machines

MDPI

Article

Multi-Strategy Sailfish Optimizer: Novel Algorithm with Dynamic Sardine Population and Improved Search Technique for Efficient Robot Path Planning

Saboohi Naeem Ahmed [1], Muhammad Rizwan Tanweer [2], Adnan Ahmed Siddiqui [3], Salman A. Khan [4], Muhammad Hassan Tanveer [5,*] and Razvan Cristian Voicu [5]

[1] Faculty of Engineering Science and Technology, Hamdard University, Karachi 74600, Pakistan; saboohi.naeem@hamdard.edu
[2] Department of Decision Sciences, Karachi School of Business and Leadership, Karachi 74800, Pakistan; rizwan.tanweer@ksbl.edu.pk
[3] Department of Computer Science, Usman Institute of Technology (UIT) University, Karachi 75300, Pakistan; aasiddiqui@uitu.edu.pk
[4] College of Computing and Information Science, Karachi Institute of Economics and Technology, Karachi 74700, Pakistan; sakhan@kiet.edu.pk
[5] Department of Robotics and Mechatronics Engineering, Kennesaw State University, Marietta, GA 30060, USA; rvoicu@kennesaw.edu
* Correspondence: mtanveer@kennesaw.edu

Abstract

The sailfish optimizer is a recent swarm-intelligence-based optimization algorithm which mimics the hunting behavior of sailfish in the ocean. It consists of two types of populations, namely, sailfish and sardine, where sailfish represent the candidate solutions and sardines freely maneuver in the search space. Existing research studies have shown that the sailfish optimizer suffers from limited global exploration capability, with local optimum stagnation and slow convergence speed. To address these limitations, an improved sailfish optimizer, namely, the Multi-Strategy Sailfish Optimizer, is proposed in this study. This improved version incorporates a modified search strategy for both sailfish and sardines, a non-linear attack power parameter, an improved hunting procedure, and a dynamic sardine population. The impact of all suggested improvements is analyzed experimentally. Several experiments with single-objective problems presented at the Congress on Evolutionary Computation in 2022 are performed to prove the effectiveness and efficiency of the proposed algorithm. This improved algorithm is compared with well-known optimization algorithms, such as the whale optimization algorithm, the sine–cosine algorithm, etc., and improved variants of those algorithms. A convergence behavior analysis is also performed using convergence graphs. Furthermore, the significance of the work is statistically validated. The analysis indicates that the Multi-Strategy Sailfish Optimizer performs significantly better than other optimization algorithms. It is also applied to solve the tension/compression spring design problem and the mobile robot path planning problem.

Keywords: improved sailfish optimizer; evolutionary computation; swarm intelligence; Friedman test; mobile robot path planning

Check for updates

Academic Editor: Philip Moore

Received: 3 November 2025
Revised: 15 December 2025
Accepted: 18 December 2025
Published: 28 December 2025

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Nature-inspired optimization algorithms (NIOAs) are a set of algorithms that are useful for finding optimal/near-optimal solutions to complex problems, such as

the traveling salesman problem (TSP) [1], robot path planning [2], natural language processing [3], scheduling problems [4,5], complex engineering problems [6–8], optimizing network topology design [9], classification of ECG signals [10], manufacturing processes [11], electrical engineering [12], and control engineering [13], among others. Applications of NIOAs in diversified areas of engineering and the sciences have made them the preferred choice of the research community [14–16]. The methodology of these algorithms is inspired by several natural phenomena, such as birds searching for food, whales or sailfish hunting prey in the ocean, evolutionary theory, arithmetic operations, and gravity concepts. The use of nature-inspired algorithms began with the advent of the genetic algorithm (GA) [17], which is based on Darwin's theory of evolution. Later, evolutionary strategy (ES) [18], differential evolution (DE) [19], and ant colony optimization (ACO) [20] were introduced. The inception of particle swarm intelligence (PSO) [21], in which the behavior of birds in the environment when they search for food is simulated, was the game changer. Thereafter, many studies related to stochastic nature algorithms emerged, which include the bat algorithm (BA) [22], the sine–cosine algorithm (SCA) [23], the whale optimization algorithm (WOA) [24], the gray wolf optimizer (GWO) [25], the white shark optimization (WSO) algorithm [26], the teaching–learning-based optimizer (TLBO) [27], the sailfish optimizer [28], the arithmetic optimization algorithm (AOA) [29], and the chimp optimization (CHO) algorithm [30], among many others.

All NIOAs perform optimization using two common search behaviors known as exploration and exploitation. In NIOAs, these two behaviors are implemented using different strategies, leading to various nomenclatures for the classification of these algorithms. Hui Li et al. [31] classified NIOAs as single-elite, multi-elite, and non-elite algorithms. Similarly, Siddique and Adeli [32] provided a classification of algorithms based on physics, chemistry, or biology, which are further divided into evolutionary algorithms (EAs), bio-inspired algorithms, and swarm-intelligence-based algorithms (SIAs) [14]. Another categorization is based on the equation/procedure presented by Yang [15]. In a recent study by Moshtaghi et al. [16], all the aforementioned categories are presented in a novel manner based on the country of origin.

Among all the categories of NIOAs, SIAs have been immensely popular (over 94,000 citations for PSO as of September 2025). The common search behaviors of NIOAs are exploration (search for the optimal solution in a global search space) and exploitation (refining a specific search region to reach the best value). In each search, the algorithm attempts to update the position of each participant with the given technique iteratively, hence reaching an optimal or near-optimal value. Research studies have shown that SIAs suffer from local optimum stagnation, poor exploration capabilities, and slow convergence [33]. Therefore, various improved algorithms addressing those limitations have been introduced, including the improved sine–cosine algorithm (ISCA) [34], the directionally driven self-regulating PSO algorithm [35], the improved salp swarm algorithm (ISSA) [36], the balanced teaching–learning-based optimizer (BTLBO) [37], and the improved whale optimization algorithm (IWOA) [38], among others. The main motivation behind all these improved techniques is the "No Free Lunch Theorem" (NFL) [39], which implies that no single optimization technique is a master of all. Instead, it is possible to have an algorithm, "A", perform better for one set of problems, while the same algorithm may not be suitable for other sets of problems.

Motivated by the NFL theorem, this study attempts to improve an existing SIA, namely, the sailfish optimizer (SFO) [28], which has been applied in various optimization problems. Despite its popularity, the SFO suffers from slow convergence and local optimum stagnation due to a weak exploration phase [6]. Also, the SFO uses two types of populations, namely, sailfish and sardines, which results in increased algorithmic complexity. Few studies have contributed to the improvement of the SFO. Therefore, there is ample room for improvement

in the selected algorithm. Accordingly, the key contributions of the present study are listed as follows:

- A review of the existing literature on the SFO is presented, discussing all the variants and applications present in the literature.
- An improved version of the SFO, namely, the Multi-Strategy Sailfish Optimizer (MSSFO), is designed with an improved learning strategy for both sardines and sailfish, a non-linear attack power parameter, a modified hunting technique, and a dynamic sardine population.
- The impact of all the proposed techniques is analyzed by applying each modification to the SFO individually; then a fully modified version is prepared, and the results are presented along with a statistical evaluation and convergence graphs.
- The experimental evaluation of the MSSFO is performed in comparison with the original SFO, well-known optimization algorithms, and improved variants of the selected algorithms. Experiments are performed using the CEC 2022 single-objective bound-constrained optimization (SO-BO) functions. The results are presented with rank-based analysis and statistical evaluation.
- The MSSFO is successfully applied to solve two real-world optimization problems, the tension/compression spring design (CSD) problem and the mobile robot path planning problem.

The remainder of this paper is organized as follows: Section 2 presents the review of the literature. Section 3 provides the basic algorithm with mathematical formulation of the SFO and the SCA, and then the design of the MSSFO is presented with the improvements suggested in the current study. Section 4 is dedicated to parameter sensitivity analysis, analysis of the proposed strategy, experimental results, and a discussion along with statistical validation, convergence analysis, and real-world applications. Section 5 presents the discussion of the results. Section 6 concludes the work presented in this study.

2. Literature Review

The SFO [28] was introduced in 2019. Since then, the algorithm, along with a few improved versions, has been applied to various real-world problems. In this section, the variants of the SFO are discussed, and then the applications presented in the literature are summarized.

2.1. Variants of the Sailfish Optimizer

In response to the challenges posed by the SFO, many researchers have investigated the algorithm to identify and propose novel improvements aimed at refining its search capabilities and overall efficiency. This subsection is devoted to presenting modifications of the SFO available in the literature. The first improvement was proposed in the year 2019 by Hammouti et al. [40] to enhance the local search capabilities in the last stages of the SFO, generating two new solutions for each population type and then choosing the best one.

