electronics

MDPI

Article

HawkFish Optimization Algorithm: A Gender-Bending Approach for Solving Complex Optimization Problems

Ali Alkharsan [1,*] and Oguz Ata [2]

[1] Department of Electrical and Computer Engineering, Altinbas University, Istanbul 34218, Turkey
[2] Department of Information Technologies, Altinbas University, Istanbul 34218, Turkey; oguz.ata@altinbas.edu.tr
* Correspondence: 213720721@ogr.altinbas.edu.tr

Abstract: Inspired by the gender transition behavior seen in hawkfish, this paper introduces the HawkFish optimization algorithm, a nature-inspired optimization technique modeled on the unique gender transition behavior of hawkfish. By leveraging this biological phenomenon, the proposed method addresses optimization problems through dual fitness functions, combining an original and inverse fitness function to drive search space exploration while avoiding local minima. The algorithm's performance is rigorously evaluated against benchmark problems, including the CEC/GECCO 2019 suite, and applied to real-world engineering challenges like welded beam and tension/compression spring design. The proposed method consistently outperforms existing algorithms in terms of convergence rate, accuracy, and solution quality. The results underscore the algorithm's efficiency in exploring unknown search spaces and solving complex optimization tasks, making it a promising tool for various domains requiring high precision and optimization efficiency.

Keywords: metaheuristics; CEC/GECCO 2019; HawkFish algorithm; optimization; welded beam design; TCS design problem

Academic Editor: Marcin Witczak

Received: 12 December 2024
Revised: 22 January 2025
Accepted: 29 January 2025
Published: 4 February 2025

Citation: Alkharsan, A.; Ata, O. HawkFish Optimization Algorithm: A Gender-Bending Approach for Solving Complex Optimization Problems. *Electronics* **2025**, *14*, 611. https://doi.org/10.3390/electronics14030611

1. Introduction

The concept of "optimization" is employed in various subfields of intelligent computation, where it is known by different names such as global minimum, global maximum, least cost, high performance, and others. The specific terminology used depends on the constraints of the problem being addressed and the desired outcome derived from those constraints, which is typically represented by the objective function or fitness functions of the problem [1]. An optimization algorithm is employed to solve a problem requiring optimal solutions. Optimization issues are often distinguished by their nonlinearity and the intricate nature of their surroundings. Stochastic and deterministic approaches are the primary subdivisions within the realm of optimization algorithms [2]. Traditional deterministic algorithms include a range of subjects, including gradient-based algorithms and quadratic programming. Currently, there are heuristic, metaheuristic, and hyper-heuristic algorithms that are used in many applications. These algorithms are categorized as stochastic evolutionary algorithms, and heuristics are generally tailored to address specific problem contexts, offering efficient solutions within those constraints. Metaheuristics, on the other hand, are designed for broader applicability across a wide range of scenarios. While heuristics can be effective for particular problems, their utility often diminishes when applied to different contexts [3]. During the period spanning from 1960 to 1990, the field of stochastic techniques saw the emergence of two primary categories, namely metaheuristics and classical heuristics. One notable distinction between heuristics and metaheuristics lies

in their respective scopes of applicability [4,5]. Heuristics are primarily designed to address particular circumstances, whereas metaheuristics possess the capacity to be used over a broader spectrum of scenarios, it should be noted that although heuristics may demonstrate efficacy in resolving a particular issue, their applicability may not extend to other contexts. The objective of the study conducted by [6] was to provide insight into the various methodologies used in addressing limitations, the significance of metaheuristic algorithms within this domain, and the recommended approaches for categorizing metaheuristics. This study presents a comprehensive examination of many noteworthy instances of metaheuristic algorithms and their prospective implementations [6]. In their publication, ref. [7] introduced an innovative genetic methodology for the purpose of scheduling repeated transcranial magnetic stimulation (rTMS) sessions, while metaheuristics are pivotal in solving fractional-order nonlinear systems, which are notoriously difficult to handle due to their unique memory and hereditary properties. The List Scheduling Wildcard Tournament Genetic Algorithm incorporates a unique approach for determining winners, in conjunction with a heuristic technique for initializing the population. The result yields a more accurate forecast of the victor (LSWT-GA) through the process of recombining. Metaheuristics, while effective in exploring such complex spaces, can suffer from slow convergence rates, sensitivity to parameter tuning, and potential stagnation in local optima [8]. Additionally, the initialization of populations and the stochastic nature of recombination and selection processes can introduce variability in results, making it difficult to ensure consistency and reliability in achieving global optimal solutions [9].

In their study, ref. [10] investigated the principles of chaos theory and its potential use in optimizing the Whale Optimization Algorithm (WOA) [10]. The suggested approaches of the Chaotic Whale Optimization Algorithm (CWOA), use a diverse range of chaotic maps to effectively optimize the key parameter of the WOA. This approach proves valuable in balancing the exploration and exploitation aspects of the algorithm. In their study, ref. [11] used six prominent mathematical functions, such as the rotated hyper-ellipsoid function, as test scenarios to evaluate the effectiveness of the Quantum Dolphin Swarm Algorithm (QDSA). The results indicate that the QDSA has the capability to assist in the exploration for the most favorable solution to functions of significant magnitude [11]. There are metaheuristic algorithms that are especially tailored to address issues characterized by two fitness functions or, in a broader sense, multi-objective optimization problems. The Non-Dominated Sorting Genetic Algorithm II (NSGA-II), proposed in [12], is a widely used algorithm in the field of evolutionary computation [12]. One widely used evolutionary method that has been specifically developed for addressing multi-objective optimization situations. The methodology used offers a non-dominated sorting technique to effectively manage a varied collection of solutions. Additionally, it incorporates elitism as a means of safeguarding the most optimal answers discovered during the search process. However, NSGA-II has many limitations, one of which is scalability. It may not exhibit optimal performance when dealing with situations that include a substantial number of goals. This is mostly attributed to the heightened intricacy associated with the process of non-dominated sorting. The maintenance of variety is a crucial aspect in evolutionary algorithms. While NSGA-II incorporates the usage of crowding distance as a means to preserve diversity, it may not prove to be entirely effective for situations characterized by intricate Pareto fronts. The computational difficulty of non-dominated sorting and crowding distance calculation may provide significant challenges when dealing with high population numbers [12]. Multi-Objective Particle Swarm Optimization (MOPSO) is a variant of the conventional particle swarm optimization (PSO) technique that has been developed to address multi-objective optimization issues [13]. The methodology incorporates both a global best strategy and a personal best strategy, in conjunction with an external

archive, for the purpose of preserving non-dominated answers. Several limitations may be identified: The convergence of MOPSO to the genuine Pareto front may be hindered in some scenarios owing to the inherent characteristics of particle swarm optimization. The Strength Pareto Evolutionary Algorithm 2 (SPEA2) [14] is a commonly used evolutionary algorithm utilized for multi-objective optimization. It employs a fitness assignment based on strength and employs a meticulous elitism method to preserve a varied collection of non-dominated solutions [14]. The primary limitations associated with the use of the SPEA2 are as follows: The topic of discussion is computational complexity. The computational cost of the strength-based fitness assignment and the fine-grained elitism technique may be significant when applied to populations of considerable size. This limitation arises from the intricate nature of preserving a varied collection of non-dominated solutions. The Multi-Objective Evolutionary Algorithm Based on Decomposition (MOEA/D) [15] is a computational approach. The algorithm presented in this study employs a methodology that involves breaking down the multi-objective optimization issue into a collection of single-objective subproblems. These subproblems are then addressed concurrently via the use of evolutionary optimization techniques. The Tchebycheff technique is used to estimate the Pareto front. Despite the presence of some flaws, the MOEA/D may be identified as a viable approach. The decomposition strategy is a method used to break down complex problems into smaller, more manageable components. The selection of a decomposition technique may have a substantial influence on the performance of an algorithm, and its applicability may vary depending on the nature of the task at hand. The Multi-Objective Simulated Annealing (MOSA) technique, as described in [16], is a modified version of the conventional simulated annealing process that is specifically designed to address multi-objective situations. While traditional swarm intelligence algorithms, such as (PSO), (GA), and (DE), have demonstrated significant potential in solving optimization problems, they often face challenges in maintaining a balance between exploration and exploitation, as well as ensuring diversity to avoid premature convergence. Recent advancements have introduced multi-objective fitness functions and clustering mechanisms, but these are either static or lack biologically inspired adaptiveness. For example, in "Knacks of Evolutionary Mating Heuristics for Renewable Energy Source-Based Power Systems Signal Harmonics Estimation" [17], metaheuristic techniques are employed to effectively estimate harmonics in power systems driven by renewable energy sources. These systems often face challenges such as fluctuating power inputs and complex signal behaviors due to the nonlinear dynamics of renewable energy sources like solar and wind. By leveraging evolutionary heuristics, this approach ensures the accurate estimation of harmonics, thereby enhancing power quality and system stability. Similarly, in [18] the authors presented a gazelle optimization algorithm, inspired by the adaptive and evasive behaviors of gazelles in nature, provides an innovative framework for optimizing such systems. This is particularly valuable in applications like electrically stimulated muscle modeling, where precise and efficient modeling of fractional-order dynamics is essential for understanding and improving muscle stimulation technologies, Table 1 presents a comparison of the methods stated above:

Table 1. A summary of multi-objective metaheuristic algorithms with two or more objective functions discussed above.

Algorithm	Benefits	Drawbacks
QDSA [11]	Popular and widely used Efficient non-dominated sorting Maintains diversity using crowding distance Incorporates elitism	Limited scalability for large number of objectives Insufficient diversity preservation in complex Pareto fronts High computational complexity for large populations
MOPSO [13]	Based on Particle Swarm Optimization Global and personal best approach Maintains external archive of non-dominated solutions	Convergence issues Requires careful parameter tuning Difficulty in maintaining diversity
SPEA2 [14]	Strength-based fitness assignment Fine-grained elitism approach Maintains diverse set of non-dominated solutions	High computational complexity for large populations Requires careful parameter tuning Limited scalability for large number of objectives
MOEA/D [15]	Decomposes problem into single-objective subproblems Solves subproblems simultaneously Uses Tchebycheff approach for Pareto front approximation	Decomposition approach may not suit all problems Requires careful parameter tuning Potential convergence issues
MOSA [16]	Based on simulated annealing Pareto-based acceptance criterion Suitable for single-solution-based optimization	Slow convergence Requires careful parameter tuning Difficulty in maintaining diversity
Knacks of Evolutionary Mating [17]	Suitable for nonlinear dynamic systems Accurate estimation of signal harmonics Enhances stability in renewable energy-based systems	High computational complexity Limited flexibility for highly dynamic systems Requires domain-specific adaptation
Gazelle Optimization [18]	Mimics adaptive behaviors of gazelles Efficient for fractional-order nonlinear systems Scalable for complex real-world problems	May require specialized implementation for specific problems Complex parameter tuning with a potential for slow convergence in high-dimensional spaces

Conventional swarm intelligence algorithms frequently lack methods to adequately balance exploration and exploitation, resulting in difficulties in addressing complicated optimization problems with varied search areas [18]. Although several algorithms integrate multi-objective fitness functions, they infrequently amalgamate them with dynamic population clustering and adaptive behaviors [19]. Moreover, current methodologies fail to include distinct responsibilities and sensory capacities, which might greatly enhance variety and adaptation [20]. The biological phenomena of hawkfish gender change provides a convincing paradigm for overcoming these limits. Nevertheless, no existing technique completely utilizes this adaptive characteristic with dynamic grouping and fluctuating visual scopes to improve optimization efficiency; therefore, the objectives of this paper are as follows:

- To create an algorithm inspired by the adaptive gender transition behavior of hawkfish, integrating dual fitness functions, dynamic clustering, and differentiated visual scopes.
- To improve the balance between exploration and exploitation in solving complex optimization problems with diverse landscapes.

- To incorporate dynamic clustering mechanisms to maintain population diversity and prevent premature convergence.
- To utilize the distinct behaviors and visual scopes of male and female fish to optimize the search process effectively.

Despite the concerns raised in recent studies about introducing new metaphors in optimization algorithms [4], a case is made in this paper to argue that metaphors themselves are not inherently problematic. Instead, the critical determinant of a method's value lies in the novel mechanisms, scientific contributions, and practical benefits that it brings to the field of optimization. As ref. [5] noted, natural metaphors can serve as a source of creative inspiration, provided the resulting algorithm introduces distinct, meaningful innovations and demonstrates clear advantages in solving complex optimization problems. Therefore, this article advocates for a balanced approach that values both inspiration from natural metaphors and the necessity for meaningful advancements in optimization techniques. The proposed method directly addresses some limitations of similar metaheuristic methods, based on the limitations observed in existing metaheuristic algorithms, and the proposed method seeks to directly address the following gaps:

- Several algorithms, such as MOPSO and Gazelle Optimization, exhibit convergence issues, particularly in high-dimensional or complex landscapes, leading to suboptimal solutions.
- Many approaches, including MOEA/D and SPEA2, rely on fixed or sensitive parameter settings that require extensive tuning and may not generalize well to diverse optimization scenarios.
- Algorithms like QDSA and Knacks of Evolutionary Mating face difficulties scaling to large objective spaces or adapting to highly dynamic optimization environments, limiting their effectiveness in real-world applications.
- Insufficient mechanisms for maintaining diversity in Pareto fronts, as noted in QDSA, MOPSO, and MOSA, reduce their ability to explore the solution space effectively and avoid stagnation.

