J Lightweight Mater Manuf 4:460–467

Qin Y, Kavetski D, Kuczera G (2018) A robust Gauss-Newton algorithm for the optimization of hydrological models: benchmarking against industry-standard algorithms. Water Resour Res 54:9637–9654

Rabanal P, Rodríguez I, Rubio F (2019) Towards applying river formation dynamics in continuous optimization problems. in International Work-Conference on Artificial Neural Networks, pp. 823–832

Rachappanavar V, Padiyal A, Sharma JK, Negi N (2022) Analytical Pipelines for the GBS Analysis. Genotyping by Sequencing for Crop Improvement, pp. 161–187

Rajwar K, Deep K, Das S (2023a) An exhaustive review of the metaheuristic algorithms for search and optimization: taxonomy, applications, and open challenges. Artif Intell Rev 56:13187–13257

Rakhshani H, Rahati A (2017) Snap-drift cuckoo search: a novel cuckoo search optimization algorithm. Appl Soft Comput 52:771–794

Ramos-Figueroa O, Quiroz-Castellanos M, Mezura-Montes E, Schütze O (2020) Metaheuristics to solve grouping problems: a review and a case study. Swarm Evol Comput 53:100643

Rashedi E, Rashedi E, Nezamabadi-Pour H (2018) A comprehensive survey on gravitational search algorithm. Swarm Evol Comput 41:141–158

Sasikala J (2019) Firefly optimization strategy for dynamic economic load dispatch. J Comput Theor Nanosci 16:1612–1616

Sengupta S, Basak S, Peters RA (2018) Particle Swarm Optimization: a survey of historical and recent developments with hybridization perspectives. Mach Learn Knowl Extraction 1:157–191

Shadkam E (2021) Cuckoo optimization algorithm in reverse logistics: a network design for COVID-19 waste management. Waste Manag Res 0734242X211003947

Shadravan S, Naji HR, Bardsiri VK (2019) The Sailfish Optimizer: a novel nature-inspired metaheuristic algorithm for solving constrained engineering optimization problems. Eng Appl Artif Intell 80:20–34

Sharma H, Hazrati G, Bansal JC (2019) Spider monkey optimization algorithm. in Evolutionary and swarm intelligence algorithms, ed: Springer, pp. 43–59

Sharma B, Sharma VK, Kumar S (2020) Sigmoidal spider monkey optimization algorithm. in Soft Computing: Theories and Applications, ed: Springer, pp. 109–117

Shehab M, Khader AT, Al-Betar MA (2017) A survey on applications and variants of the cuckoo search algorithm. Appl Soft Comput 61:1041–1059

Shishavan ST, Gharehchopogh FS (2022) An improved cuckoo search optimization algorithm with genetic algorithm for community detection in complex networks. Multimed Tools Appl 81:1–27

Singh PR, Abd Elaziz M, Xiong S (2018) Modified spider monkey optimization based on Nelder-Mead method for global optimization. Expert Syst Appl 110:264–289

Sivaram M, Batri K, Amin Salih M, Porkodi V (2019) Exploiting the local optima in genetic algorithm using tabu search. Indian J Sci Technol 12:1–13

Slowik A, Kwasnicka H (2020) Evolutionary algorithms and their applications to engineering problems. Neural Comput Appl 32:12363–12379

Song Y, Zhao G, Zhang B, Chen H, Deng W, Deng W (2023) An enhanced distributed differential evolution algorithm for portfolio optimization problems. Eng Appl Artif Intell 121:106004

Syah R, Isola LA, Guerrero JWG, Suksatan W, Sunarsi D, Elveny M et al (2021) Optimal parameters estimation of the PEMFC using a balanced version of Water Strider Algorithm. Energy Rep 7:6876–6886

Talatahari S, Azizi M (2021) Chaos Game Optimization: a novel metaheuristic algorithm. Artif Intell Rev 54:917–1004

Teodorović D, Davidović T, Šelmić M, Nikolić M (2021) Bee colony optimization and its applications. Handbook of AI-based Metaheuristics, pp. 301–322

Tzanetos A, Dounias G (2020) A comprehensive survey on the applications of swarm intelligence and bio-inspired evolutionary strategies. Machine Learning Paradigms, pp. 337–378

Uthayakumar J, Metawa N, Shankar K, Lakshmanaprabu S (2020) Financial crisis prediction model using ant colony optimization. Int J Inf Manage 50:538–556

Wang Y, Wang P, Zhang J, Cui Z, Cai X, Zhang W et al (2019) A novel bat algorithm with multiple strategies coupling for numerical optimization. Mathematics 7:135

Xiuwu Y, Qin L, Yong L, Mufang H, Ke Z, Renrong X (2019) Uneven clustering routing algorithm based on glowworm swarm optimization. Ad Hoc Netw 93:101923

Yan D, Lu Y (2018) Recent advances in particle swarm optimization for large scale problems. J Autonom Intell 1:22–35

Yang L, Shami A (2020) On hyperparameter optimization of machine learning algorithms: theory and practice. Neurocomputing 415:295–316

Yu J, Kim C-H, Rhee S-B (2020) Clustering cuckoo search optimization for economic load dispatch problem. Neural Comput Appl 32:16951–16969

Zanbouri K, Jafari Navimipour N (2020) A cloud service composition method using a trust-based clustering algorithm and honeybee mating optimization algorithm. Int J Commun Syst 33:e4259

Zhao J, Gao Z-M (2020) The multi-start mayfly optimization algorithm. in 2020 7th International Forum on Electrical Engineering and Automation (IFEEA), pp. 879–882

Zheng-Ming G, Su-Ruo L, Juan Z, Yu-Rong H (2020) Heterogeneous mayfly optimization algorithm. in 2020 2nd International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI), pp. 227–230

Zhong M, Wen J, Ma J, Cui H, Zhang Q, Parizi MK (2023) A hierarchical multi-leadership sine cosine algorithm to dissolving global optimization and data classification: the COVID-19 case study. Comput Biol Med 164:107212

Zhou X-H, Zhang M-X, Xu Z-G, Cai C-Y, Huang Y-J, Zheng Y-J (2019) Shallow and deep neural network training by water wave optimization. Swarm Evol Comput 50:100561