Numerous researchers have studied the exploitation behavior of the SFO and suggested improvements for better convergence. Ghosh et al. [41] integrated another technique called beta hill climbing with the SFO to improve the exploitation capability. For mapping continuous values to binary ones, a sigmoid transfer function is used. Zhang and Mo [6] proposed three improvements, including a chaotic initial population, an adaptively decreasing attack power parameter, and a modified equation to update the position of the sardine. In another study [42], the same authors proposed an adaptive t-distribution mechanism, where the survival of the fittest and the selection operator in the GA is applied to consider high-quality solutions and discard the worst ones. Mohammadi et al. [43] incorporated a weight factor (w) into the search strategy of both the sailfish and the sardines

and applied this improved SFO for efficient clustering in IoT devices in software-defined networks (SDNs).

To enhance the global search formula and to resolve premature convergence of the SFO, Liao et al. [44] incorporated a leak-proof net and a cross-mutation propagation mechanism. Li et al. [8] modified the SFO in three ways. Firstly, the local search is modified using an inertia weight. Thereafter, a modified global search formula is applied. Finally, the Levy flight strategy is applied to the best solutions to further refine them. Another recent study by Shajin et al. [45] also addressed premature convergence and weak global capabilities and proposed various improvements in the original algorithm. In their study, Levy flight is embedded in the position update equation. Furthermore, chaotic maps and opposition-based learning are used to generate an initial population.

Li Cao et al. [46] integrated the osprey optimization algorithm's search strategy in the SFO. Within the hunting phase, the Cauchy mutation is used to update the positions of the sailfish and sardines. Amin et al. [47] employed the Cauchy mutation mechanism to enhance the local exploration capability and introduced diversity and balance within the SFO. Babu et al. [48] modified the standard SFO by introducing an adaptive formula for updating the position of the sailfish during the optimization process. Numerous researchers have hybridized the SFO with the DE algorithm to enhance the overall performance, to design a multi-objective version [49], and to solve the feature selection problem [50].

The basic SFO operates within a continuous search space. However, problems that involve discrete or binary data cannot be solved with a continuous version of the algorithm. Hence, a need arises for a discrete version of the SFO tailored to address and effectively handle such problems [41]. In this context, Li et al. [51] and Togacar et al. [52] used a mapping function to convert continuous search space into the binary one.

A well-known technique to enhance the efficiency of an optimization algorithm is to implement it in a distributed or parallel setting. By doing so, the advantages of parallel or distributed programming can be leveraged to optimize the algorithm's performance. To accelerate the SFO, Naji et al. [53] and Shadravan et al. [54] implemented the algorithm in a distributed setting on a Graphics Processing Unit (GPU). They achieved performance enhancement over a sequential SFO. GPUs using Compute Unified Device Architecture (CUDA) were employed to meet the massive computation requirements in both approaches. Another parallel implementation of the SFO by Geetha et al. [55] showed a 50% improvement in the algorithm's running time. A summary of the improved variants of the SFO is provided in Table 1.

2.2. Applications of the Sailfish Optimizer

Despite limitations, the SFO (only two internal parameters) has been applied to various real-world problems, including the berth allocation problem, as well as problems related to wireless sensor networks, machine learning and deep learning, data mining, cloud computing, chemical processes, power system engineering, electrical engineering, and signal processing, among others. This section summarizes various applications of the SFO.

The berth allocation problem revolves around the efficient distribution of berth space among vessels at container terminals [40]. Specifically, the SFO was applied to solve the dynamic berth allocation problem [40,56]. Likewise, for recommendation of answers in a question–answer system [51], a cluster of related questions is formed using the SFO.

Wireless sensor networks (WSNs) have many areas that need optimization. The SFO has been utilized for cluster head selection [57,58], optimal path selection [59], prediction of blackholes in WSNs [60], and cluster optimization [61] in WSNs. Furthermore, control node selection using the multi-objective SFO is performed for software-defined WSNs [62].

The SFO is applied to a transmission network to improve the voltage profile and is able to minimize the total real power losses [63].

Table 1. Available variants of the SFO along with limitations addressed and improvements made.

References	Limitations Studied	Implemented Improvements
[40]	Weak exploitation phase.	Two fresh solutions are generated for each population type, and then the best one between them is selected when $AP < 0.5$.
[41]	Weak exploitation and continuous search space.	Search space is converted to the binary values. Adaptive β hill climbing is integrated.
[52]	Continuous search space.	Binary transfer function is used to create binary variant.
[51]	Continuous search space, local optimum stagnation, and single-objective nature of SFO.	A binary variant is created. Crossover and mutation operators of genetic algorithm are embedded in local search. Multi-objective version is designed.
[6]	Initial population is randomly generated. Local optimum stagnation and improper balance in exploration and exploitation.	Chaotic maps are used to generate the initial population. Attack power is adaptively decreased. Search equations are modified for sardines.
[42]	Slow convergence speed, premature convergence, and improper balance between global search and local search capabilities.	Chaotic population, adaptive t-distribution to balance exploration, exploitation, and genetic characteristics within sardine population.
[44]	Premature convergence.	Cross-mutation propagation mechanism.
[8]	Local optimum stagnation and improper balance in exploration and exploitation.	Weight inertia, global search formula, and Levy flight strategy.
[45]	Premature convergence and weak exploration.	Levy-flight strategy, opposition-based learning, and chaotic maps.
[53]	Sequential implementation takes a lot of time.	Distributed implementation.
[54]	Non-distributed implementation.	Distributed implementation
[55]	High computational speed and less accurate results.	Initial chaotic population and parallel implementation.

Feature selection is a process in machine learning and statistics where a subset of relevant and significant features or variables are chosen from a larger set [41]. Ghosh et al. [41] created a binary variant of the SFO to solve the well-known UCI datasets. Duhayyim et al. [64] proposed an extreme machine learning model (ELM) to detect oral cancer through images, where the SFO is employed to select the effective parameters of the ELM. Similarly, a data mining model by Hamza et al. [65] seeks to extract insights from educational data and utilized the SFO for the feature selection. In [66], a generative adversarial network (GAN) based on the SFO is developed to detect lung nodules in CT images. Khan et al. [67] presented a maximum power point tracking control, where a general regression neural network was trained with the sailfish optimizer. Another study [68] on the automatic identification of diseases in plant leaves used the SFO for the identification of the affected plant region. In [69], a deep convolution neural network for diagnosing sugarcane billet damage is proposed, where the SFO is utilized for sugarcane classification.

Cloud computing refers to the provision of IT resources over the Internet on a pay-as-you-go basis. Resource sharing and scheduling in cloud services require optimization, and numerous studies have been conducted for such processes. A visual cryptography model was proposed for image sharing in the cloud using the SFO [70]. A new scheduling algorithm for cloud was proposed in [4], and privacy preservation in cloud computing was addressed in [71]. An image classification model was proposed utilizing the binary sailfish optimization technique for effective feature selection [52].

Chemical dynamic processes are optimized based on an improved SFO [6]. In the context of the intelligent transportation system (ITS), an efficient algorithm for optimal charging schedule is proposed [5], leveraging the GWO and the SFO. A framework using the SFO was proposed for optimal selection of variational mode decomposition (VMD) parameters [72]. The research study in [73] implemented a fuzzy logic-based Tilt–Integral–Derivative (FTID) controller to enhance

the controller's performance, where the parameters of the FTID controller were optimized using the SFO.

Despite several diverse applications of the SFO, the performance of the SFO has not been studied in the mobile robot path planning problem, although other SI-based algorithms have been used to solve this problem. For example, an improved PSO-GWO algorithm was proposed for the path planning of robots [74]. Similarly, an improved GA and ACO are used for robot path planning in static environments in [75,76]. Motivated by these findings, the Multi-Strategy Sailfish Optimizer (MSSFO) proposed in this study is applied to the mobile robot path planning problem and to a well-known engineering design problem, namely, the compression spring design (CSD) problem.

2.3. An Overview of the Limitations of the Sailfish Optimizer

Despite its competitive performance on various optimization problems, the SFO exhibits several limitations identified in the previous subsections. One of the main weaknesses is its tendency towards premature convergence, where the algorithm may get trapped in local optima due to a weak search strategy [43,44,46]. Another weakness is the improper balance between exploration and exploitation phases of the SFO [6,45], specifically due to the linear attack power (AP) parameter. Additionally, the global search mechanism of the SFO often leads to a slow convergence rate during later stages of optimization [40,47], reducing the algorithm's efficiency when approaching the optimum solution. These weaknesses motivate the development of an improved SFO to enhance robustness, convergence speed, and solution accuracy.

3. Proposed Multi-Strategy Sailfish Optimizer

In this section, the improvements suggested in the MSSFO are discussed in detail. For a better understanding, the basic techniques, including the SFO and the SCA, are discussed first. This is followed by the improvements applied to the SFO along with the mathematical model. At the end, the pseudo-code and the flowchart of the MSSFO are provided, incorporating all modifications.