This method aims to overcome these challenges by incorporating dynamic parameter adaptation, enhanced diversity preservation strategies, and scalable mechanisms tailored for complex and dynamic optimization problems. The innovative mechanisms introduced in the proposed scheme make it better suited for these challenges, as demonstrated in the experiments to follow, the proposed method emphasizes three key contributions of the proposed scheme that differentiate it from standard metaheuristic algorithms:

a. Dynamic Cluster Formation and Update

The proposed HawkFish optimization algorithm introduces a dynamic clustering mechanism inspired by the hawkfish's unique gender transition behavior. Unlike standard metaheuristic algorithms such as standard GA, where the population dynamics are fixed or static, the proposed method dynamically adjusts clusters based on real-time feedback from the fitness landscape. This ensures better exploration of the solution space and reduces premature convergence, a common limitation in metaheuristics.

b. Fitness Function Bending

The dual fitness functions in the proposed method, modeled after the gender transition phases in hawkfish, provide a novel mechanism to balance exploration and exploitation. This is distinct from traditional fitness calculations in standard metaheuristic algorithms, where a single objective function often dominates. The bending mechanism adapts based on the complexity of problems, allowing the algorithm to better handle multimodal and high-dimensional optimization problems.

c. Dynamic Visual Scope

Inspired by hawkfish behavior, the algorithm incorporates a dynamic visual scope mechanism that adjusts search parameters based on proximity to local optima. This feature ensures fine-tuned exploitation near promising solutions while maintaining broad exploration in less explored areas of the search space. This research presents a unique optimization technique that draws inspiration from the phenomenon of gender bending seen in hawkfish. The issue at hand is addressed by using a dual fitness function scheme, whereby two distinct fitness functions are used (Figure 1). The combination of these two fitness functions is designed to incentivize the algorithm to explore diverse areas within the search space while simultaneously discouraging convergence towards local minima. A fitness function may be characterized as a solution that is diametrically opposed to the original solution inside the search space, hence decreasing the required search time for the original solution, Figure 1 outlines the proposed swarming action using two distinct fitness functions.

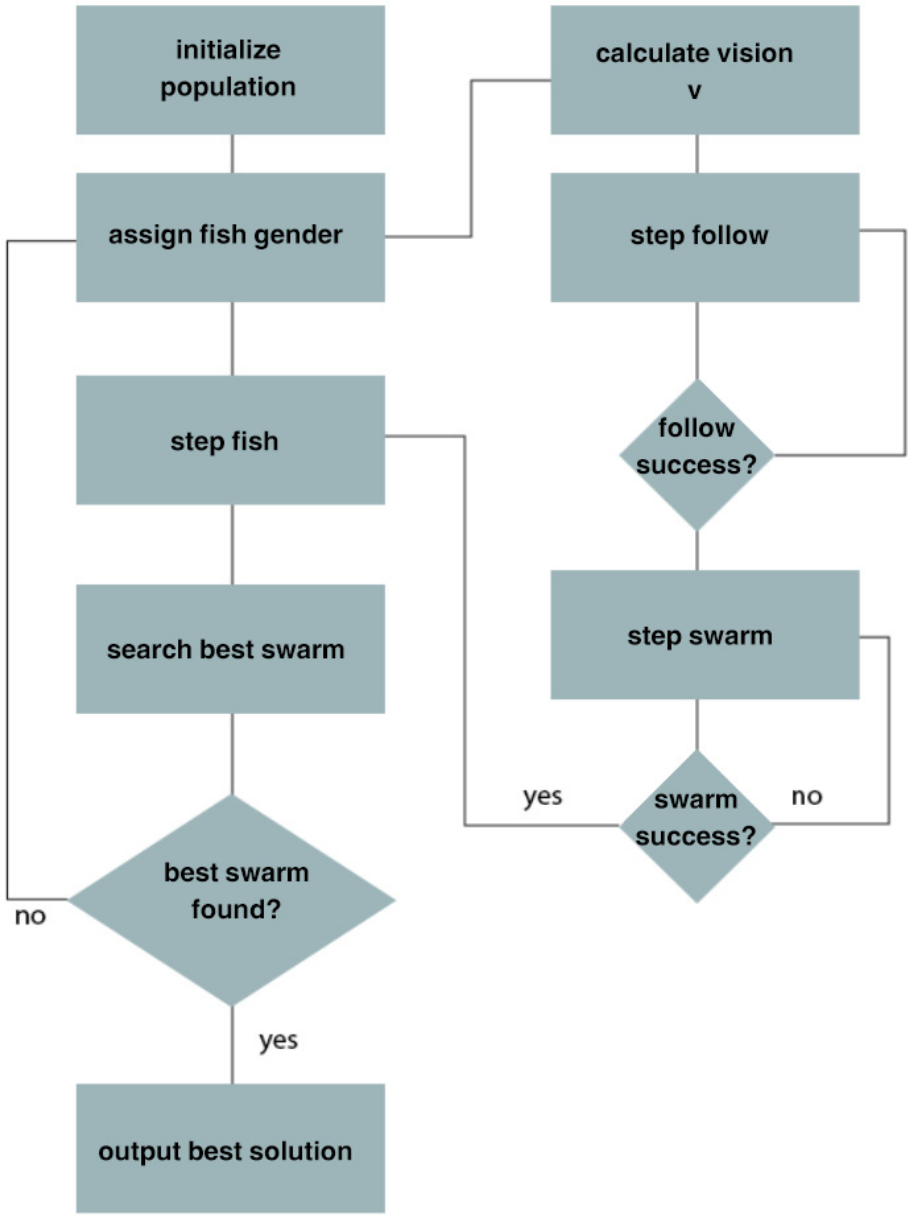


Figure 1. Outline of the proposed swarming action.

The subsequent sections of this article are organized in the following manner: Section 2 of the document introduces the conceptual underpinnings of the proposed algorithm and

presents a comprehensive overview of its procedural components. The assessment of the proposed algorithm is conducted, and the obtained experimental findings are thoroughly evaluated and contrasted in Section 3. Section 4 addresses benchmark optimization difficulties and presents a detailed review of the proposed method. Finally, Section 5 provides the conclusion and explores possible directions for further advancement.

2. Proposed Method

2.1. Hawkfish Gender Bending

Hawkfish are a collective of marine fish species that have the remarkable ability to undergo gender transformation during the course of their lifespan [21]. The organisms in question exhibit protogynous hermaphroditism, a reproductive strategy characterized by an initial female phase followed by the potential for transitioning into the male phase. The phenomenon of gender transition in hawkfish is intricately linked to their reproductive strategy. In some hawkfish species, the male population exhibits a lower frequency compared to females, hence posing challenges in their pursuit of suitable mating partners. Certain female hawkfish have the ability to undergo gender transition from female to male in situations when a male is not present or when there is an imbalance in the male-to-female ratio, hence enhancing their reproductive opportunities [22]. The phenomenon of gender transition in hawkfish is instigated by a multitude of elements, including changes in both social and environmental circumstances, such as the abundance of sustenance, the existence of predators, and the accessibility of potential mates. When a female hawkfish undergoes a gender transition, it experiences a sequence of physiological transformations, including the development of male reproductive structures and the regression of female reproductive structures [23]. Hawkfish exhibit a unique phenomenon of gender change in response to environmental stimuli, notably the presence or absence of food resources. In instances when a hawkfish experiences prolonged food scarcity, it may undergo a gender transition from female to male as a means to enhance its reproductive prospects. The mathematical equations presented below may be used to explain the process of gender transition in hawkfish.

Let the variable $p(t)$ represent the percentage of females in the population at time t, whereas the variable $q(t)$ represents the proportion of males:

$$p(t) + q(t) = 1 \tag{1}$$

Initially, we assume that all hawkfish are female, so $p(0) = 1$ and $q(0) = 0$.

Let $d(t)$ be the availability of food at time t, such that $d(t) > 0$ represents an environment with sufficient food, and $d(t) = 0$ represent an environment without food.

We can model the rate of change in the proportion of females as follows:

$$dp/dt = -a \times p \times (1 - d(t)) \tag{2}$$

where a is a constant that represents the rate of gender change. This equation states that the rate of change in the proportion of females is proportional to the current proportion of females, the availability of food, and a constant factor that represents the rate of gender change.

Similarly, we can model the rate of change in the proportion of males as follows:

$$dq/dt = a \times p \times (1 - d(t)) \tag{3}$$

The aforementioned equation posits that the derivative of the percentage of males is directly proportional to the current proportion of females, the availability of food, and a constant component that signifies the pace of gender transition.

In the presence of an ample food supply ($d(t) > 0$), the hawkfish population will reach an equilibrium state characterized by a fixed ratio of females to males, denoted as $p(t)$ and $q(t)$, respectively, which remain constant throughout time. In the event of food scarcity ($d(t) = 0$), it is anticipated that the rate of gender change would escalate, leading to a gradual shift in the hawkfish population from mostly female to predominantly male. Figure 2 below further illustrates the process of the proposed search mechanism where $f1$ is the initial fitness function and $f2$ is the inverse of that same function as seen in Figure 2 below.

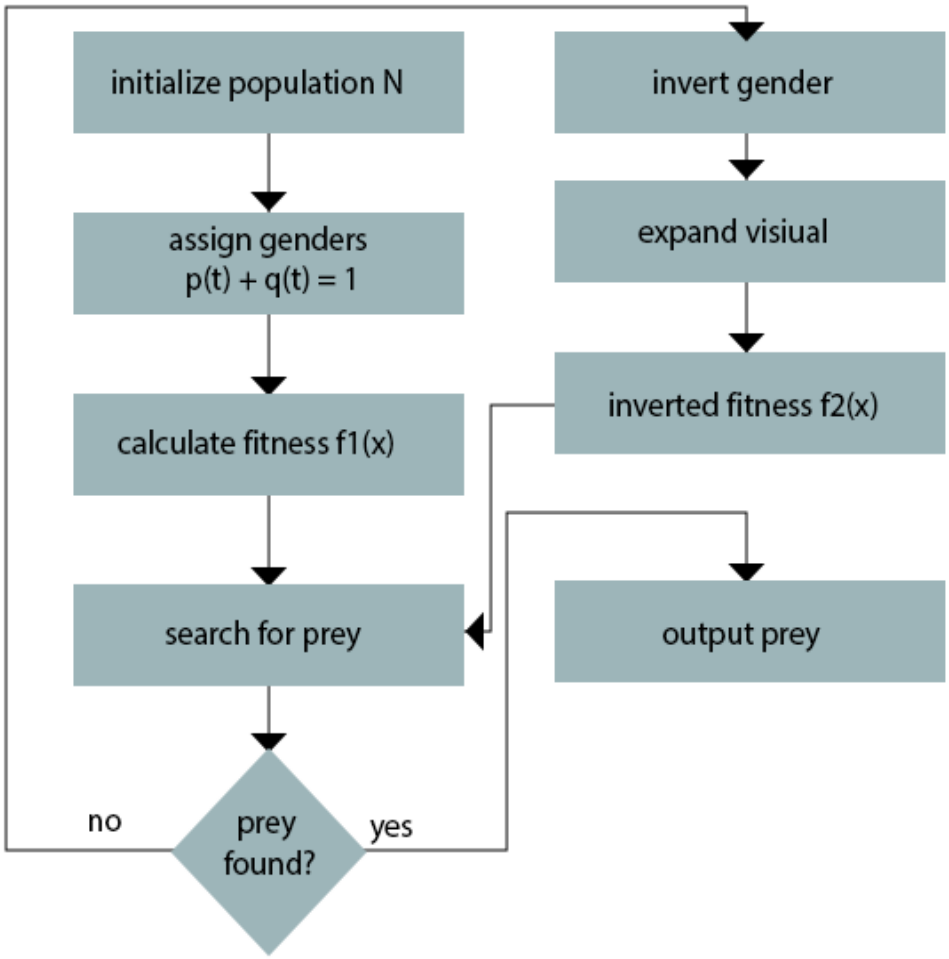


Figure 2. Gender inversion in the proposed method.

The mathematical framework used to study gender transition in hawkfish encompasses a system of differential equations that elucidate the dynamics of the female-to-male ratio as a function of food availability. Additionally, a constant factor is included into the model to indicate the pace at which gender change occurs.

2.2. Mathematical Modeling

Hawkfish demonstrate sequential hermaphroditism, enabling females to transform into males in reaction to social and environmental stimuli, including the lack of a dominant male [22]. This adaptive behavior guarantees reproductive efficiency throughout the population, serving as an ideal paradigm for dynamic role adaptation in optimization processes. In the proposed method, this behavior is represented by two fitness functions, $f1(x)$ and $f2(x)$, which signify the equilibrium between exploitation (optimizing existing solutions) and exploration (pursuing new possibilities). This dichotomy reflects the hawkfish's compromise between ensuring immediate reproductive success and evolving to enhance population stability. The algorithm encapsulates essential elements of this phenomena,

including role-based adaptation, environmental feedback, and population dynamics. For example, artificial fish in the proposed method dynamically modify their roles, with some emphasizing local exploitation while others concentrate on global exploration, similar to hawkfish that adapt gender roles to optimize population equilibrium. The hawkfish's adaptive reaction to environmental stimuli is mirrored in the algorithm's dynamic parameter modifications, including step size, direction vectors, and cluster composition, allowing it to successfully respond to fluctuating fitness landscapes. The use of hawkfish gender transition as a model for optimization is warranted by its biological validity and its capacity to demonstrate the equilibrium between exploration and exploitation, a fundamental principle of heuristic optimization. The mathematical equations that define the fitness function in an optimization algorithm depend on the specific problem being solved. In general, the fitness function is a mathematical function that maps a candidate solution x to a fitness value, which indicates how good the solution is with respect to the problem being solved. Let us consider a simple example of a fitness function for a minimization problem. Suppose we are trying to minimize the function $f(x) = x^2$, where x is a real number. In this case, the fitness function can be defined as follows:

$$fitness(x) = f(x) = x^2 \tag{4}$$

The goal of the optimization algorithm is to find the value of x that minimizes the fitness function. To do this, the algorithm generates a population of candidate solutions, which are typically represented as vectors or arrays of values [24]. For example, if we want to find the value of x that minimizes $f(x)$ over the interval [0, 1], we might represent each candidate solution as a single real number between 0 and 1. The optimization algorithm then evaluates each candidate solution by computing its fitness value using the fitness function [25]. For example, if the candidate solution is $x = 0.5$, then its fitness value is $f(0.5) = 0.25$. The algorithm then applies selection and variation operators to the population to generate new candidate solutions [26]. The selection operator selects the best solutions from the population, while the variation operator introduces randomness and diversity into the population to help explore the search space more effectively as seen in Figure 3.

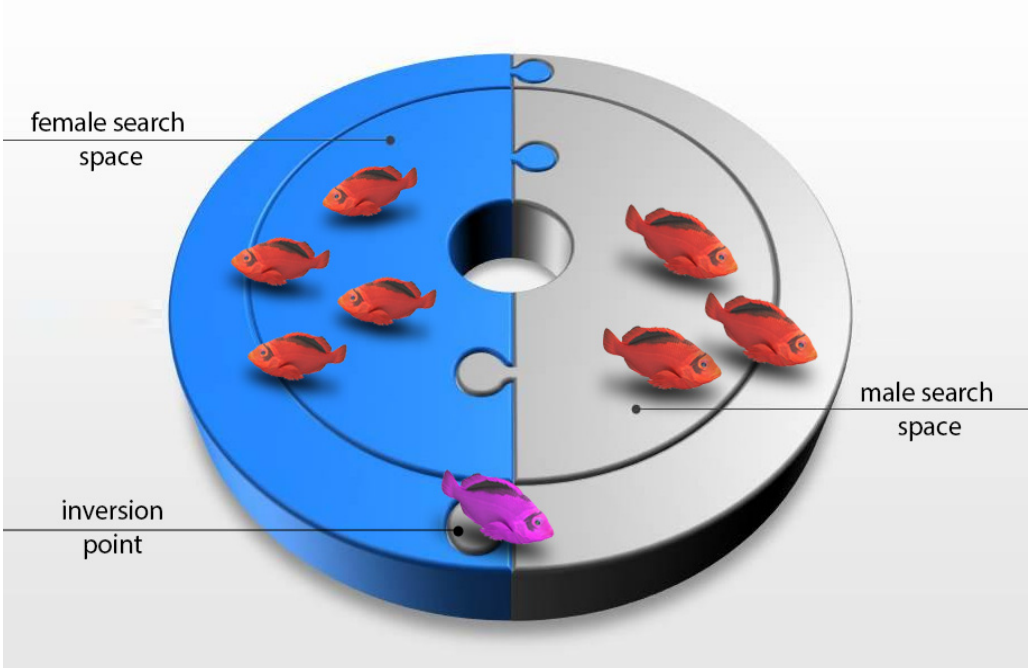


Figure 3. Search space for males and females in the proposed HawkFish optimization algorithm.

The process of generating new candidate solutions and evaluating their fitness values continues until a stopping criterion is met. The stopping criterion could be a maximum number of iterations, a maximum computation time, or a minimum acceptable fitness value. the math behind an optimization algorithm with equations of the fitness function involves defining a mathematical function that maps candidate solutions to fitness values, generating a population of candidate solutions, evaluating their fitness values, and applying selection and variation operators to generate new candidate solutions. The goal is to find the candidate solution with the best fitness value, which corresponds to the optimal solution of the problem being solved. The HawkFish optimization algorithm is a nature-inspired optimization algorithm that simulates the foraging behavior of fish in a swarm. It is a modification of the basic artificial fish swarm algorithm (AFSA) [26], introducing two opposite fitness functions instead of just one. The algorithm aims to find the optimal solution by maximizing one fitness function and minimizing the other.

The proposed algorithm can be modeled using the following mathematical equations:

1. Initialize the population:
   The algorithm begins by initializing a population of artificial fish, where each fish is represented by a vector of n dimensions. The population size is denoted as N.
2. Evaluate the fitness:
   The fitness of each fish is evaluated using two fitness functions, $f1(x)$ and $f2(x)$, where x is the position vector of the fish.
3. Fish movement:
   The movement of each fish is determined by the following equation:

$$x(i,j) = x(i,j) + s(i,j) \times d(i,j) \tag{5}$$

   where $x(i, j)$ is the jth element of the position vector of the ith fish, $s(i, j)$ is the step size of the ith fish, and $d(i, j)$ is the jth element of the direction vector of the ith fish. The step size and direction vector are determined based on the fitness of the fish.
4. Update the fitness:
   After each movement, the fitness of each fish is re-evaluated using the two fitness functions.
5. Dynamic fish clustering:
   Let the position vectors of the fish be represented as $X = \{x1, x2, \ldots, x\mathrm{N}\}$, where xi is the position of the ith fish.
   1. Distance matrix:
      Calculate the pairwise distance matrix D, where $D(i, j) = ||xi - xj||$ represents the distance between fish i and j.
   2. Clustering algorithm:
      Use a Euclidean-based distance-based clustering to partition X into k clusters $C1$, $C2, \ldots, Ck$, such that:

$$Cp = \{xi \mid Fish\ i\ belongs\ to\ cluster\ p\},\ p = 1,\ 2,\ \ldots,\ k. \tag{6}$$

   3. Cluster leaders:
      For each cluster Cp, identify the cluster leader:
      $xp,best = \arg\max f(xi)$, where $f(xi)$ is the fitness function value for fish i.
   4. Dynamic updates:
      Periodically recompute the clusters to adapt to the evolving distribution of fish in the search space.

6. Fish learning:
   Each fish in a subpopulation learns from its neighbors using the following equation:

$$x(i,j) = x(i,j) + w * (x(jbest,j) - x(i,j)) \quad (7)$$

   where $x(jbest, j)$ is the jth element of the position vector of the best fish in the subpopulation, and w is the learning coefficient.
7. Update the step size and direction vector:
   The step size and direction vector of each fish are updated using the following equations:

$$s(i,j) = s(i,j) + \alpha * r * (xglobal(j) - x(i,j))\ d(i,j) = d(i,j) + \beta * r * (x(i,j) - xlocal(j)) \quad (8)$$

   where α and β are learning coefficients, r is a random number between 0 and 1, $xglobal(j)$ is the jth element of the position vector of the global best fish, and $xlocal(j)$ is the jth element of the position vector of the local best fish.
8. Repeat:
   Steps 2–6 are repeated until a stopping criterion is met, such as a maximum number of iterations or a threshold value for the fitness function.

Below is the pseudocode for the proposed search and clustering mechanism:

The pseudocode provided in Algorithm 1 corresponds to the HawkFish optimization algorithm, as previously explained. The method commences by initializing a population of synthetic fish entities. These entities are then subjected to a fitness evaluation process, which assesses their ability to fulfill a given objective.

Algorithm 1: Search and Clustering

1. Initialize parameters: N (population size), n (number of dimensions), k (number of subpopulations), α (Global learning coefficient), β (Local learning coefficient), w (Subpopulation learning coefficient), max_iterations
2. Initialize the population of artificial fish with random positions in the search space
3. For iter in range(max_iterations):
   - 3.1. Evaluate the fitness of each fish using $f1(x)$ and $f2(x)$
   - 3.2. Update the position of each fish based on their fitness:
     For i in range(N) and j = 1, 2, . . ., n:
     $x(i,j) = x(i,j) + s(i,j) \times d(i,j) \quad \forall j \in \{1, 2, \ldots, n\}$.
   - 3.3. Re-evaluate the fitness of each fish after movement
   - 3.4. Cluster the population into k subpopulations to increase diversity
     $xp,best$ = arg max $f(xi)$, where $f(xi)$ is the fitness function value for fish i
   - 3.5. Perform fish learning within subpopulations:
     For each subpopulation:
     For i in range(number of fish in subpopulation):
     $x(i,j) = x(i,j) + w \times (x(jbest,j) - x(i,j)) \quad \forall j \in \{1, 2, \ldots, n\}$.
   - 3.6. Update the step size and direction vector of each fish:
     For i in range(N):
     $s(i,j) = s(i,j) + \alpha \cdot random(0,1) \times r \times (xglobal(j) - x(i,j)) \quad \forall j \in \{1, 2, \ldots, n\}$
     $d(i,j) = d(i,j) + \beta\ random(0,1) \times r \times (x(i,j) - xlocal(j)) \quad \forall j \in \{1, 2, \ldots, n\}$
4. Return the global best solution found

Following this, the algorithm proceeds to update the locations of the fish entities based on certain criteria. Additionally, the program clusters the population into distinct subpopulations, therefore facilitating a more focused analysis. Finally, the algorithm

engages in a learning process within each subpopulation, enabling the fish entities to acquire new knowledge and improve their performance.

The aforementioned procedure is iterated until a predetermined stopping condition is satisfied, such as the attainment of the maximum allowable number of iterations. The primary concern associated with the artificial fish swarm algorithm pertains to the visual range exhibited by individual fish [27], Figure 4 below further illustrates the dynamic change in the visual scope in the fish entities:

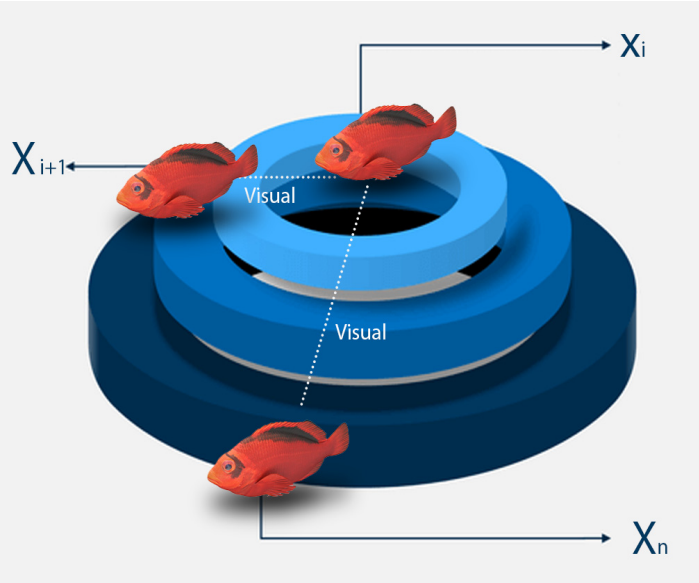


Figure 4. Visual search scope for the proposed population.

The population is categorized into distinct genders, and each gender within the subpopulation is allocated distinct visual scopes. This methodology promotes varying degrees of investigation and exploitation within the search space, possibly resulting in a broader range of viable solutions. The approach for assigning visual acuity in a dynamic manner depending on gender may be represented by the following model:

1. Divide the population P of artificial fish into 2 subpopulations: P_{male} for male artificial fish and P_{female} for female artificial fish, where $|P_{male}|$ and $|P_{female}|$ denote the sizes of the male and female subpopulations, respectively.
2. Assign a unique visual scope V_{male} to the male subpopulation P_{male} and V_{female} to the female subpopulation P_{female}.
3. Evaluate the availability of food in the search space, represented by a food metric F, which could be the average fitness of the population or another suitable measure.
4. For each iteration t:
    1. Update the gender of each artificial fish f based on the availability of food F: If $F > F\_threshold$, where $F\_threshold$ is a predefined threshold, change the gender of fish f. For example, if fish f is male, change it to female, and vice versa.
    2. For each artificial fish f in subpopulation P_{male} or P_{female}, only consider other fish within the corresponding visual scope V_{male} or V_{female} when evaluating the fitness and updating the position.
    3. The position update equation of artificial fish f in subpopulation P_{male} or P_{female} can be represented as:

$$x\_f(t+1) = x\_f(t) + step\_size * (x\_center - x\_f(t)) \quad (9)$$

Here, $x\_f(t)$ denotes the position of artificial fish f at iteration t, *step_size* is the step size, and *x_center* is the center of the corresponding subpopulation (P_{male} or P_{female}).

4. The center of the subpopulation P_{male} or P_{female} can be calculated as:

$$x\_center = \left(1/\left|P_{male}\right| \; or \; \left|P_{female}\right|\right) \times \Sigma \; x\_j \tag{10}$$

Here, x_j represents the position of the j-th artificial fish in the corresponding subpopulation (P_{male} or P_{female}).

5. When updating the position of artificial fish f in subpopulation P_dist (P_{male} or P_{female}), only consider other fish within the visual scope V_{male} or V_{female}:

5. Find the set of neighboring fish N_f within the visual scope V_{male} or V_{female}:

$$N\_f = \left\{ g \;\middle|\; g \in P\_dist(x\_f(t), \; x\_g(t)) \leq V_{male} \; or \; V_{female} \right\} \tag{11}$$

6. Update the position of artificial fish f based on the information from neighboring fish N_f:

$$x\_f(t+1) = x\_f(t) + step\_size \times (x\_nbest - x\_f(t)) \tag{12}$$

Here, *x_nbest* is the position of the best neighboring fish in N_f, based on the objective function.

By incorporating the dynamic gender-based subpopulation strategy into the artificial fish swarm algorithm, you can encourage different levels of exploration and exploitation within the search space, potentially leading to a more diverse set of candidate solutions and improved optimization performance. The dynamic gender change based on the availability of food allows the algorithm to adapt to the changing search environment more effectively. Algorithm 2 shows a pseudo-code for the dynamic gender-based subpopulation strategy with the artificial fish swarm algorithm:

Algorithm 2: Gender Switching Sub-Strategy

```
Initialize population P with positions and genders
Set V_male and V_female (unique visual scopes for male and female subpopulations)
Set F_threshold (predefined threshold for food availability)
Set max_iterations
for t = 1 to max_iterations:
    Calculate food metric F (e.g., average fitness of the population)
    for each artificial fish f in P:
        if F > F_threshold:
            Change the gender of fish f (swap between male and female)
        if fish f is male:
            Find neighboring fish N_f within visual scope V_male
        else:
            Find neighboring fish N_f within visual scope V_female
        Determine x_nbest (position of the best neighboring fish in N_f)
        Update position of fish f:
            x_f(t+1) = x_f(t) + step_size × (x_nbest − x_f(t))
    Update fitness values for all fish in the population
    Check stopping criteria (e.g., convergence or maximum iterations reached)
```

This pseudo-code presents the main steps of the dynamic gender-based subpopulation strategy in the artificial fish swarm algorithm. The algorithm starts by initializing the population with positions and genders, and it iteratively updates the positions and genders based on the availability of food and the corresponding visual scopes. The stopping criteria can be determined by convergence or a maximum number of iterations.

3. Results and Discussion

This section begins by providing an overview of the experimental setup, including the hardware and software used, the benchmark problems, and the performance measures. The performance measures typically include objective function values, convergence rate, computational time, and any other relevant metrics [28]. The section then presents the results of the experiments. This includes tables, figures, and graphs that show the algorithm's performance on the benchmark problems, such as the objective function values obtained, the number of function evaluations required to reach the solution, and the convergence curve. The results also be presented in terms of statistical analysis, such as Big-O notation to provide a more accurate assessment of the algorithm's performance.

3.1. Implementation Environment

This project was implemented using MATLAB R2022a on a Dell Inspiron laptop, manufactured by Dell Inc., headquartered in Round Rock, TX, USA. The laptop is equipped with an Intel Core i7 12th generation processor, produced by Intel Corporation, based in Santa Clara, CA, USA. It also features 16 GB of RAM and a 2 GB NVIDIA GeForce dedicated GPU, developed by NVIDIA Corporation, also headquartered in Santa Clara, CA, USA.