3.1. The Sailfish Optimizer (Original Version)

Being a nature-inspired optimization algorithm, the SFO [28] draws inspiration from the hunting behavior of sailfish, which are among the fastest marine predators. Like other metaheuristic algorithms, the SFO is designed to solve optimization problems. The algorithm models the interactions between the prey (sardines) and the predators (sailfish) in the ocean. The sardines represent potential moving solutions or candidates in the search space, while the sailfish represent actual candidates to be optimized. Sailfish in the natural world often use group hunting strategies to capture their prey. Similarly, the SFO employs a group-based approach, in which a population of sailfish collaboratively searches for the optimal solutions. Sailfish are known for their incredible speed and agility. In the algorithm, this speed and direction are used to guide the search. Sailfish adjust their positions and directions based on the quality of the solutions (sardines) they encounter. In the SFO, Equation (1) is used to update the position of sailfish.

$$X^i_{\text{new_SF}} = X^i_{\text{elite_SF}} - \lambda_i \times \left(\text{rand}(0,1) \times \left(\frac{X^i_{\text{elite_SF}} + X^i_{\text{injuredS}}}{2} \right) - X^i_{\text{old_SF}} \right) \quad (1)$$

where $X^i_{\text{elite_SF}}$ is the best position of sailfish, X^i_{injuredS} is the best position of sardines, $X^i_{\text{old_SF}}$ is the current position of sailfish, rand is a random number from 0 to 1, and λ_i is the coefficient at the ith iteration defined in Equation (2) as follows:

$$\lambda_i = 2 \times \text{rand}(0,1) \times PD - PD \tag{2}$$

where PD represents prey density, showing the number of sardines in each iteration. It is worth mentioning here that the sardines decrease at each iteration. The prey density is calculated as shown in Equation (3):

$$PD = 1 - \left(\frac{N_{\text{SF}}}{N_{\text{SF}} + N_S} \right) \tag{3}$$

where N_{SF} and N_{S} are the number of sailfish and the number of sardines, respectively. Also, N_{SF} is defined as $N_{\text{S}} \times$ PP, where PP is the percentage of the sardine population that initially forms the sailfish population. Sardine position is calculated using Equation (4):

$$X^i_{\text{new_S}} = r \times \left(X^i_{\text{elite_SF}} - X^i_{\text{old_S}} + AP \right) \tag{4}$$

where $X^i_{\text{old_S}}$ is the current position of sardines and $X^i_{\text{elite_SF}}$ is the best position of sailfish, r is a random number between 0 and 1, and AP is the attack power of the sailfish in each iteration represented by Equation (5):

$$AP = A \times (1 - (2 \times \text{Itr} \times \varepsilon)) \tag{5}$$

where Itr is the iteration number, and A and ε are the coefficients for decreasing AP linearly from A to 0. In the final stages of hunting, when a sardine becomes fitter than a sailfish (i.e., when a sardine has a lower cost compared with sailfish's cost), the sardine is inserted into the sailfish population using Equation (6):

$$X^i_{SF} = X^i_s \quad \text{if} \quad f(S_i) < f(SF_i) \tag{6}$$

The complete algorithm of the SFO is provided in Algorithm 1.

Algorithm 1 The sailfish optimizer [28].

```
Initialize the population of sailfish and sardines randomly based on PP
Initialize parameters A = 4, ε = 0.001
Compute the fitness of sailfish and sardines
Find the best sailfish and injured sardine
while termination conditions are not satisfied do
    for each sailfish do
        Calculate λ_i using Equation (2)
        Update the position of sailfish using Equation (1)
    end for
    Calculate Attack Power (AP) using Equation (5)
    if AP < 0.5 then
        Calculate α = N_S × AP
        Calculate β = d_i × AP
        Select a set of sardines based on α and β
        Update the position of selected sardines using Equation (4)
    else
        Update the position of all sardines using Equation (4)
    end if
    Calculate the fitness of all sardines
    if there is a better solution in sardine population then
        Replace a sailfish with injured sardine using Equation (6)
        Remove the hunted sardine from the population
        Update the best sailfish and injured sardine
    end if
end while
return best sailfish
```

3.2. Sine–Cosine Algorithm (Original Version)

The SCA [23] is a population-based optimization algorithm having mathematical models based on sine and cosine functions to explore and exploit the search space. It is a powerful yet simple technique for avoiding local optima and achieving convergence. It uses Equations (7) and (8) for updating particles in the search space:

$$X_i^{(t+1)} = X_i^t + r_1 \times \sin(r_2) \times \left| r_3 P_i^t - X_i^t \right| \tag{7}$$

$$X_i^{(t+1)} = X_i^t + r_1 \times \cos(r_2) \times \left| r_3 P_i^t - X_{ii}^t \right| \tag{8}$$

where X_i^t is the current position in the ith dimension and in the tth iteration; r_1, r_2, and r_3 are random numbers; P_i^t is the destination position (the global best position so far); and $||$ indicates the absolute value. The above two equations are combined in Equation (9):

$$X_i^{(t+1)} = \begin{cases} X_i^t + r_1 \times \sin(r_2) \times \left| r_3 P_i^t - X_i^t \right|, & \text{if } r_4 < 0.5 \\ X_i^t + r_1 \times \cos(r_2) \times \left| r_3 P_i^t - X_{ii}^t \right|, & \text{if } r_4 \geq 0.5 \end{cases} \tag{9}$$

where $r_2 \in [0, 2\pi]$, r_3 is randomly distributed between 0 and 2, and r_4 is in the range 0 and 1. r_1 is a key component to lead the search either globally or locally and is calculated using Equation (10):

$$r_1 = a - \frac{ta}{T} \tag{10}$$

The complete SCA algorithm is given in Algorithm 2.

Algorithm 2 Sine–cosine algorithm [23].

```
Initialize population (solutions) X
repeat
    Evaluate each search agent using the objective function
    Update the best solution obtained so far: P = X*
    Update r1, r2, r3, and r4
    Update the positions of search agents using Equation (9)
until t > maximum number of iterations
return the global best value
```

3.3. Multi-Strategy Sailfish Optimizer

This section provides a detailed description of the improvements applied to the SFO with regard to the mathematical model, pseudo-code, and flowchart of the proposed MSSFO. Firstly, the search strategies for sailfish and sardines are modified. Secondly, the worst solution is removed from the candidate solution, and the linear convergence factor, AP, is replaced with a non-linear random parameter. Furthermore, the MSSFO introduces a new concept to the original algorithm; when all sardines are hunted, more sardines are dynamically added to the sardine population. This diversifies the search space, thus improving the global optimization capabilities of the algorithm.

3.3.1. Improved Search Strategy

There are two types of population in the SFO. The first one is the sailfish that are the candidate solutions, and the second one is the sardine population. Despite the use of the sardine population, the SFO suffers from local optimum stagnation and slow convergence speed. Hence, it is much needed to redefine the search strategy. Inspired by the quantum behavior of particles in the PSO algorithm [77], the position update equation for sailfish (Equation (11)) is improved using the mean sailfish position. By using the mean position of the sailfish population, a collective intelligence effect is introduced to

improve divergence [78]. When the algorithm uses only elite sailfish and injured sardine, the search becomes biased towards one or two solutions, reducing the overall diversity of the solution [79]. Therefore, introducing the mean sailfish position will potentially improve the diversification of the search space. A modified search equation for the sailfish position is presented in Equation (11):

$$X^i_{\text{new_SF}} = X^i_{\text{elite_SF}} - \lambda_i \times \left(\text{rand}(0,1) \times \text{Mean}(X_{SF}) - X^i_{\text{old_SF}}\right) \tag{11}$$

The sardine population is used to enhance the exploration capabilities of the algorithm so that more areas in the search space are explored. However, the SFO still has weak exploration capability. To overcome this, the sine–cosine search strategy is employed to modify the position update equation of sardines as shown in Equations (12) and (13):

$$X^i_{\text{new_S}} = X^t_{\text{old_S}} + r_1 \times \sin(r_2) \times \left|r_3 X^i_{\text{elite_SF}} - X^t_{\text{old_S}}\right| \tag{12}$$

$$X^i_{\text{new_S}} = X^t_{\text{old_S}} + r_1 \times \cos(r_2) \times \left|r_3 X^i_{\text{elite_SF}} - X^t_{\text{old_S}}\right| \tag{13}$$

where $X^t_{\text{old_S}}$ is the current position of sardines in the t^{th} iteration, $r_2 \in [0, 2\pi]$, r_3 is in the range [0, 2], and $X^i_{\text{elite_SF}}$ is the elite sailfish position (global best position so far). In the original SCA, r_1 is the linearly decreasing parameter ranging from 2 to 0. However, few studies [34,80] suggested using an exponentially decreasing r_1 parameter from 2 to 0 to provide a better balance between the exploration and exploitation phases. The exponentially decreasing strategy is adopted in this study, and a new equation for r_1 is developed as given below:

$$r_1 = a\left(1 - \left(\frac{t}{T}\right)^{\alpha}\right)^{\beta} \tag{14}$$

where t is the current iteration, T is the total number of iterations, and a, α, and β are internal parameters.