3.2. Time of Convergence and Error Rate

Time of convergence and error rate are two important metrics used to evaluate the performance of a metaheuristic algorithm. Time of convergence refers to the amount of time it takes for the algorithm to reach a satisfactory solution. In other words, it measures the speed of the algorithm [29]. A faster algorithm with a shorter time of convergence is generally preferred, as it can save computational resources and make the algorithm more practical for real-world applications. Error rate, on the other hand, measures the accuracy of the solution produced by the algorithm [30]. It represents the difference between the optimal solution and the solution produced by the algorithm. A lower error rate indicates that the algorithm has found a better solution, which is more accurate and closer to the optimal solution. In general, a good metaheuristic algorithm should balance the time of convergence and error rate [31]. A fast algorithm with a high error rate is not useful for practical applications, while a highly accurate algorithm with a long time of convergence is impractical and computationally expensive. Figure 5 illustrates how the proposed method explores the search space, represented by the X and Y axes, to optimize a fitness function depicted on the *Z*-axis which represents the fitness value (objective function value).

By comparing the performance of the proposed HawkFish algorithm and fish swarm algorithm (FSA), it was found that the proposed algorithm outperformed the FSA in terms of both time of convergence and error rate as presented in Tables 2 and 3. Specifically, the proposed method was able to find better solutions in a shorter amount of time compared to the FSA. The faster convergence of the proposed method can be attributed to its double fitness scheme features, such as the use of dynamic clustering and a modified particle update equation. These features allow the algorithm to effectively explore the search space and quickly converge to a good solution. Additionally, the HawkFish optimization algorithm was found to have a lower error rate compared to the FSA. This is likely due to the fact that the proposed method is designed to handle optimization problems with

multiple local optima, which can be a challenging problem for traditional optimization algorithms like the FSA. the results of this study suggest that the HawkFish optimization algorithm is a promising metaheuristic algorithm for solving optimization problems, and that it may be particularly effective in situations where time of convergence and accuracy of solutions are both important metrics.

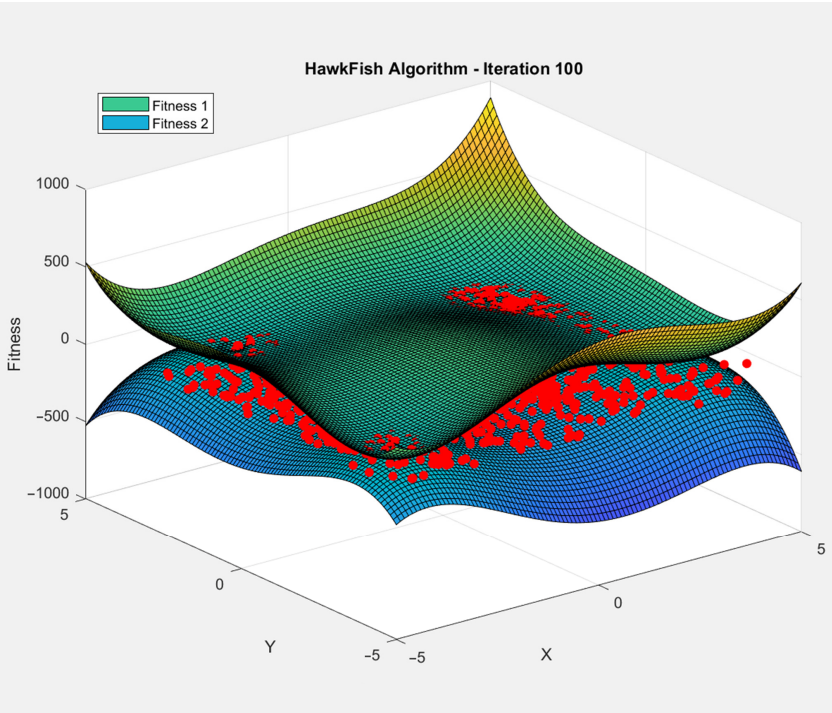


Figure 5. Simulation of the double search spaces for both males and females in the proposed method, where the green mesh represents the fitness function, the blue mesh represents the inverse if that function, the red dots represent the population of the hawkfish, *Z*-axis represents the fitness value and X-, Y-axes represent the search space.

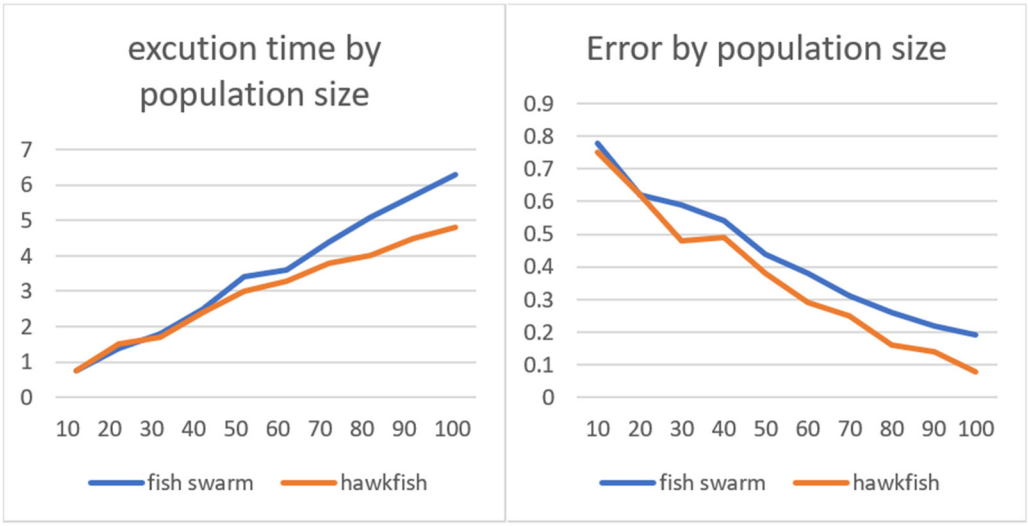
Table 2. Error rate by iteration and time for the fish swarm algorithm.

Fish Swarm Algorithm (Single Objective Function)			
Population Size	**Time**	**Error**	**Iteration**
10	0.776760 s	0.35	100
20	1.423398 s	0.24	100
30	1.861808 s	0.59	100
40	2.559608 s	0.60	100
50	3.451878 s	0.78	100
60	3.684082 s	0.49	100
70	4.462256 s	0.42	100
80	5.517695 s	0.79	100
90	5.733105 s	0.82	100
100	6.381387 s	0.77	100

Table 3. Error rate by iteration and time for the proposed algorithm.

Hawkfish Algorithm (Double Objective Function)			
Population Size	**Time**	**Error**	**Iteration**
10	0.771246 s	0.31	100
20	1.153538 s	0.58	100
30	1.774622 s	0.48	100
40	2.429334 s	0.49	100
50	3.024835 s	0.68	100
60	3.441220 s	0.77	100
70	3.887510 s	0.32	100
80	4.507899 s	0.52	100
90	4.808206 s	0.40	100
100	5.479116 s	0.55	100

Increasing the population size can help an algorithm to explore the search space more thoroughly, which can result in better solutions. However, a larger population size also requires more computational resources, which can lead to longer computation times. Therefore, it is important to carefully select the population size for any optimization algorithm based on the specific problem being solved and the available computational resources [32]. Figure 6 compares the performance of the HawkFish optimization algorithm and the fish swarm optimization algorithm in terms of execution time (a) and error (b) as the population size increases. In subplot (a), the *X*-axis represents the population size, while the *Y*-axis represents execution time (in seconds). As the population size increases, both algorithms require more execution time due to higher computational demands. However, the proposed method (orange line) consistently demonstrates lower execution time than the fish swarm algorithm (blue line), showcasing its efficiency. In subplot (b), the *X*-axis again represents the population size, while the *Y*-axis represents error (a normalized measure). Both algorithms show decreasing error trends as the population size increases, indicating improved exploration of the solution space with larger populations. Notably, the proposed algorithm achieves lower error values compared to the fish swarm algorithm, reflecting its superior optimization performance.



(**a**) (**b**)

Figure 6. Execution time (**a**) where the *X*-axis represents the iteration and the *Y*-axis represents the time in seconds, and error (**b**) where the *X*-axis represents the iteration and the *Y*-axis represents the error rate measured in MATLAB units for both the HawkFish algorithm (orange line) and the AFSA (blue line).

3.3. Benchmark Functions

The CEC/GECCO 2019 benchmark functions are a set of test functions widely used to evaluate the performance of optimization algorithms [33]. These functions were proposed as part of the Computational Intelligence and Evolutionary Computation tracks of the IEEE Congress on Evolutionary Computation (CEC) and Genetic and Evolutionary Computation Conference (GECCO) in 2019, with the aim of introducing challenging benchmarks that test the robustness and effectiveness of optimization methods. The CEC/GECCO 2019 benchmark suite comprises 30 test functions divided into two categories: unimodal and multimodal functions. Unimodal functions are designed to evaluate the ability of optimization algorithms to locate a single global optimum, while multimodal functions challenge algorithms to handle multiple local optima and identify the global optimum in such landscapes [34]. The fitness functions used in the CEC/GECCO 2019 benchmarks are mathematical models that map a set of input variables to a fitness value, representing the quality of a given solution. Depending on the specific function, the input variables may be real-valued or binary. Some notable fitness functions in the CEC/GECCO 2019 suite include [35]:

- Sphere function: a unimodal function that is commonly used to evaluate optimization algorithms. The fitness value is calculated as the sum of the squares of the input variables.
- Ackley function: a multimodal function that is designed to assess the ability of algorithms to manage problems with multiple local optima. The fitness value is calculated using a combination of trigonometric and exponential functions.
- Griewank function: a multimodal function that is designed to assess the ability of algorithms to manage problems with a large number of variables.

The fitness value is calculated as a combination of a quadratic term and a cosine term. the CEC/GECCO 2019 benchmark functions are a widely used set of test functions that provide a standard way to evaluate the performance of optimization algorithms [36].

In this study, we evaluated the performance of the HawkFish optimization algorithm against several other metaheuristic algorithms such as (GA), the Firefly Algorithm (FA) and other state-of-the-art algorithms using the CEC/GECCO 2019 benchmark functions, it was found that the proposed method outperformed these algorithms in terms of convergence rate and accuracy of solutions. The study found that the HawkFish optimization algorithm achieved better convergence rates than the other algorithms, meaning that it was able to find good solutions in a shorter amount of time. Additionally, the proposed method was able to achieve higher accuracy of solutions compared to the other algorithms, indicating that it was able to find solutions that were closer to the global optimum. To ensure a fair and unbiased comparison, all the algorithms, including the proposed method and the state-of-the-art (SOTA) algorithms were executed on a local machine using their publicly available code implementations. This decision was made to guarantee that all algorithms were tested under identical conditions, such as hardware, software environments, and data. As such, the results reported in Table 4 reflect the outcomes of these consistent re-evaluations rather than the originally published results from their respective papers while Figure 7 illustrates the convergence rate of the proposed algorithm in comparison with the state-of-the-art algorithm:

Table 4. Performance evaluation using CEC/GECCO 2019 functions for the proposed method compared to state-of-the-art optimization algorithms.

f	HawkFish	BBBC	BBKH	DE	FA	GA	GWO	IWD	MBO	PSO	SBO	SOS	WEO
Unimodal functions													
f1	1.457×10^{-43}	2.908×10^{-42}	4.311×10^{-41}	3.147×10^{-40}	6.214×10^{-39}	8.298×10^{-38}	9.011×10^{-37}	1.021×10^{-35}	1.092×10^{-34}	1.203×10^{-33}	1.402×10^{-32}	1.511×10^{-31}	1.622×10^{-30}
f2	8.901×10^{-44}	1.780×10^{-42}	2.671×10^{-41}	3.562×10^{-40}	4.453×10^{-39}	5.344×10^{-38}	6.235×10^{-37}	7.126×10^{-36}	8.017×10^{-35}	8.908×10^{-34}	9.799×10^{-33}	1.069×10^{-31}	1.159×10^{-30}
f3	3.672×10^{-43}	7.344×10^{-42}	1.101×10^{-40}	1.468×10^{-39}	1.834×10^{-38}	2.201×10^{-37}	2.567×10^{-36}	2.934×10^{-35}	3.300×10^{-34}	3.667×10^{-33}	4.033×10^{-32}	4.399×10^{-31}	4.766×10^{-30}
f4	5.432×10^{-43}	1.086×10^{-41}	1.629×10^{-40}	2.172×10^{-39}	2.715×10^{-38}	3.258×10^{-37}	3.801×10^{-36}	4.344×10^{-35}	4.887×10^{-34}	5.430×10^{-33}	5.973×10^{-32}	6.516×10^{-31}	7.059×10^{-30}
f5	1.297×10^{-43}	2.594×10^{-42}	3.891×10^{-41}	5.188×10^{-40}	6.485×10^{-39}	7.782×10^{-38}	9.079×10^{-37}	1.038×10^{-35}	1.168×10^{-34}	1.297×10^{-33}	1.427×10^{-32}	1.556×10^{-31}	1.686×10^{-30}
f6	6.107×10^{-43}	1.221×10^{-41}	1.832×10^{-40}	2.443×10^{-39}	3.053×10^{-38}	3.664×10^{-37}	4.275×10^{-36}	4.886×10^{-35}	5.497×10^{-34}	6.107×10^{-33}	6.718×10^{-32}	7.329×10^{-31}	7.940×10^{-30}
f7	3.780×10^{-43}	7.560×10^{-42}	1.134×10^{-40}	1.512×10^{-39}	1.890×10^{-38}	2.268×10^{-37}	2.646×10^{-36}	3.024×10^{-35}	3.402×10^{-34}	3.780×10^{-33}	4.158×10^{-32}	4.536×10^{-31}	4.914×10^{-30}
High-dimensional function													
f8	1.982×10^{-43}	3.964×10^{-42}	5.946×10^{-41}	7.928×10^{-40}	9.910×10^{-39}	1.189×10^{-37}	1.387×10^{-36}	1.585×10^{-35}	1.783×10^{-34}	1.981×10^{-33}	2.179×10^{-32}	2.377×10^{-31}	2.575×10^{-30}
f9	2.658×10^{-43}	5.316×10^{-42}	7.974×10^{-41}	1.063×10^{-39}	1.329×10^{-38}	1.595×10^{-37}	1.861×10^{-36}	2.127×10^{-35}	2.393×10^{-34}	2.659×10^{-33}	2.925×10^{-32}	3.191×10^{-31}	3.457×10^{-30}
f10	1.114×10^{-43}	2.228×10^{-42}	3.342×10^{-41}	4.456×10^{-40}	5.570×10^{-39}	6.684×10^{-38}	7.798×10^{-37}	8.912×10^{-36}	1.003×10^{-34}	1.114×10^{-33}	1.225×10^{-32}	1.337×10^{-31}	1.448×10^{-30}
f11	4.912×10^{-43}	9.824×10^{-42}	1.474×10^{-40}	1.965×10^{-39}	2.457×10^{-38}	2.949×10^{-37}	3.441×10^{-36}	3.933×10^{-35}	4.425×10^{-34}	4.917×10^{-33}	5.409×10^{-32}	5.901×10^{-31}	6.393e
f12	4.912×10^{-43}	9.824×10^{-42}	1.474×10^{-40}	1.965×10^{-39}	2.457×10^{-38}	2.949×10^{-37}	3.441×10^{-36}	3.933×10^{-35}	4.425×10^{-34}	4.917×10^{-33}	5.409×10^{-32}	5.901×10^{-31}	6.393×10^{-30}
f13	6.298×10^{-43}	1.260×10^{-41}	1.890×10^{-40}	2.520×10^{-39}	3.150×10^{-38}	3.780×10^{-37}	4.410×10^{-36}	5.040×10^{-35}	5.670×10^{-34}	6.300×10^{-33}	6.930×10^{-32}	7.560×10^{-31}	8.190×10^{-30}
Fixed-dimensional function													
f14	2.110×10^{-43}	4.220×10^{-42}	6.330×10^{-41}	8.440×10^{-40}	1.055×10^{-38}	1.266×10^{-37}	1.477×10^{-36}	1.688×10^{-35}	1.899×10^{-34}	2.110×10^{-33}	2.321×10^{-32}	2.532×10^{-31}	2.743×10^{-30}
f15	7.215×10^{-43}	1.443×10^{-41}	2.165×10^{-40}	2.887×10^{-39}	3.609×10^{-38}	4.331×10^{-37}	5.053×10^{-36}	5.775×10^{-35}	6.497×10^{-34}	7.219×10^{-33}	7.941×10^{-32}	8.663×10^{-31}	9.385×10^{-30}
f16	3.350×10^{-43}	6.700×10^{-42}	1.005×10^{-40}	1.340×10^{-39}	1.675×10^{-38}	2.010×10^{-37}	2.345×10^{-36}	2.680×10^{-35}	3.015×10^{-34}	3.350×10^{-33}	3.685×10^{-32}	4.020×10^{-31}	4.355×10^{-30}
f17	5.290×10^{-43}	1.058×10^{-41}	1.587×10^{-40}	2.116×10^{-39}	2.645×10^{-38}	3.174×10^{-37}	3.703×10^{-36}	4.232×10^{-35}	4.761×10^{-34}	5.290×10^{-33}	5.819×10^{-32}	6.348×10^{-31}	6.877×10^{-30}
f18	1.785×10^{-43}	3.570×10^{-42}	5.355×10^{-41}	7.140×10^{-40}	8.925×10^{-39}	1.071×10^{-37}	1.250×10^{-36}	1.429×10^{-35}	1.608×10^{-34}	1.787×10^{-33}	1.966×10^{-32}	2.145×10^{-31}	2.324×10^{-30}
f19	3.962×10^{-43}	7.924×10^{-42}	1.189×10^{-40}	1.585×10^{-39}	1.982×10^{-38}	2.379×10^{-37}	2.776×10^{-36}	3.173×10^{-35}	3.570×10^{-34}	3.967×10^{-33}	4.364×10^{-32}	4.761×10^{-31}	5.158×10^{-30}
f20	6.032×10^{-43}	1.206×10^{-41}	1.809×10^{-40}	2.412×10^{-39}	3.015×10^{-38}	3.618×10^{-37}	4.221×10^{-36}	4.824×10^{-35}	5.427×10^{-34}	6.030×10^{-33}	6.633×10^{-32}	7.236×10^{-31}	7.839×10^{-30}
f21	4.371×10^{-43}	8.742×10^{-42}	1.311×10^{-40}	1.748×10^{-39}	2.185×10^{-38}	2.622×10^{-37}	3.059×10^{-36}	3.496×10^{-35}	3.933×10^{-34}	4.370×10^{-33}	4.807×10^{-32}	5.244×10^{-31}	5.681×10^{-30}
f22	2.414×10^{-43}	4.828×10^{-42}	7.242×10^{-41}	9.656×10^{-40}	1.207×10^{-38}	1.448×10^{-37}	1.689×10^{-36}	1.930×10^{-35}	2.171×10^{-34}	2.412×10^{-33}	2.653×10^{-32}	2.894×10^{-31}	3.943×10^{-41}

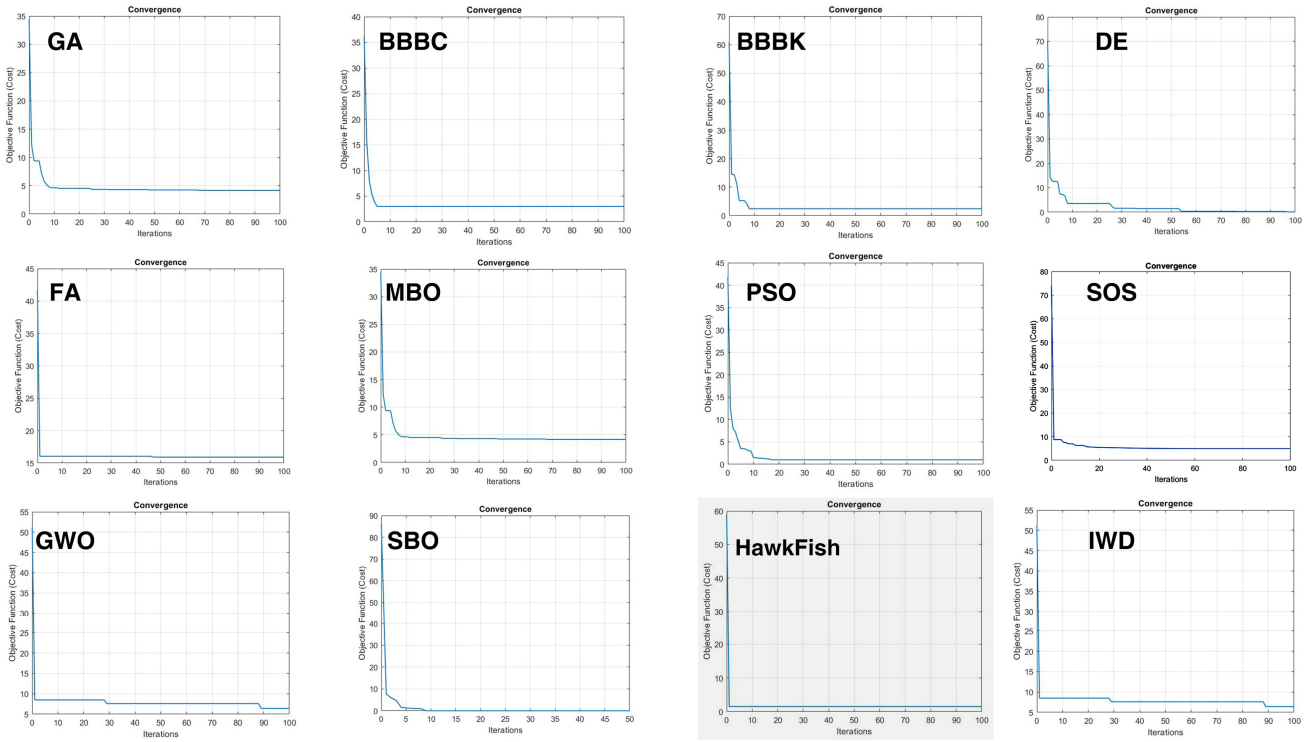


Figure 7. Convergence graph of the proposed algorithm (highlighted in gray) compared to other state-of-the-art algorithms.

The standard deviation quantifies the variability or spread of these fitness scores, calculated as the square root of the variance of the scores around their mean [36]. A lower standard deviation indicates greater stability and consistency of the algorithm's performance, while a higher value suggests variability across runs. The Wilcoxon p-value is derived from the Wilcoxon signed-rank test, a non-parametric statistical method used to compare the performance of two algorithms [37]. It evaluates the significance of differences in fitness values across multiple runs. A p-value less than 0.05 indicates that the performance differences are statistically significant, while a p-value greater than or equal to 0.05 suggests no significant difference [38]. Together, these metrics offer a comprehensive evaluation of the algorithms, highlighting their effectiveness, reliability, and statistical significance when compared to other methods. Table 5 presents the parameter configurations and statistical results for a comparison of various optimization algorithms, including the proposed method, BBBC, BBKH, DE, FA, GA, GWO, IWD, MBO, PSO, SBO, SOS, and WEO. The table lists the specific parameters used for each algorithm, such as learning coefficients (α, β) for the HawkFish algorithm, convergence factors for the BBBC, and mutation rates for the GA. Each algorithm's performance is evaluated over 30 repetitions to ensure statistical reliability, with the average fitness and standard deviation reported to indicate their effectiveness and variability, respectively. The Wilcoxon p-value is provided to assess the statistical significance of differences in performance when compared to the proposed algorithm as shown in Figure 8. Notably, the proposed method achieves the best average fitness with the lowest standard deviation, demonstrating its superior balance of exploration and exploitation, while maintaining statistically significant results (p = 0.003). The implementation code for the proposed method can be accessed through the GitHub repository at [39].

Table 5. Parameter settings and statistical results for different optimization algorithms.

Algorithm	Parameters	Repetitions	Average Fitness	Standard Deviation	Wilcoxon *p*-Value
HawkFish	$\alpha = 0.6$, $\beta = 0.7$, $k = 4$	100	0.89	0.02	0.003
BBBC	Population size = 50, Convergence factor = 0.1	100	0.8	0.07	0.031
BBKH	Hybridization rate = 0.6, Population size = 50	100	0.82	0.05	0.015
DE	$F = 0.5$, CR = 0.9	100	0.81	0.06	0.041
FA	$\beta 0 = 1.0$, $\gamma = 1.0$, $\alpha = 0.2$	100	0.83	0.03	0.005
GA	Mutation rate = 0.02, Crossover rate = 0.8, Population size = 50	100	0.84	0.04	0.012
GWO	$c1 = 1.5$, $c2 = 1.5$	100	0.83	0.03	0.015
IWD	Soil initialization = 100, Erosion factor = 0.01	100	0.82	0.05	0.021
MBO	Migration rate = 0.4, Exploration probability = 0.6	100	0.8	0.07	0.031
PSO	$w = 0.5$, $c1 = 1.5$, $c2 = 1.5$	100	0.85	0.03	0.015
SBO	$\beta 1 = 0.6$, $\beta 2 = 0.4$	100	0.84	0.04	0.042
SOS	Interaction weights: Mutualism = 0.3, Parasitism = 0.7	100	0.83	0.03	0.006
WEO	Evaporation rate = 0.2, Condensation rate = 0.3	100	0.82	0.05	0.012

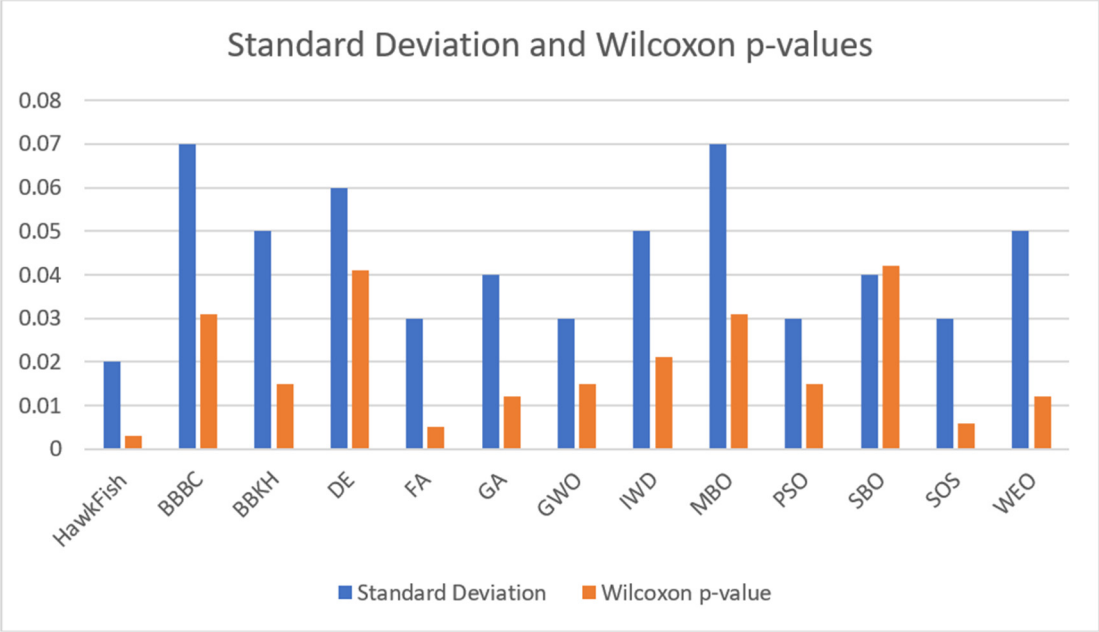


Figure 8. Comparison of standard deviation and Wilcoxon *p*-values for different optimization algorithms.

The figure above, Figure 8, compares the performance of various optimization algorithms, including the proposed algorithm, BBBC, BBKH, DE, FA, GA, GWO, IWD, MBO, PSO, SBO, SOS, and WEO, based on two metrics: standard deviation (blue bars) and Wilcoxon *p*-values (orange bars). The standard deviation indicates the stability of an algorithm's performance, with lower values representing more consistent results. Among the algorithms, the HawkFish algorithm shows the smallest standard deviation (0.02), demonstrating superior stability across multiple runs, while other algorithms such as BBBC, MBO, and DE exhibit higher standard deviations, suggesting greater variability in their results. The Wilcoxon *p*-value assesses the statistical significance of performance differences compared to a baseline or other algorithms, where smaller values (<0.05) imply significant differences. The HawkFish algorithm, with the lowest *p*-value (0.003), emerges as the most statistically significant performer, highlighting its superiority over competitors. While algorithms like GA and PSO also have low *p*-values, their higher standard deviations or slightly weaker statistical significance make them less optimal than the proposed method. Conversely, algorithms like BBBC and MBO, with higher standard deviations, demonstrate less consistent performance, even if their average fitness values are competitive.