3.3.2. The Modified Attack Power Parameter

In the SFO, AP is a linearly decreasing parameter that is based on the parameter A and the convergence factor ϵ (Equation (5)). It is clearly evident in Figure 1 as a straight line. When a sailfish attacks sardines (prey), they immediately disperse in different directions. Therefore, the sailfish should change the search direction to find the optimal solution. However, due to the linear AP parameter, the SFO does not address this random behavior of sardines. It is worth mentioning that AP is the main parameter for switching between exploration and exploitation. Therefore, the parameter needs improvement to align with the behavior of sardines. In the MSSFO, the AP is improved from a linear to a uniform random parameter within the range [0, 1], as shown in Figure 1 with circles. This promotes proper balance between exploration and exploitation and enables faster convergence towards the optimum value.

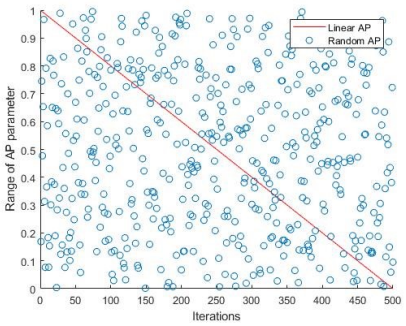


Figure 1. Attack power (AP) in linear and non-linear settings.

3.3.3. The Modified Hunting Process

Towards the last stage of the optimization process in the original SFO, if the injured sardine (the best sardine) is better than any of the sailfish, then the sardine is replaced with that sailfish in the candidate solution (represented by Equation (6)). This way, a sardine is hunted and hence removed from the sardine population. The problem with this technique is that the best sailfish already present in the candidate solution might be removed, and there is a chance that the algorithm gets stuck in a local optimum despite all the hunting process.

To address this limitation, the hunting mechanism has been modified by substituting an injured (best) sardine into the sailfish population if the worst solution in the sailfish population is also poorer than the best sardine. This avoids losing the best solutions in the sailfish population, and candidate solutions will be refined to have better positions in the search space. The modified hunting procedure is implemented using Equation (15):

$$X_{\text{worst_SF}} = X_{\text{injuredS}} \quad \text{if} \quad fitness(X_{\text{injuredS}}) < fitness(X_{\text{worst_SF}}) \tag{15}$$

where $X_{\text{worst_SF}}$ is the worst sailfish in the sailfish population and X_{injuredS} is the best sardine in the sardine population. If X_{injuredS} has a lower cost compared with SF_{worst} (i.e., the injured sardine is better than the worst sailfish), then the injured sardine is replaced with the worst sailfish and removed from the sardine population.

3.3.4. Dynamic Sardine Population

The purpose of having one more population (i.e. the sardine population) in the SFO is to prevent the algorithm from getting stuck in the local optima and to provide better exploration capabilities. It is also important to note that the sailfish position is updated based on the parameter λ, which further depends on prey density. In the original SFO, when the sardine population becomes zero, the global best position of the sailfish becomes fixed at that point, and no further improvement is possible. This leads to premature convergence.

In this study, when all the sardines are either hunted or have run away from the search space, then sailfish will look for more sardines. In other words, when the sardine population becomes empty, more sardines are added to the sardine population. This is referred to as dynamic sardine population in this study and is employed when the sardine population becomes empty. Incorporating all the modifications, the MSSFO is presented in Algorithm 3, along with the flowchart in Figure 2.

3.3.5. Complexity of the Proposed Algorithm

In this section, the computational complexity of the MSSFO is discussed. Here, the number of sailfish is denoted by N, number of sardines is taken as M, and the dimensions are represented by D.

- Initialization of the sailfish and sardine takes $\mathcal{O}(N \cdot D + M \cdot D)$ time.
- To update the position of sailfish, $\mathcal{O}(N \cdot D)$ time is needed.
- To update the position of sardine population, approximately $\mathcal{O}(M \cdot D)$ time is required.

It is important to highlight that the MSSFO retains the same computational complexity as the original SFO. The modification strategies applied to improve the SFO do not alter its overall complexity. Therefore, the computational cost of the algorithm remains unchanged throughout the execution.

Algorithm 3 Proposed Multi-Strategy Sailfish Optimizer.

Initialize the population of sailfish and sardines randomly based on $PP = 0.3$
Set parameters $a = 2$, $\alpha = 0.01$, $\beta = 0.2$
Compute the fitness of sailfish and sardine solutions
Find the best sailfish and injured sardine
while termination conditions are not satisfied **do**
    Calculate PD using Equation (3)
    for each sailfish **do**
        Calculate λ_i using Equation (2)
        Update the position of sailfish using Equation (11)
    end for
    Generate AP, r_2, r_3 randomly within specified ranges
    Generate r_1 using Equation (14)
    if $AP < 0.5$ **then**
        Update the position of sardine using Equation (12)
    else
        Update the position of sardine using Equation (13)
    end if
    Compute the fitness of sailfish and sardines using the objective function
    Update best sailfish, worst sailfish, and injured sardine
    if injured sardine is better than worst sailfish **then**
        Replace worst sailfish using Equation (15)
        Remove hunted sardine from the population
        Update best sailfish and injured sardine
    end if
    if sardine population is empty **then**
        Re-initialize sardine population
        Update injured sardine
    end if
end while
return best sailfish

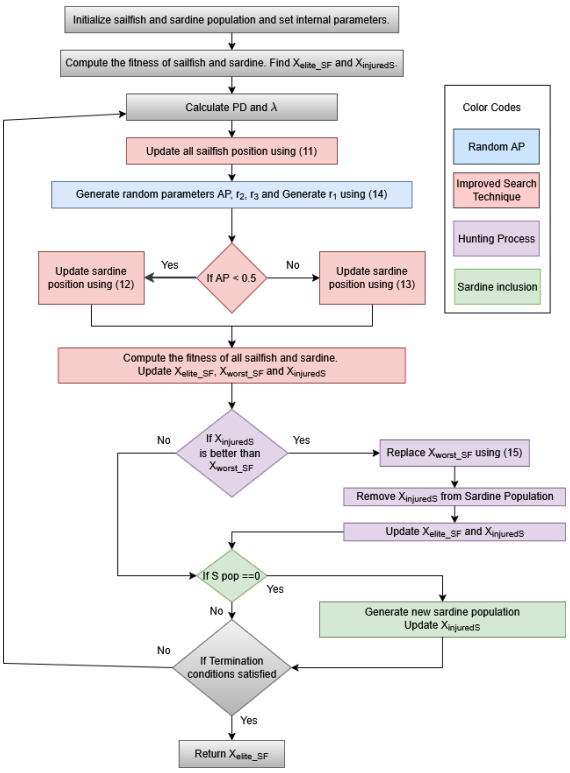


Figure 2. Flowchart of the proposed Multi-Strategy Sailfish Optimizer.

4. Experimental Testing and Evaluation

This section is dedicated to the experimental evaluations and analysis. First, the experimental settings and the benchmark functions are discussed. This is followed by a discussion of results in terms of

- Parameter sensitivity analysis.
- Analysis of the proposed strategy.
- Comparative performance evaluation on benchmark functions.
- Statistical analysis.
- Performance evaluation on two real-world problems.

4.1. Experimental Setup and Benchmark Functions

Simulations are performed to observe and prove the significance of the MSSFO compared with the other metaheuristics employed in this study. All experiments are implemented in MATLAB R2019b with 64-bit Microsoft Windows 10, Intel(R) Core (TM) i5-11400 @ 2.60 Ghz, and 32 GB of RAM. CEC 2022 single-objective bound-constrained (SO-BO) numerical optimization problems [81] are employed to test the performance of the proposed MSSFO. The CEC 2022 SO-BO problems consist of twelve minimization functions divided into four categories: unimodal (F1), multimodal (F2–F5), hybrid (F6–F8), and composite functions (F9–F12). All the functions are executed 30 times on 10 and 20 dimensions with the Maximum Function Evaluations (MaxFEs) set to 200,000 and 1,000,000, respectively, and the error values [F(x) − F(x*)] are obtained for all the test problems. All the experimental settings are kept consistent as per the CEC 2022 SO-BO settings.

4.2. Parameter Sensitivity Analysis

In this section, a parameter sensitivity analysis is performed for parameters α and β along with the exponentially decreasing parameter a. In the MSSFO, these parameters are used to find r_1, which is used to balance the exploration and exploitation processes. Two functions from each category of the CEC 2022 SO-BO are chosen (except the unimodal category because F1 is the only unimodal function) to perform this analysis, namely, F1, F2, F3, F6, F7, F11, and F12. The original source uses $\alpha = 0.03$ and $\beta = 0.2$ with a = 2 without providing a specific reason for the use of these values. Therefore, to verify the validity of the approach and to find the best combinations, experiments are run using different combinations of α and β first, and then the top three combinations with the lowest mean error values [F(x) − F(x*)] are tested with three different values of the parameter a = 2, 3, and 4. Finally, the best combination of the set of values for all the parameters is chosen based on the results obtained.

Parameter α ranges from 0.01 to 0.05, and parameter β ranges from 0.1 to 0.5, making twenty-five combinations with both the parameters. Using these combinations, mean results for 30 independent runs are reported in Tables 2 and 3 along with an individual ranking of each combination. It is interesting to note that the 0.01/0.2 combination is dominant in all the test problems, except F2 and F11. For both F2 and F11, the 0.01/0.2 combination is listed in the top three, making it a good choice for further experimentation. A mean rank based on the Friedman test is also computed and presented in Figure 3, where the superiority of the second combination (0.01/0.2) is clearly visible. Next, the experimentation is conducted to find the best value of parameter a with the top three combinations of α and β.