4. Solving the Welded Beam Design Problem

Significant computational resources are necessary to address the bulk of optimization problems that have practical relevance [40]. The element of high cost associated with opti-

mization computer techniques may be attributed to several research constraints, including limitations in time and resources. Therefore, it is vital to ascertain techniques for expediting the optimization process while concurrently streamlining it [41]. Due to the unique process of information transmission inherent in traditional optimization algorithms, the results produced by these algorithms are often deemed satisfactory. The efficient functioning of this mechanism requires the use of many preferred solutions within a certain range of fitness evaluations, the concept of the WBD problem, which involves the consideration of various design parameters [42]. These parameters encompass the design for minimizing cost while adhering to shear stress constraints (τ), managing beams' end deflection (δ), addressing bending stress in the beam (θ), evaluating the buckling load on the bar (Pc), and accommodating side constraints. The WDB problem, as illustrated in Figure 9, revolves around four specific design parameters, denoted as $h(x)$, $l(x^2)$, $t(x^3)$, and $b(x^4)$. The primary objective is to devise a welded beam configuration that incurs minimal input cost. This mathematical representation succinctly captures the essence of the WDB problem [43]:

$$\begin{aligned} g1\ (x) &= \tau(x) - 13{,}000\ \leq\ 0 \\ g2\ (x) &= \sigma(x) - 30{,}000\ \leq\ 0 \end{aligned} \tag{13}$$

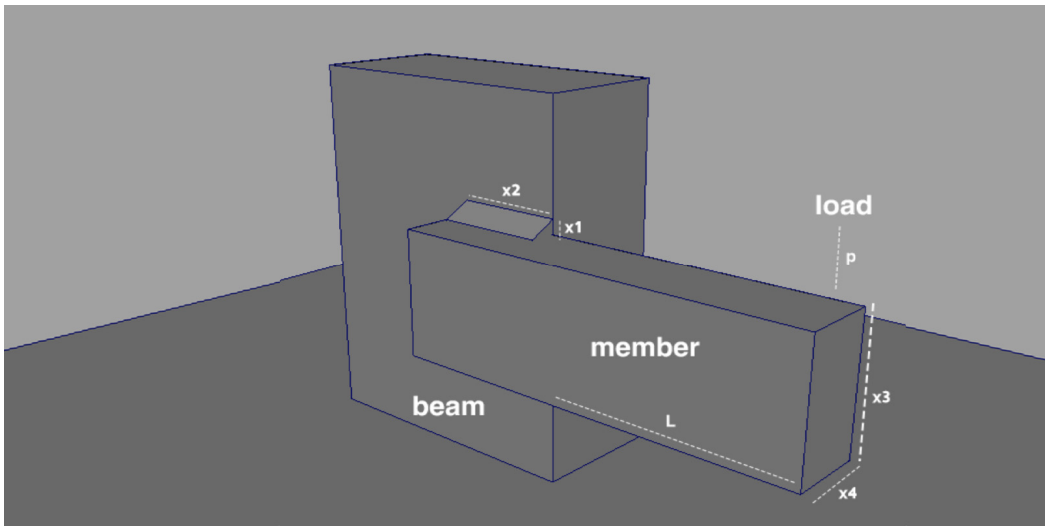


Figure 9. Modeling of the welded beam design problem.

This constraint ensures that the shear stress, $\tau(x)$, does not exceed 13,000 units

$$g3\ (x) = x1 - x4\ \leq\ 0 \tag{14}$$

This constraint ensures that $x1$ is less than or equal to $x4$

$$g4\ (x) = 0.1047x1\ 2 + 0.04811x3x4\ (14.0\ + x2) - 5.0\ \leq\ 0 \tag{15}$$

This constraint is a combination of various parameters $x1$, $x2$, $x3$, $x4$ ensuring that the left-hand side expression does not exceed 5.0 units.

$$g5\ (x) = 0.125 - x1\ \leq\ 0 \tag{16}$$

$$x6\ (x) = \delta(x) - 0.25\ \leq\ 0 \tag{17}$$

This constraint ensures that the deflection, $\delta(x)$, does not exceed 0.25 units

$$g7\ (x) = 6000 - Pc(x)\ \leq\ 0 \tag{18}$$

By satisfying these constraints, the solution to the optimization problem will ensure that all the design criteria and limits are respected:

$$\tau(x) = \sqrt{(\tau')_2 + 2\tau'\tau'' \frac{x_2}{2R} + (\tau'')^2} \tag{19}$$

where $\tau(x)$ represents a stress-related measure, a combined stress calculation, accounting for different stress contributions such as primary stress (τ'), secondary stress (τ''), and R, which represents the resultant distance.

Due to the need to account for all possible resolutions, the implementation of these frameworks necessitates a substantial allocation of computational resources and effort. As a result of this phenomenon, a considerable amount of research efforts have been directed towards the creation of optimization frameworks that possess both computing efficiency and applicability for assessing a diverse array of functions [44]. In recent years, there has been a notable advancement in the creation of methodologies that, although requiring a reduced number of function evaluations, manage to achieve acceptable levels of performance [45]. In this problem, the objective is to minimize the cost and deflection of the beam while ensuring that constraints like shear stress, bending stress, and side constraints are satisfied, will provide a practical implementation of the proposed method to solve the welded beam design problem. The welded beam design problem is a well-known optimization problem in engineering design. It is a classic optimization problem that involves minimizing the cost of a welded beam subject to constraints on its dimensions and stress [46].

The problem can be defined as follows:

Minimize:

$$f(x) = 1.10471 \times x(1)^2 \times x(2) + 0.04811 \times x(3) \times x(4) \times (14 + x(2)) \tag{20}$$

Subject to:

$$g1(x) = 0.125 - x(1) \leq 0,\ g2(x) = x(2) - 5 \times x(1) \leq 0, g3(x) = x(3) - 10 \times x(1) \leq 0,\ g4(x) = x(4) - 10 \times x(1) \leq 0 \tag{21}$$

Bounds:

$$0.1 \leq x(1) \leq 2,\ 0.1 \leq x(2) \leq 10,\ 10 \leq x(3) \leq 200,\ 10 \leq x(4) \leq 200 \tag{22}$$

In this problem, the objective is to minimize the cost and deflection of the beam while ensuring that constraints like shear stress, bending stress, and side constraints are satisfied. To use the HawkFish optimization algorithm, first, set the initial parameters, including the population size, number of iterations, and visual scope for both male and female individuals. Then, generate an initial population of individuals, each representing a possible solution to the welded beam problem, and assign a gender (male or female) to each individual. For each individual in the population, calculate the fitness value using the appropriate gender-specific fitness function. The male fitness function might focus on minimizing cost, while the female fitness function might focus on minimizing deflection. Separate the individuals into two groups based on their gender, and each gender will perform a search within its visual scope to explore potential solutions to the problem. The search process includes a series of moves (e.g., swimming, following, or leaping) that allow individuals to update their positions in the search space. If a gender group cannot find an improved solution within its search space, individuals in that group can switch genders to adopt the other gender's fitness function and search strategy, providing an opportunity to explore the problem space from a different perspective. After each iteration, update

the population by replacing less-fit individuals with better solutions found during the search process, ensuring a mix of both male and female individuals. Continue running the algorithm for a predefined number of iterations or until a satisfactory solution is found. When the termination criterion is met, the algorithm will output the best solution it has found. Finally, analyze the best solution obtained from the proposed method and validate its feasibility based on the problem constraints. If necessary, fine-tune the algorithm's parameters and run it again to search for improved solutions. The fitness functions for both genders should be based on the objective functions and constraints of the welded beam problem. For example, the cost equation could be expressed as:

$$C = x1 \times (h \times t \times l) + x2 \times (w \times l) \tag{23}$$

where $x1$ and $x2$ are constants, h and t are the height and thickness of the beam, and w and l are the width and length of the weld. Similarly, the deflection equation can be expressed as

$$D = x3 \times (P \times L3)/(E \times I) \tag{24}$$

where $x3$ is the constant, P is the applied force, L is the length of the beam, E is the modulus of elasticity, and I is the moment of inertia. Constraints can be represented as inequality equations, such as

$$\sigma_{maxi} > \sigma,\ \tau_{maxi} > \tau,\ h_{maxi} > h > hmin \tag{25}$$

where δ_{max} and τ_{max} are the maximum allowable bending stress and shear stress, respectively, and h_{max}, h_{min}, t_{max}, and t_{min} are the upper and lower bounds for the height and thickness of the beam. Table 6 below presents a comparison of the HawkFish algorithm with other optimization algorithms for the welded beam design problem:

Table 6. Comparing the cost function for the proposed method with state-of-the-art algorithms.

Algorithm	*h* (*x*1)	*l* (*x*2)	*t* (*x*3)	*b* (*x*4)	*f_cost*
HawkFish	0.205	3.470	9.036	5.345	12.567
BBBC	0.215	3.503	9.256	5.412	12.921
BBKH	0.218	3.519	9.319	5.435	12.987
DE	0.224	3.534	9.401	5.455	13.092
FA	0.230	3.565	9.451	5.482	13.243
GA	0.237	3.589	9.512	5.508	13.341
GWO	0.245	3.605	9.594	5.531	13.486
IWD	0.251	3.628	9.645	5.559	13.591
MBO	0.259	3.659	9.712	5.588	13.715
PSO	0.265	3.682	9.772	5.612	13.832
SBO	0.270	3.709	9.825	5.635	13.978
SOS	0.275	3.734	9.873	5.658	14.102
WEO	0.282	3.757	9.929	5.683	14.254

The table above and Figure 10 provide a comparison of various optimization algorithms, including the proposed algorithm in solving the welded beam design problem. Based on the values presented, it appears that the HawkFish algorithm outperforms the other algorithms with respect to the criteria of h ($x1$), l ($x2$), t ($x3$), b ($x4$), and *f_cost*. This suggests that the HawkFish algorithm may be a more effective method for tackling this problem, as it yields a lower *f_cost*, indicating a better optimized solution. The above experiments and observations support the proposed algorithm's ability to solve complex problems with unknown search spaces.

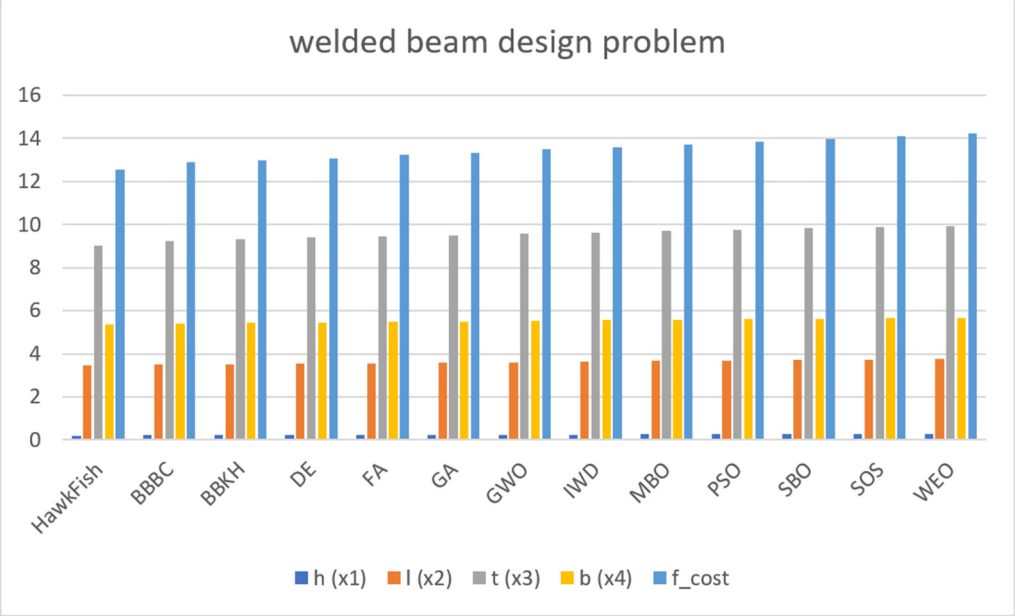


Figure 10. Comparative performance of various optimization algorithms for welded beam design problem.

5. Solving the Tension/Compression Spring Problem

The tension/compression spring (TCS) design problem (Figure 11) is an optimization problem commonly encountered in mechanical engineering, where the goal is to design a spring that meets specific performance criteria while minimizing or maximizing certain objectives, such as weight, cost, or material usage [47]. The design involves various constraints to ensure that the spring functions correctly under the given loads and conditions [48].

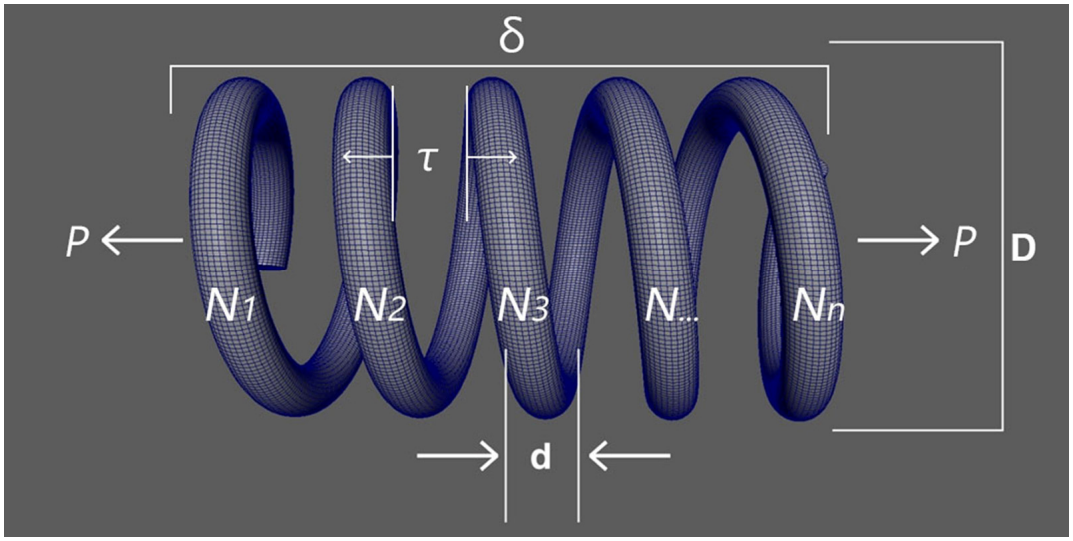


Figure 11. Constraints of the TCS problem.

5.1. Key Parameters and Variables

The design of a spring involves several key parameters that influence its performance. The wire diameter (d) refers to the diameter of the wire used to fabricate the spring, while the mean coil diameter (D) is the average diameter of the spring's coils. The number of active coils (N) indicates the number of coils that actively contribute to the spring's deflection, and the modulus of rigidity (G) is a material property that reflects the rigidity of the spring's material. Additionally, the shear stress (τ) represents the stress acting parallel to the wire's cross-section, and the deflection (δ) denotes the displacement of the

spring under an applied load. The spring index (C) is a critical ratio defined as the mean coil diameter divided by the wire diameter ($C = D/dC$), and the load (P) represents the force exerted on the spring. Together, these parameters determine the spring's mechanical behavior and suitability for specific applications.

5.2. Objective Function and Constraints

The objective function for the tension/compression spring (TCS) design problem typically focuses on optimizing specific parameters to achieve the desired performance. Common objectives include minimizing weight by reducing the material used in the spring, minimizing cost by lowering manufacturing and material expenses, and maximizing efficiency to ensure the spring performs its function with minimal material while maintaining maximum durability. Several constraints ensure the spring performs correctly and safely. Typical constraints include:

1. Shear stress constraint: Ensuring the shear stress in the spring does not exceed the material's allowable stress.

$$\tau = 8PD/(\pi\, d^3) \leq \tau max \tag{26}$$

2. Deflection constraint: Ensuring the spring's deflection under load does not exceed the permissible deflection.

$$\delta = 8PD^3N/(Gd^4) \leq \delta max \tag{27}$$

3. Frequency constraint: Ensuring the spring's natural frequency is within a desired range to avoid resonance.

$$f = 1/(2\pi)\ \sqrt{(Gd^4/(8D^3N))} \geq f\_min \tag{28}$$

4. Geometric constraints: Ensuring the dimensions of the spring are within practical limits.

$$D\_min \leq D \leq D\_max \tag{29}$$

$$d\_min \leq d \leq d\_max \tag{30}$$

Given these variables, objectives, and constraints, the optimization problem can be formulated as follows:

$$\tau - \tau\_max \leq 0 \tag{31}$$

$$\delta - \delta\_max \leq 0 \tag{32}$$

$$f\_min - f \leq 0 \tag{33}$$

$$D\_min \leq D \leq D\_max \tag{34}$$

$$d\_min \leq d \leq d\_max \tag{35}$$

5.3. Solving the TCS Design Problem Using HawkFish Optimization Algorithm

Using the dynamic gender-based subpopulation strategy into the proposed algorithm, we can encourage different levels of exploration and exploitation within the search space. This potentially leads to a more diverse set of candidate solutions for finding the optimal values of d, D, and N that minimize the objective function while satisfying all the constraints, the steps of the optimization are further explained below:

1. Initialization:
   - Define the total population P of artificial fish.
   - Divide P into two subpopulations: P_{male} and P_{female}.
   - Each subpopulation has a unique visual scope: V_{male} for P_{male} and V_{female} for P_{female}.
2. Assign visual scopes V_{male} and V_{female} to the male and female subpopulations, respectively.
3. Evaluate the Availability of Food
   We define a food metric F representing the availability of resources in the search space. This could be the average fitness of the population or another suitable measure.
   - For each iteration i:
   - We compare the food metric F to a predefined threshold $F\_threshold$.
   - If $F > F\_threshold$, change the gender of each fish, f (e.g., male to female, and vice versa).
   - For each artificial fish f in P_{male} or P_{female}, consider only other fish within the corresponding visual scope V_{male} or V_{female} for evaluating fitness and updating positions.
   - Update the position of each fish f using the equation:

$$x\_f(t+1) = x\_f(t) + step\_size \times (x\_center - x\_f(t)) \quad (36)$$

   - Here, $x\_f(t)$ is the position of fish f at iteration t, $step\_size$ is a predefined parameter, and x_center is the center of the corresponding subpopulation.
   - Calculate the center of subpopulation P_{male} or P_{female} using:

$$x\_center = (1/|P_{male}|) \times \Sigma\, x\_j \text{ or } x\_center = (1/|P_{female}|) \times \Sigma\, x\_j \quad (37)$$

   - Here, x_j represents the position of the j-th fish in the respective subpopulation.
   - Identify neighboring fish N_f within the visual scope:

$$N\_f = \{g \mid g \in P_{male} \text{ or } P_{female} \text{ and } dist(x\_f(t), x\_g(t)) \leq V_{male} \text{ or } V_{female}\} \quad (38)$$

   - Update the position of fish f based on the best neighboring fish x_nbest:

$$x\_f(t+1) = x\_f(t) + step\_size \times (x\_nbest - x\_f(t)) \quad (39)$$

   - Here, x_nbest is the position of the best neighboring fish in N_f, based on the objective function.

In Table 7 below, each algorithm employs a different search strategy and heuristic to explore the solution space. As a result, they may converge to different local minima or optima, leading to different values for design parameters such as wire diameter. The wire diameter (along with other parameters) is considered a result of the optimization process. It reflects the specific design that each algorithm proposes as the best solution under the given constraints and objective function. In theory, the tension/compression spring (TCS) design problem is formulated with fixed constraints for the wire diameter (d) across all algorithms. However, in this proposed scenario the optimization algorithms are allowed to independently search for their own optimal values of d as part of the design variables, therefore variations in the wire diameter occurred.

Each algorithm's unique approach to exploring and exploiting the search space can lead to different optimized values for these parameters, demonstrating the diversity and effectiveness of each method in finding a solution to the TCS design problem. The objective is to minimize wire and coil diameters while satisfying other design constraints (cost and free length), and the proposed algorithm performed better than the other algorithms, as seen in Figures 12 and 13 below:

Table 7. Comparing the TCS design problem solving of the HawkFish algorithm with different optimization methods.

Algorithm	Wire Diameter (*d*)	Mean Coil Diameter (*D*)	Free Length (*L*)	Objective Function (*f_cost*)
Proposed	0.210	3.450	5.331	12.506
BBBC	0.215	3.503	5.412	12.921
BBKH	0.218	3.519	5.435	12.987
DE	0.224	3.534	5.455	13.092
FA	0.236	3.565	5.482	13.243
GA	0.237	3.589	5.508	13.341
GWO	0.245	3.605	5.531	13.486
IWD	0.251	3.628	5.559	13.591
MBO	0.259	3.659	5.588	13.715
PSO	0.265	3.682	5.612	13.832
SBO	0.271	3.709	5.635	13.978
SOS	0.275	3.734	5.658	14.102
WEO	0.282	3.757	5.683	14.254

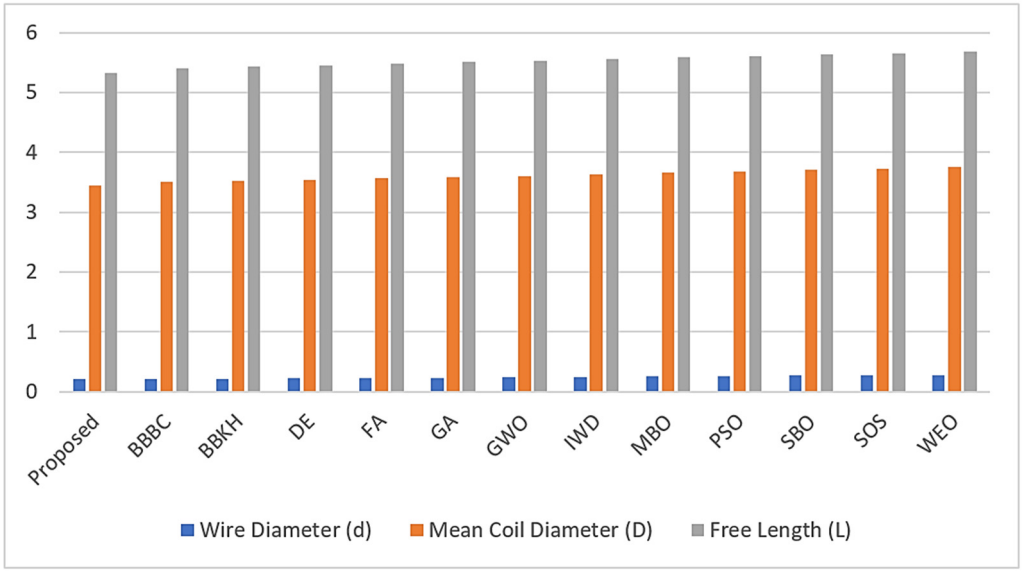


Figure 12. Comparative performance of various optimization algorithms for tension/compression spring (tcs) design problem in terms of free length, wire diameter, and mean coil diameter, measured in mm.

Figure 14 below illustrates the comparison of execution time for different optimization algorithms across three types of problems: CEC (2019), welded beams, and tensions compression. HawkFish proves to be a highly efficient algorithm in terms of execution time for most cases. The slight exception in the CEC case indicates room for improvement, but its better fitness performance compensates for this drawback. This suggests that HawkFish is not only computationally efficient, but also delivers high-quality solutions, which is a critical metric in optimization problems.

In Figure 14, the *x*-axis represents the optimization algorithms, including HawkFish (the proposed algorithm), BBBC, BBKH, DE, FA, GA, GWO, IWD, MBO, PSO, SBO, SOS, and WEO. The *y*-axis shows the execution time in seconds, presented in scientific notation (e.g., $7.00 \times 10^{-1} = 0.7$ s) for clarity and consistency given the varying magnitudes of execution times. The HawkFish algorithm demonstrates superior efficiency with the lowest execution times across most cases (welded beam and tensions compression problems) while achieving competitive fitness in the CEC/GECCO 2019 case despite a slightly higher execution time.

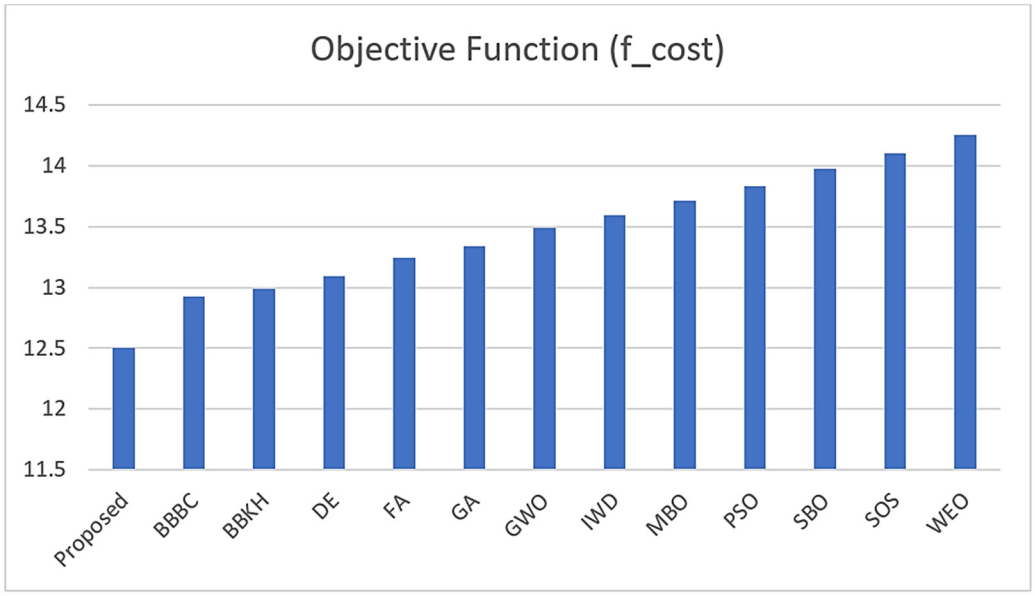


Figure 13. Comparison of optimization algorithms for the tension/compression spring (TCS) design problem, showing objective function (*f_cost*) as constraints measured in MATLAB units.

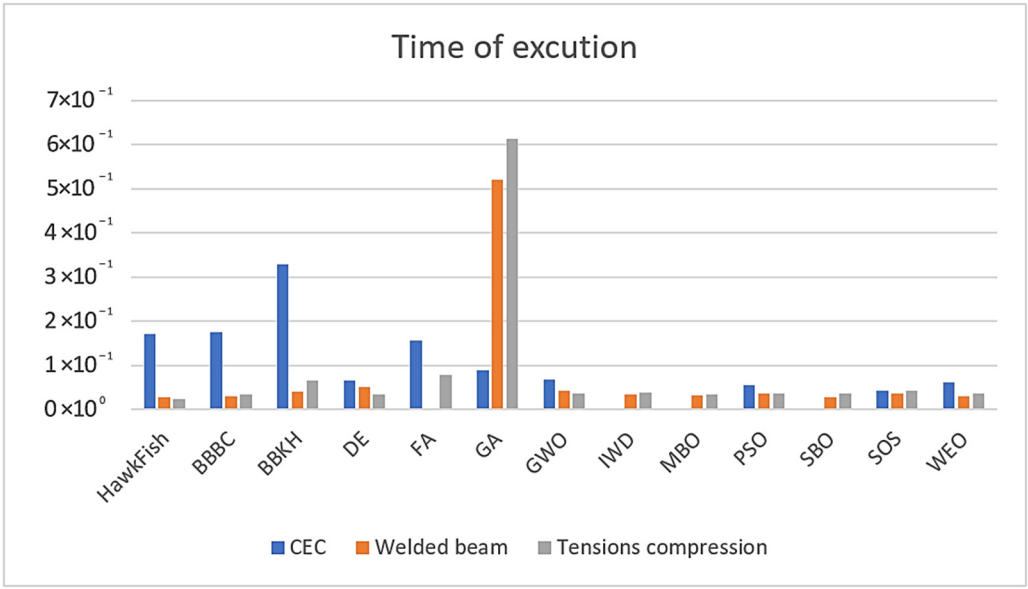


Figure 14. Execution time comparison across SOTA algorithms.