The top three combinations of α/β are 0.01/0.2, 0.04/0.4, and 0.05/0.3, as evident from Figure 3. The original source uses a value of 2 for the parameter a; hence, the values 2, 3, and 4 are used for this evaluation. The mean results of all the nine combinations are

reported in Table 4, where the best result is highlighted, which verifies the superiority of value 2 with the 0.01/0.2 combination of α/β. Therefore, the final values of the parameters a, α, and β for further experiments are set to 2, 0.01, and 0.2, respectively.

Table 2. Results of F1–F3 for different combinations of α and β.

		F1		F2		F3	
α	β	Avg	Rank	Avg	Rank	Avg	Rank
0.01	0.1	6.6386×10^{-8}	6	1.1699×10^{1}	9	1.4805×10^{1}	25
	0.2	0.0000×10^{0}	1.5	7.3515×10^{0}	3	9.1143×10^{0}	1
	0.3	6.5874×10^{0}	19	2.0029×10^{1}	18	9.1990×10^{0}	3
	0.4	1.0665×10^{4}	24	1.7693×10^{2}	24	9.8464×10^{0}	8
	0.5	2.0577×10^{4}	25	9.7362×10^{2}	25	1.3235×10^{1}	24
0.02	0.1	1.4822×10^{-7}	8	1.4619×10^{1}	12	1.2358×10^{1}	22
	0.2	1.0529×10^{-8}	4	1.5030×10^{1}	15	1.2367×10^{1}	23
	0.3	1.6868×10^{-4}	15	1.9460×10^{1}	16	9.8004×10^{0}	7
	0.4	2.8618×10^{2}	21	3.1422×10^{1}	21	1.0632×10^{1}	11
	0.5	9.3231×10^{3}	23	1.1485×10^{2}	23	1.0796×10^{1}	12
0.03	0.1	1.5133×10^{-7}	9	4.9968×10^{0}	1	1.1919×10^{1}	18
	0.2	6.3978×10^{-8}	5	7.3503×10^{0}	2	1.0989×10^{1}	14
	0.3	2.2637×10^{-7}	10	1.0080×10^{1}	7	9.4864×10^{0}	5
	0.4	9.8952×10^{-3}	17	1.9895×10^{1}	17	1.0553×10^{1}	10
	0.5	2.1275×10^{3}	22	3.8568×10^{1}	22	1.2303×10^{1}	21
0.04	0.1	6.8786×10^{-9}	3	1.0605×10^{1}	8	1.2005×10^{1}	19
	0.2	2.3170×10^{-7}	11	9.7312×10^{0}	6	1.1108×10^{1}	16
	0.3	1.0971×10^{-7}	7	1.4404×10^{1}	11	9.6357×10^{0}	6
	0.4	2.4395×10^{-4}	16	1.4981×10^{1}	14	9.1159×10^{0}	2
	0.5	7.3075×10^{0}	20	2.4589×10^{1}	19	9.2072×10^{0}	4
0.05	0.1	0.0000×10^{0}	1.5	1.4245×10^{1}	10	1.0844×10^{1}	13
	0.2	6.1100×10^{-6}	13	9.4147×10^{0}	5	1.2055×10^{1}	20
	0.3	1.9883×10^{-6}	12	9.1384×10^{0}	4	1.0522×10^{1}	9
	0.4	2.1489×10^{-5}	14	1.4887×10^{1}	13	1.1502×10^{1}	17
	0.5	1.4956×10^{-1}	18	2.7080×10^{1}	20	1.1095×10^{1}	15

Note: Values highlighted in gray represent the best results.

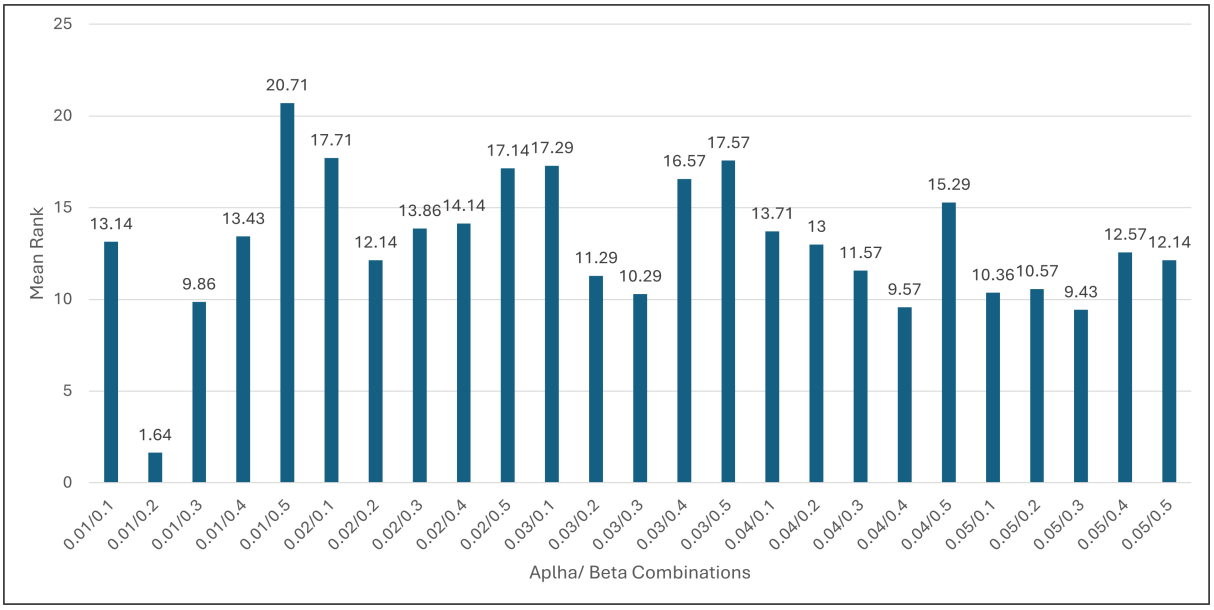


Figure 3. Friedman mean rank for different combinations of parameters alpha and beta.

4.3. Analysis of the Proposed Strategy

In this section, an analysis of the proposed improvement techniques is performed to find the impact of each strategy. Simulations are run using the CEC 2022 SO-BO functions (dim = 10) for this purpose. All the improvement techniques are applied to the SFO one by one to study the impact of each modification individually and collectively, reporting the mean error values along with the ranks presented in Table 5.

Table 3. Results of F6-F7 and F11-F12 for different combinations of α and β.

		F6		F7		F11		F12	
α	β	Avg	Rank	Avg	Rank	Avg	Rank	Avg	Rank
0.01	0.1	1.5653×10^{3}	12	4.0530×10^{1}	14	1.6284×10^{2}	25	1.6482×10^{2}	1
	0.2	1.1219×10^{3}	1	3.5773×10^{1}	1	5.1681×10^{1}	2	1.6487×10^{2}	2
	0.3	1.4805×10^{3}	6	3.9062×10^{1}	9	7.1588×10^{1}	5	1.6537×10^{2}	9
	0.4	1.6068×10^{3}	13	3.6987×10^{1}	2	6.9079×10^{1}	4	1.6569×10^{2}	19
	0.5	1.3525×10^{3}	3	4.6178×10^{1}	25	1.1906×10^{2}	20	1.6592×10^{2}	23
0.02	0.1	1.9647×10^{3}	21	4.1815×10^{1}	17	1.2807×10^{2}	22	1.6577×10^{2}	22
	0.2	1.8852×10^{3}	20	3.8632×10^{1}	7	6.2328×10^{1}	3	1.6552×10^{2}	13
	0.3	1.7815×10^{3}	19	4.2817×10^{1}	19	3.0588×10^{1}	1	1.6574×10^{2}	20
	0.4	1.7097×10^{3}	17	3.7336×10^{1}	3	9.4186×10^{1}	12	1.6557×10^{2}	14
	0.5	2.0522×10^{3}	23	3.7783×10^{1}	5	1.1678×10^{2}	19	1.6557×10^{2}	15
0.03	0.1	2.1016×10^{3}	24	4.2843×10^{1}	20	1.4338×10^{2}	24	1.6714×10^{2}	25
	0.2	1.5529×10^{3}	9	4.3706×10^{1}	22	1.1307×10^{2}	16	1.6543×10^{2}	11
	0.3	1.7032×10^{3}	16	4.3825×10^{1}	23	8.1911×10^{1}	8	1.6500×10^{2}	3
	0.4	1.7657×10^{3}	18	4.4734×10^{1}	24	8.4768×10^{1}	9	1.6574×10^{2}	21
	0.5	1.6508×10^{3}	14	4.0891×10^{1}	15	1.1938×10^{2}	21	1.6534×10^{2}	8
0.04	0.1	1.4633×10^{3}	5	4.3571×10^{1}	21	1.3339×10^{2}	23	1.6565×10^{2}	17
	0.2	1.5340×10^{3}	7	4.2702×10^{1}	18	1.1255×10^{2}	15	1.6567×10^{2}	18
	0.3	2.0100×10^{3}	22	3.9717×10^{1}	12	8.7831×10^{1}	11	1.6544×10^{2}	12
	0.4	1.5610×10^{3}	10	3.8962×10^{1}	8	8.7742×10^{1}	10	1.6526×10^{2}	7
	0.5	1.5622×10^{3}	11	4.1371×10^{1}	16	1.0816×10^{2}	13	1.6607×10^{2}	24
0.05	0.1	1.4348×10^{3}	4	3.9656×10^{1}	11	1.1485×10^{2}	17	1.6562×10^{2}	16
	0.2	1.2086×10^{3}	2	3.7919×10^{1}	6	1.1574×10^{2}	18	1.6540×10^{2}	10
	0.3	2.1855×10^{3}	25	3.7618×10^{1}	4	7.3165×10^{1}	6	1.6525×10^{2}	6
	0.4	1.6923×10^{3}	15	3.9376×10^{1}	10	1.1157×10^{2}	14	1.6519×10^{2}	5
	0.5	1.5442×10^{3}	8	4.0060×10^{1}	13	7.6264×10^{1}	7	1.6513×10^{2}	4

Note: Values highlighted in gray represent the best results.

Table 4. Mean results for top three combinations from Figure 3 with different values of a.

Parameter a	α, β	F1	F2	F3	F6	F7	F11	F12
2	0.01, 0.2	0.0000×10^{0}	7.3515×10^{0}	9.1143×10^{0}	1.1219×10^{3}	3.5773×10^{1}	5.1681×10^{1}	1.6487×10^{2}
3	0.01, 0.2	1.4316×10^{-4}	1.0520×10^{1}	1.4401×10^{1}	1.3445×10^{3}	4.3822×10^{1}	5.1903×10^{1}	1.6628×10^{2}
4	0.01, 0.2	1.5481×10^{-6}	8.5955×10^{0}	1.2741×10^{1}	1.3761×10^{3}	4.8739×10^{1}	1.1726×10^{2}	1.6555×10^{2}
2	0.04, 0.4	2.4395×10^{-4}	1.4981×10^{1}	9.1159×10^{0}	1.4911×10^{3}	3.8962×10^{1}	8.7742×10^{1}	1.6526×10^{2}
3	0.04, 0.4	4.4471×10^{-3}	1.6519×10^{1}	1.2924×10^{1}	2.1332×10^{3}	4.8526×10^{1}	5.3347×10^{1}	1.6674×10^{2}
4	0.04, 0.4	1.0362×10^{-2}	1.3194×10^{1}	1.4388×10^{1}	1.9223×10^{3}	4.1522×10^{1}	1.2788×10^{2}	1.6622×10^{2}
2	0.05, 0.3	1.9883×10^{-6}	9.1384×10^{0}	1.0522×10^{1}	1.5946×10^{3}	3.7618×10^{1}	7.3165×10^{1}	1.6525×10^{2}
3	0.05, 0.3	1.9657×10^{-2}	1.4837×10^{1}	1.1758×10^{1}	2.4062×10^{3}	4.5720×10^{1}	1.1643×10^{2}	1.6631×10^{2}
4	0.05, 0.3	8.7778×10^{-2}	7.7199×10^{0}	1.5210×10^{1}	2.2510×10^{3}	4.8176×10^{1}	1.1237×10^{2}	1.6611×10^{2}

Note: Values highlighted in gray represent the best results.

First, the performance of the original SFO has been evaluated on all the test functions. Thereafter, individual improvements are applied to the SFO, named MSFO-A (Modified Search Strategy), MSFO-B (Random AP Parameter), MSFO-C (Modified Hunting Technique), and MSFO-D (Dynamic Sardine Population). The mean results from the 30 runs

are reported in Table 5 along with the ranks reported within parenthesis. It is evident that the MSFO-A strategy shows better performance in F2, F6, F11, and F12 and the MSFO-B strategy is better in F2, F4, F6, and F12 compared with the SFO. Interestingly, the MSFO-C strategy does not have much impact on the SFO alone, except function F12, for which performance is improved. The MSFO-D strategy has the most significant impact on the performance of the SFO, and the results are evident in 9 out of the 12 problems, i.e., F2 to F5 and F7 to F12.

Table 5. Comparison of the original SFO and its modified variants across 12 test functions. Numbers in parentheses represent the rank.

F	SFO	MSFO-A	MSFO-B	MSFO-C	MSFO-D	MSFO-AC	MSFO-AD	MSSFO
Test Function	**Original**	**Modified Search Strategy**	**Random AP Parameter**	**Modified Hunting Technique**	**Dynamic Sardine Population**	**MSFO-A and MSFO-C Combined**	**MSFO-A and MSFO-D Combined**	**All Techniques Combined**
1	2.6163×10^4 (4)	3.5154×10^4 (5)	3.8178×10^4 (8)	3.6665×10^4 (7)	3.6644×10^4 (6)	2.0103×10^4 (3)	3.6491×10^{-4} (2)	0.0000×10^0 (1)
2	9.9338×10^2 (7)	5.3271×10^2 (4)	9.2094×10^2 (6)	1.1435×10^3 (8)	6.8244×10^2 (5)	2.1423×10^2 (3)	1.5793×10^1 (2)	7.3515×10^0 (1)
3	6.2493×10^1 (5)	7.0819×10^1 (8)	6.4794×10^1 (6)	7.0212×10^1 (7)	5.4497×10^1 (3)	5.5280×10^1 (4)	1.2307×10^1 (2)	9.1143×10^0 (1)
4	8.1641×10^1 (6)	8.6126×10^1 (7)	7.2922×10^1 (4)	9.5074×10^1 (8)	7.7996×10^1 (5)	6.5611×10^1 (3)	3.2569×10^1 (2)	2.0463×10^1 (1)
5	1.2375×10^3 (5)	1.5085×10^3 (7)	1.4391×10^3 (6)	1.7471×10^3 (8)	7.8368×10^2 (4)	7.4164×10^2 (3)	8.8941×10^0 (2)	3.6275×10^{-1} (1)
6	2.3658×10^8 (6)	1.3652×10^8 (4)	1.9521×10^8 (5)	3.0914×10^8 (8)	2.4086×10^8 (7)	5.4424×10^6 (3)	1.5912×10^3 (2)	1.1219×10^3 (1)
7	1.7343×10^2 (5)	1.7665×10^2 (6)	1.7917×10^2 (7)	1.9100×10^2 (8)	1.4060×10^2 (3)	1.4145×10^2 (4)	4.2933×10^1 (2)	3.5773×10^1 (1)
8	1.2375×10^2 (5)	1.3864×10^2 (7)	1.3211×10^2 (6)	1.4663×10^2 (8)	1.2109×10^2 (4)	1.1304×10^2 (3)	2.9527×10^1 (2)	2.5802×10^1 (1)
9	4.7218×10^2 (5)	4.7919×10^2 (6)	4.9568×10^2 (7)	5.0672×10^2 (8)	4.3381×10^2 (3)	4.5286×10^2 (4)	2.3529×10^2 (2)	2.2930×10^2 (1)
10	7.8722×10^2 (5)	1.1059×10^3 (8)	9.6408×10^2 (6)	1.0234×10^3 (7)	6.8686×10^2 (4)	4.3610×10^2 (3)	1.3343×10^2 (2)	1.0545×10^2 (1)
11	9.6630×10^2 (6)	8.9888×10^2 (5)	1.0446×10^3 (8)	1.0230×10^3 (7)	8.4426×10^2 (4)	5.5028×10^2 (3)	1.0522×10^2 (2)	5.1681×10^1 (1)
12	2.5415×10^2 (8)	2.4837×10^2 (5)	2.5253×10^2 (7)	2.5223×10^2 (6)	2.3693×10^2 (3)	2.3738×10^2 (4)	1.6599×10^2 (2)	1.6487×10^2 (1)

The combined effect of the modifications is then analyzed by integrating the MSFO-A strategy with the MSFO-C and MSFO-D strategies separately. Since the MSFO-A strategy already incorporates a randomized AP parameter, it is unnecessary to combine it with the MSFO-B strategy. The two combined variants, the MSFO-AC and the MSFO-AD strategies, are reported in Table 5. To statistically prove effectiveness, a Wilcoxon signed-rank test is also employed for the SFO vs. MSFO-AC and MSFO-AD strategies, giving a p-value less than the significance level ($p < 0.05$), which proves that both the combinations are significantly better than the SFO.

The final version of the improved SFO with all the strategies combined is termed the MSSFO, and the results are presented in the last column of Table 5. The MSSFO achieves the top rank by exhibiting top performance among all variants. To further analyze the significance of the MSSFO, a Wilcoxon signed-rank test is performed at a 95% significance level, which results in a p-value of less than 0.05. This validates that the MSSFO has significantly improved the SFO and can be further evaluated with the other optimization algorithms.