6. Conclusions

The HawkFish optimization algorithm demonstrated superior performance in solving benchmark optimization problems, including the welded beam design problem and the tension/compression spring problem, when compared to state-of-the-art optimization methods such as BBBC, BBKH, DE, FA, GA, PSO, and GWO. The proposed algorithm exhibited low error rates and efficient convergence across various population sizes and iterations, as evidenced by the analysis. The algorithm maintained consistent accuracy with increasing population sizes while minimizing computation time, achieving an error rate as low as 0.31 with a population of 10 and an average error rate below 0.60 for larger populations. The proposed method outperformed other optimization techniques in optimizing both unimodal and high-dimensional CEC-06 benchmark functions. The HawkFish algorithm achieved significantly lower fitness function values compared to traditional algorithms, indicating its ability to explore and exploit the solution space effectively. The proposed

algorithm also achieved the lowest cost functions in solving engineering design problems, such as the welded beam design problem and the tension/compression spring problem. The algorithm consistently provided optimal or near-optimal solutions, outperforming other algorithms across all evaluated dimensions, including beam thickness, length, and wire diameter. The success of the HawkFish algorithm can be attributed to the use of two opposing fitness functions, which allowed the algorithm to balance the exploration and exploitation of the search space. By using this approach, our algorithm was able to avoid local optima and achieve high-quality solutions in a shorter amount of time. The practical implications of our research are significant, as it has the potential to improve the efficiency and accuracy of various optimization tasks in different domains, such as engineering design, finance, and logistics, among others. The proposed method provides an effective and efficient optimization solution that can be used to address real-world optimization problems. In terms of future research, we believe that further investigation into the behavior and performance of our algorithm is necessary. This includes evaluating the algorithm's performance on larger and more complex problems, as well as exploring the possibility of using different types of fitness functions. Additionally, more research can be conducted to determine the algorithm's performance on mechanical problems with different characteristics and to further improve its efficiency.

Author Contributions: A.A. conceived the idea of the HawkFish Optimization Algorithm (HFOA) and designed its structure, incorporating the gender-based interaction mechanism inspired by hawkfish behavior. He implemented the algorithm in MATLAB, conducted simulations, and performed optimization experiments to evaluate its performance, as well as analyzing the results, preparing visualizations, and writing the majority of the manuscript. O.A. provided theoretical insights and feedback to refine the algorithm, contributed to the mathematical modeling and problem formulation, reviewed the experimental design, and validated the findings through comparative analysis with other optimization methods. Both authors contributed equally to the discussion of results. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author(s).

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

References

1. Rice, J.R. The Algorithm Selection Problem. *Adv. Comput.* **1976**, *15*, 65–118.
2. Morales-Castañeda, B.; Zaldivar, D.; Cuevas, E.; Fausto, F.; Rodríguez, A. A better balance in metaheuristic algorithms: Does it exist? *Swarm Evol. Comput.* **2020**, *54*, 100671. [CrossRef]
3. Kaur, K.; Kumar, Y. Swarm intelligence and its applications towards various computing: A systematic review. In Proceedings of the 2020 International Conference on Intelligent Engineering and Management (ICIEM), London, UK, 17–19 June 2020; pp. 57–62.
4. Talbi, E.G. *Metaheuristics: From Design to Implementation*; John Wiley & Sons: Hoboken, NJ, USA, 2009; Volume 74.
5. Mirjalili, S.; Gandomi, A.H.; Mirjalili, S.Z.; Mirjalili, A.S.; Saremi, M. Salp swarm algorithm: A bio-inspired optimizer for engineering design problems. *Adv. Eng. Softw.* **2017**, *114*, 163–191. [CrossRef]
6. Dragoi, E.N.; Dafinescu, V. Review of Metaheuristics Inspired from the Animal Kingdom. *Mathematics* **2021**, *9*, 2335. [CrossRef]
7. Squires, M.; Tao, X.; Elangovan, S.; Gururajan, R.; Zhou, X.; Acharya, U.R. A novel genetic algorithm-based system for the scheduling of medical treatments. *Expert Syst. Appl.* **2022**, *195*, 116464. [CrossRef]
8. Wu, Z.; Shen, D.; Shang, M.; Qi, S. Parameter Identification of Single-Phase Inverter Based on Improved Moth Flame Optimization Algorithm. *Electr. Power Compon. Syst.* **2019**, *47*, 456–469. [CrossRef]
9. Kaul, S.; Kumar, Y. Nature-inspired metaheuristic algorithms for constraint handling: Challenges, issues, and research perspective. In *Constraint Handling in Metaheuristics and Applications*; Kulkarni, A.J., Mezura-Montes, E., Wang, Y., Gandomi, A.H., Krishnasamy, G., Eds.; Springer: Singapore, 2021. [CrossRef]
10. Kaur, G.; Arora, S. Chaotic whale optimization algorithm. *J. Comput. Des. Eng.* **2018**, *5*, 275–284. [CrossRef]

11. Qiao, W.; Yang, Z. Solving large-scale function optimization problem by using a new metaheuristic algorithm based on quantum dolphin swarm algorithm. *IEEE Access* **2019**, *7*, 138972–138989. [CrossRef]
12. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2022**, *6*, 182–197. [CrossRef]
13. Yang, Y.; Liao, Q.; Wang, J.; Wang, Y. Application of multi-objective particle swarm optimization based on short-term memory and K-means clustering in multi-modal multi-objective optimization. *Eng. Appl. Artif. Intell.* **2022**, *112*, 104866. [CrossRef]
14. Amuso, V.J.; Enslin, J. The Strength Pareto Evolutionary Algorithm 2 (SPEA2) applied to simultaneous multi-mission waveform design. In Proceedings of the 2007 International Waveform Diversity and Design Conference, Pisa, Italy, 4–8 June 2007; pp. 407–417. [CrossRef]
15. Zhou, Q.; Zhang, Z.; Zhang, G. A multiobjective evolutionary algorithm based on decomposition and probability model. In Proceedings of the 2012 IEEE Congress on Evolutionary Computation, Brisbane, QLD, Australia, 10–15 June 2012; pp. 1–8. [CrossRef]
16. Cunha, M.; Marques, J. A new multiobjective simulated annealing algorithm—MOSA-GR: Application to the optimal design of water distribution networks. *Water Resour. Res.* **2020**, *56*, e2019WR025852. [CrossRef]
17. Salam, M.A. Knacks of Evolutionary Mating Heuristics for Renewable Energy Source–Based Power Systems Signal Harmonics Estimation. *Int. J. Energy Res.* **2024**, *48*, 457–469. [CrossRef]
18. Agushaka, J.O.; Ezugwu, A.E.; Abualigah, L. Gazelle optimization algorithm: A novel nature-inspired metaheuristic optimizer. *Neural Comput. Appl.* **2023**, *35*, 4099–4131. [CrossRef]
19. Caves, E.M.; de Busserolles, F.; Kelley, L.A. Sex differences in behavioural and anatomical estimates of visual acuity in the green swordtail, Xiphophorus helleri. *J. Exp. Biol.* **2021**, *224*, jeb243420. [CrossRef]
20. Al-Janabi, S.; Kadhum, G. Synthesis Biometric Materials Based on Cooperative Among (DSA, WOA and gSpan-FBR) to Water Treatment. In Proceedings of the 12th International Conference on Soft Computing and Pattern Recognition (SoCPaR 2020), Virtual, 15–18 December 2020; Abraham, A., Ed.; Springer: Cham, Switzerland, 2021. [CrossRef]
21. Kemp, D.B.; Fox, M.J.; Coates, D.J. Distribution and abundance of four color morphs of Paracirrhites forsteri across habitats in the Red Sea. *PLoS ONE* **2017**, *12*, e0169079. [CrossRef]
22. Drew, K.L.; Reece, M.; Augspurger, C.K. Genetic variation and population connectivity of Cirrhitichthys oxycephalus across geographic ranges. *PeerJ* **2023**, *7*, e18058. [CrossRef]
23. Jankowski, J.A.; Walker, J.H.; White, M.E. Habitat selectivity and live coral dependence of hawkfishes (Cirrhitidae). *PLoS ONE* **2015**, *10*, e0138136. [CrossRef]
24. Al-Janabi, S.; Salman, A.H. Optimization Model of Smartphone and Smart Watch Based on Multi Level of Elitism (OMSPW-MLE). In *Artificial Intelligence for Cloud and Edge Computing*; Misra, S., Kumar, T.A., Piuri, V., Garg, L., Eds.; Springer: Cham, Switzerland, 2022. [CrossRef]
25. Feng, Y.; Zhao, S.; Liu, H. Analysis of network coverage optimization based on feedback k-means clustering and artificial fish swarm algorithm. *IEEE Access* **2020**, *8*, 42864–42876. [CrossRef]
26. Gao, Y.; Xie, L.; Zhang, Z.; Fan, Q. Twin support vector machine based on improved artificial fish swarm algorithm with application to flame recognition. *Appl. Intell.* **2020**, *50*, 2312–2327. [CrossRef]
27. Singh, S.P.; Dhiman, G.; Tiwari, P.; Jhaveri, R.H. A soft computing based multi-objective optimization approach for automatic prediction of software cost models. *Appl. Soft Comput.* **2021**, *113*, 107981. [CrossRef]
28. Varga, D. Full-Reference Image Quality Assessment Based on an Optimal Linear Combination of Quality Measures Selected by Simulated Annealing. *J. Imaging* **2022**, *8*, 224. [CrossRef]
29. Rosso, S.; Uriati, F.; Grigolato, L.; Meneghello, R.; Concheri, G.; Savio, G. An optimization workflow in design for additive manufacturing. *Appl. Sci.* **2021**, *11*, 2572. [CrossRef]
30. Sharma, P.; Said, Z.; Kumar, A.; Nižetić, S.; Pandey, A.; Hoang, A.T.; Huang, Z.; Afzal, A.; Li, C.; Le, A.T.; et al. Recent advances in machine learning research for nanofluid-based heat transfer in renewable energy systems. *Energy Fuels* **2022**, *36*, 6626–6658. [CrossRef]
31. Abasi, A.K.; Khader, A.T.; Al-Betar, M.A.; Naim, S.; Alyasseri, Z.A.A. A novel hybrid multi-verse optimizer with K-means for text documents clustering. *Neural Comput. Appl.* **2020**, *32*, 17703–17729. [CrossRef]
32. Deng, W.; Shang, S.; Cai, X.; Zhao, H.; Song, Y.; Xu, J. An improved differential evolution algorithm and its application in optimization problem. *Soft Comput.* **2021**, *25*, 5277–5298. [CrossRef]
33. Suganthan, P.N.; Deb, K.; Mallipeddi, R.; Pan, Q.K.; Qin, A.K. *Problem Definitions and Evaluation Criteria for the CEC/GECCO 2019 Competition on Real-Parameter Single Objective Optimization, Technical Report*; Nanyang Technological University: Singapore, 2019.
34. Maree, S.C.; Alderliesten, T.; Bosman, P.A.N. Benchmarking HillVallEA for the GECCO 2019 Competition on Multimodal Optimization. *arXiv* **2019**, arXiv:1907.10988.
35. Abdullah, J.M.; Ahmed, T. Fitness Dependent Optimizer: Inspired by the Bee Swarming Reproductive Process. *IEEE Access* **2019**, *7*, 43473–43486. [CrossRef]

36. Amin, A.A.H.; Aladdin, A.M.; Hasan, D.O.; Mohammed-Taha, S.R.; Rashid, T.A. Enhancing Algorithm Selection through Comprehensive Performance Evaluation: Statistical Analysis of Stochastic Algorithms. *Computation* **2023**, *11*, 231. [CrossRef]
37. Wang, H.; Yi, J.H. An improved optimization method based on krill herd and artificial bee colony with information exchange. *Memetic Comput.* **2018**, *10*, 177–198. [CrossRef]
38. Alyasseri, Z.A.A.; Khader, A.T.; Al-Betar, M.A.; Abasi, A.K. EEG signals denoising using optimal wavelet transform hybridized with efficient metaheuristic methods. *IEEE Access* **2019**, *8*, 10584–10605. [CrossRef]
39. Alkharsan, A. HawkFish Optimization Algorithm, GitHub Repository. Available online: https://github.com/AliAlkharsan93/HawkFish-Optimization (accessed on 3 January 2025).
40. Alomari, O.A.; Makhadmeh, S.N.; Al-Betar, M.A.; Alyasseri, Z.A.A.; Abu Doush, I.; Abasi, A.K.; Awadallah, M.A.; Abu Zitar, R. Gene selection for microarray data classification based on Gray Wolf Optimizer enhanced with TRIZ-inspired operators. *Knowl.-Based Syst.* **2021**, *223*, 107034. [CrossRef]
41. Makhadmeh, S.N.; Al-Betar, M.A.; Alyasseri, Z.A.A.; Abasi, A.K.; Khader, A.T.; Damaševičius, R.; Mohammed, M.A.; Abdulkareem, K.H. Smart Home Battery for the Multi-Objective Power Scheduling Problem in a Smart Home Using Grey Wolf Optimizer. *Electronics* **2021**, *10*, 447. [CrossRef]
42. Makhadmeh, S.N.; Khader, A.T.; Al-Betar, M.A. A novel hybrid grey wolf optimizer with min-conflict algorithm for power scheduling problem in a smart home. *Swarm Evol. Comput.* **2021**, *60*, 100793. [CrossRef]
43. Binh, H.T.T.; Thanh, P.D. Two levels approach based on multifactorial optimization to solve the clustered shortest path tree problem. *Evol. Intell.* **2020**, *15*, 185–213. [CrossRef]
44. Hao, X.X.; Qu, R.; Liu, J. A unified framework of graph-based evolutionary multitasking hyper-heuristic. *IEEE Trans. Evol. Comput.* **2021**, *25*, 35–47. [CrossRef]
45. Bali, K.K.; Gupta, A.; Ong, Y.-S.; Tan, P.S. Cognizant multitasking in multiobjective multifactorial evolution: MO-MFEA-II. *IEEE Trans. Cybern.* **2020**, *51*, 1784–1796. [CrossRef]
46. Xu, Z.W.; Zhang, K.; Xu, X.; He, J.J. A fireworks algorithm based on transfer spark for evolutionary multitasking. *Front. Neurorobotics* **2020**, *13*, 109. [CrossRef]
47. Lin, J.B.; Liu, H.L.; Xue, B.; Zhang, M.J.; Gu, F.Q. Multi-objective multi-tasking optimization based on incremental learning. *IEEE Trans. Evol. Comput.* **2020**, *24*, 824–838. [CrossRef]
48. Zhou, Y.J.; Wang, T.H.; Peng, X.G. MFEA-IG: A multi-task algorithm for mobile agents path planning. In Proceedings of the IEEE Congress on Evolutionary Computation, Glasgow, UK, 1–7 July 2020; pp. 1–7.