A convergence analysis for selected functions (F1, F3, F7, and F11) is also performed to show the superior convergence capabilities of the MSSFO, presented in Figure 4.

One function from each category is used to visualize the convergence curve, including F1-unimodal, F3-multimodal, F7-Hybrid, and F11-composite functions. In all the problems, the MSSFO exhibits faster convergence and jumps out of the local optima quickly compared with the other variants. A similar trend can be observed in all the other functions.

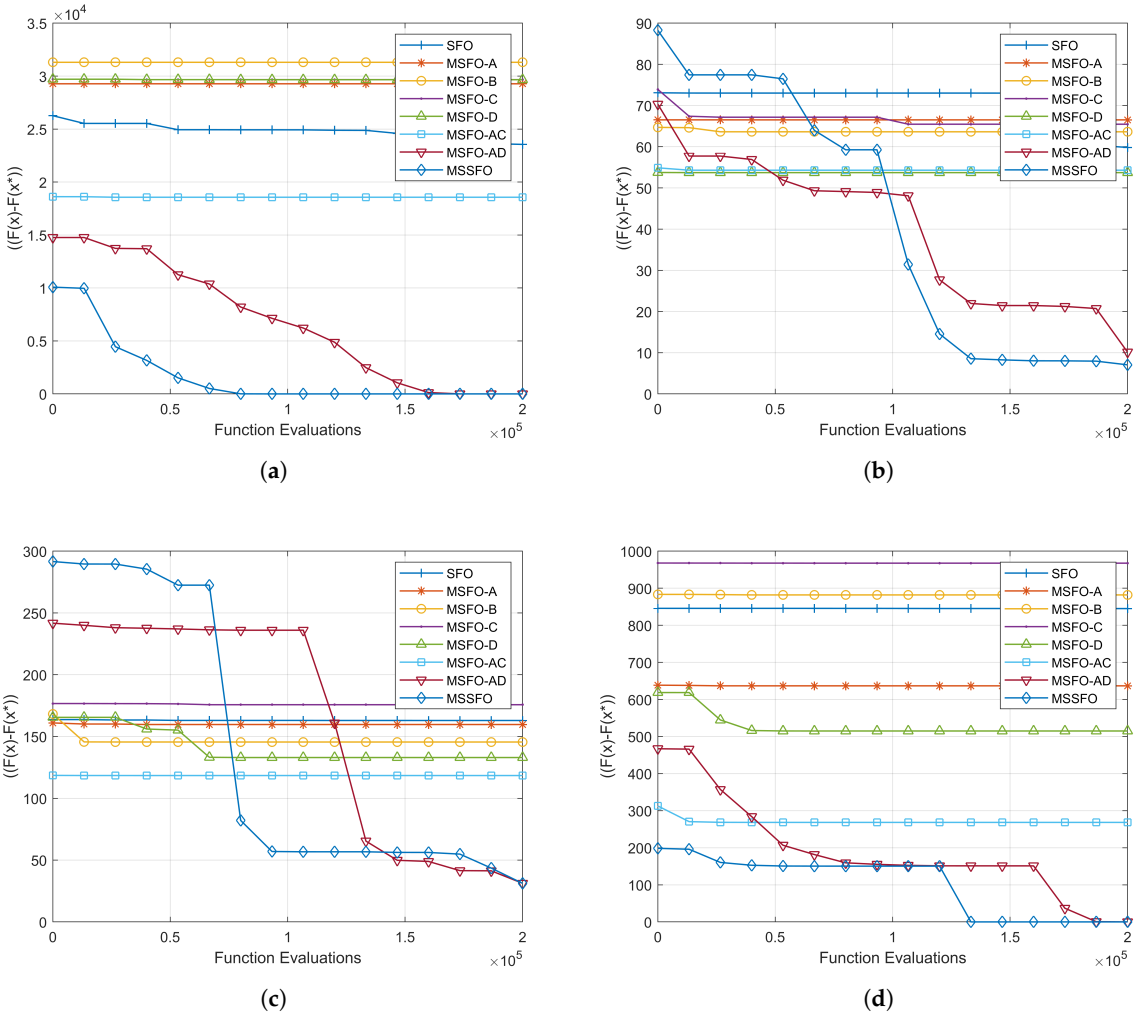


Figure 4. Convergence graphs of proposed strategy for selected CEC 2022 functions: (**a**) F1. (**b**) F3. (**c**) F7. (**d**) F11.

4.4. Comparative Performance Evaluation on Benchmark Functions

In this section, the MSSFO is applied to solve the CEC 2022 SO-BO problems and compared with various well-known optimization algorithms, including the SFO, two other variants of the SFO (MSFO-1 [6] and MSFO-2 [8]), the SCA [23], the ISCA [34], the WOA [24], and the IWOA [38]. The basic and improved techniques for all the selected algorithms are used to give a better performance comparison. The SFO is used because it is the original algorithm on which the MSSFO is based, and the SCA and ISCA are compared because in the MSSFO, the SCA strategy is used to update the sardine position. The WOA is a well-established optimization algorithm; therefore, it is used along with an improved variant for performance comparison. The initial population is set to 30 for all the algorithms, as this size is typically used by researchers in similar studies [24,28]. The rest of the parameter settings of all the compared algorithms are consistent with the original papers and are presented in Table 6.

Table 6. Parameter settings for compared algorithms.

Algorithm	Parameter	Value
SFO	PP	0.3
	Convergence parameter (A)	4
	Convergence factor (ε)	0.0001
WOA	a	Linearly decreases from 2 to 0
SCA	a	2

All the experimental settings for this experimentation follow the guidelines of the CEC 2022 SO-BO. The simulations are performed for 10 and 20 dimensions with 30 runs per problem. The MaxFEs are set to 200,000 and 1,000,000 for the 10- and 20-dimensional problems, respectively. Error values (calculated value minus optimal value) are obtained from each run for all the problems on both the dimensions. The mean, median, standard deviation, and best and worst results are presented in Tables 7 and 8.

It is worth noting that for the 10 dimensional problems, the MSSFO reaches the near-optimal solutions in all the cases, except problem F10, where the SCA and WOA are the winner algorithms. In F6, the IWOA converges faster, and in F12, the IWOA gives the best minimum results. In all the remaining 10-dimensional problems, the MSSFO is the best optimizer. For the 20-dimensional problems, the MSSFO is better than the others in F1, F2, F4, F6, and F8 to F11. In problems F3, F5, and F12, the MSSFO provides the minimum error values for 30 runs. Among the remaining functions, the SCA shows competitive performance in F3, F5, and F7. Furthermore, the WOA and IWOA are also competitive in functions F7 and F9. To highlight the overall performance, a percentage improvement is calculated for all the twelve problems, where the MSSFO shows an overall 55% improvement in the mean results. To further validate the significance of the MSSFO, ranks are calculated and a statistical analysis is performed next.

Table 7. The results of utilized algorithms on CEC 2022 test suites with dim = 10.

	Func.	SFO	MSFO1	MSFO2	SCA	ISCA	WOA	IWOA	MSSFO
Best		3.1112×10^{3}	2.0028×10^{2}	8.1724×10^{3}	1.4113×10^{2}	1.9956×10^{2}	4.4335×10^{2}	2.2715×10^{3}	0.0000×10^{0}
Worst		1.0466×10^{4}	1.1183×10^{4}	8.1323×10^{4}	1.3955×10^{3}	3.9493×10^{3}	1.3986×10^{4}	1.4241×10^{4}	0.0000×10^{0}
Median	1	1.0148×10^{4}	3.9050×10^{3}	3.0528×10^{4}	3.4154×10^{2}	4.0944×10^{2}	4.7598×10^{3}	7.4893×10^{3}	0.0000×10^{0}
Mean		9.4734×10^{3}	4.5180×10^{3}	3.2221×10^{4}	3.9230×10^{2}	5.6366×10^{2}	5.2550×10^{3}	7.5203×10^{3}	0.0000×10^{0}
Std.		1.5816×10^{3}	2.8355×10^{3}	1.7970×10^{4}	2.4691×10^{2}	6.7489×10^{2}	2.8038×10^{3}	3.3782×10^{3}	0.0000×10^{0}
Best		4.5811×10^{1}	1.6249×10^{1}	7.8909×10^{1}	2.1475×10^{1}	1.7492×10^{1}	5.3500×10^{-2}	1.1764×10^{-1}	7.9166×10^{-4}
Worst		7.2268×10^{2}	3.9727×10^{2}	1.5447×10^{3}	6.6718×10^{1}	6.4380×10^{1}	9.2519×10^{1}	9.7591×10^{1}	7.6067×10^{1}
Median	2	2.4825×10^{2}	7.5646×10^{1}	2.8871×10^{2}	4.0092×10^{1}	3.5439×10^{1}	8.9169×10^{0}	8.5089×10^{0}	4.9552×10^{0}
Mean		2.7381×10^{2}	9.3971×10^{1}	4.2051×10^{2}	4.3054×10^{1}	3.9423×10^{1}	2.6935×10^{1}	2.1280×10^{1}	7.3515×10^{0}
Std		1.6788×10^{2}	9.2479×10^{1}	3.5809×10^{2}	1.4520×10^{1}	1.2821×10^{1}	3.3593×10^{1}	2.8409×10^{1}	1.3275×10^{1}
Best		1.7260×10^{1}	6.6521×10^{0}	3.6954×10^{1}	7.8270×10^{0}	1.1126×10^{1}	1.0266×10^{1}	7.7087×10^{0}	1.2031×10^{0}
Worst		9.1782×10^{1}	5.7124×10^{1}	9.4189×10^{1}	2.2915×10^{1}	2.3835×10^{1}	5.8677×10^{1}	4.8530×10^{1}	2.1961×10^{1}
Median	3	5.1341×10^{1}	3.6442×10^{1}	5.8833×10^{1}	1.4922×10^{1}	1.5584×10^{1}	2.5239×10^{1}	2.3219×10^{1}	7.6371×10^{0}
Mean		4.9999×10^{1}	3.1812×10^{1}	6.0828×10^{1}	1.4653×10^{1}	1.6068×10^{1}	2.8396×10^{1}	2.4460×10^{1}	9.1143×10^{0}
Std.		1.7855×10^{1}	1.3294×10^{1}	1.4557×10^{1}	3.1440×10^{0}	2.7829×10^{0}	1.1357×10^{1}	1.0435×10^{1}	5.1938×10^{0}
Best		2.7512×10^{1}	1.3234×10^{1}	3.0532×10^{1}	2.0718×10^{1}	2.2784×10^{1}	1.1941×10^{1}	5.9754×10^{0}	4.9748×10^{0}
Worst		8.7690×10^{1}	6.6593×10^{1}	9.0641×10^{1}	4.7732×10^{1}	4.1864×10^{1}	8.9546×10^{1}	8.9548×10^{1}	3.4823×10^{1}
Median	4	5.4583×10^{1}	2.8026×10^{1}	6.6227×10^{1}	3.1073×10^{1}	3.1685×10^{1}	3.7314×10^{1}	3.6317×10^{1}	2.1392×10^{1}
Mean		5.6290×10^{1}	2.9891×10^{1}	6.7321×10^{1}	3.1539×10^{1}	3.1276×10^{1}	3.8578×10^{1}	3.7613×10^{1}	2.0463×10^{1}
Std.		1.4388×10^{1}	1.0873×10^{1}	1.2876×10^{1}	6.4646×10^{0}	4.4810×10^{0}	1.5247×10^{1}	1.6478×10^{1}	6.6431×10^{0}

Table 7. *Cont.*

	Func.	SFO	MSFO1	MSFO2	SCA	ISCA	WOA	IWOA	MSSFO
Best		1.8725×10^{2}	2.7617×10^{1}	3.7696×10^{2}	1.4395×10^{1}	1.6963×10^{1}	1.0114×10^{2}	5.6183×10^{1}	7.0950×10^{-6}
Worst		1.4093×10^{3}	9.9252×10^{2}	2.4508×10^{3}	1.3907×10^{2}	1.9761×10^{2}	1.4192×10^{3}	1.2206×10^{3}	1.8177×10^{0}
Median	5	6.4946×10^{2}	3.3180×10^{2}	7.6033×10^{2}	4.4918×10^{1}	5.6693×10^{1}	3.7064×10^{2}	3.1161×10^{2}	1.3447×10^{-1}
Mean		7.0498×10^{2}	3.7509×10^{2}	8.9577×10^{2}	5.0827×10^{1}	6.1942×10^{1}	4.3698×10^{2}	3.9662×10^{2}	3.6275×10^{-1}
Std.		3.4210×10^{2}	2.1513×10^{2}	4.5273×10^{2}	3.0218×10^{1}	3.6613×10^{1}	2.9928×10^{2}	2.6007×10^{2}	4.2561×10^{-1}
Best		2.3032×10^{3}	3.5453×10^{2}	2.7537×10^{4}	7.1650×10^{4}	5.1363×10^{4}	1.4430×10^{2}	1.1082×10^{2}	1.0117×10^{2}
Worst		5.3373×10^{7}	6.4910×10^{3}	3.2238×10^{8}	4.3103×10^{6}	2.0638×10^{6}	6.3453×10^{3}	6.2977×10^{3}	3.5740×10^{3}
Median	6	5.5233×10^{5}	1.8812×10^{3}	9.4941×10^{6}	5.1273×10^{5}	5.4777×10^{5}	1.1608×10^{3}	8.5510×10^{2}	6.7390×10^{2}
Mean		6.5601×10^{6}	2.3311×10^{3}	4.4071×10^{7}	8.7380×10^{5}	6.8474×10^{5}	1.8973×10^{3}	1.5660×10^{3}	1.1219×10^{3}
Std.		1.3693×10^{7}	1.9078×10^{3}	8.0660×10^{7}	8.8073×10^{5}	5.0380×10^{5}	1.8918×10^{3}	1.5659×10^{3}	1.0712×10^{3}
Best		6.3844×10^{1}	4.2106×10^{1}	6.6883×10^{1}	3.4702×10^{1}	3.4877×10^{1}	2.5160×10^{1}	2.3967×10^{1}	2.1158×10^{1}
Worst		3.7723×10^{2}	2.3858×10^{2}	3.2251×10^{2}	5.9795×10^{1}	7.0241×10^{1}	8.5812×10^{1}	1.2074×10^{2}	6.6095×10^{1}
Median	7	1.2745×10^{2}	9.2575×10^{1}	1.6535×10^{2}	4.4507×10^{1}	4.5670×10^{1}	5.1682×10^{1}	5.1386×10^{1}	3.2768×10^{1}
Mean		1.4497×10^{2}	9.8453×10^{1}	1.7682×10^{2}	4.5463×10^{1}	4.9019×10^{1}	5.4640×10^{1}	5.5589×10^{1}	3.5773×10^{1}
Std.		6.2784×10^{1}	4.7794×10^{1}	6.5254×10^{1}	5.3851×10^{0}	8.5996×10^{0}	1.6217×10^{1}	2.2751×10^{1}	1.1103×10^{1}
Best		2.8076×10^{1}	2.3269×10^{1}	4.0190×10^{1}	2.3761×10^{1}	2.3128×10^{1}	1.9611×10^{1}	1.1913×10^{1}	3.6426×10^{0}
Worst		2.6165×10^{2}	1.6464×10^{2}	3.7516×10^{2}	3.2769×10^{1}	3.4385×10^{1}	4.3236×10^{1}	4.4081×10^{1}	3.5542×10^{1}
Median	8	7.0587×10^{1}	4.3154×10^{1}	1.2803×10^{2}	2.9680×10^{1}	2.9449×10^{1}	3.0820×10^{1}	3.0541×10^{1}	2.6217×10^{1}
Mean		9.6254×10^{1}	7.8773×10^{1}	1.3496×10^{2}	2.9284×10^{1}	2.9404×10^{1}	3.1491×10^{1}	3.0543×10^{1}	2.5802×10^{1}
Std.		6.3497×10^{1}	5.6069×10^{1}	7.9203×10^{1}	2.1213×10^{0}	2.4960×10^{0}	5.8880×10^{0}	6.2490×10^{0}	6.5687×10^{0}
Best		3.0685×10^{2}	2.3128×10^{2}	3.0678×10^{2}	2.3512×10^{2}	2.3597×10^{2}	2.2929×10^{2}	2.2929×10^{2}	2.2928×10^{2}
Worst		5.0785×10^{2}	4.7505×10^{2}	7.5581×10^{2}	2.7088×10^{2}	2.8941×10^{2}	3.7622×10^{2}	2.8510×10^{2}	2.2933×10^{2}
Median	9	4.4218×10^{2}	3.5607×10^{2}	4.5840×10^{2}	2.4781×10^{2}	2.5453×10^{2}	2.2937×10^{2}	2.2934×10^{2}	2.2929×10^{2}
Mean		4.3408×10^{2}	3.4517×10^{2}	4.9056×10^{2}	2.4856×10^{2}	2.5759×10^{2}	2.3800×10^{2}	2.3748×10^{2}	2.2930×10^{2}
Std.		5.0937×10^{1}	5.9495×10^{1}	1.1079×10^{2}	8.1763×10^{0}	1.7699×10^{1}	2.9119×10^{1}	1.8907×10^{1}	1.5923×10^{-2}
Best		1.0067×10^{2}	1.0056×10^{2}	1.0554×10^{2}	1.0081×10^{2}	1.0080×10^{2}	1.0031×10^{2}	1.0040×10^{2}	1.0047×10^{2}
Worst		1.9875×10^{3}	1.5189×10^{3}	1.9705×10^{3}	1.0536×10^{2}	2.4260×10^{2}	2.4679×10^{2}	2.4729×10^{2}	2.2873×10^{2}
Median	10	2.4902×10^{2}	2.3197×10^{2}	2.7244×10^{2}	1.0136×10^{2}	1.0148×10^{2}	1.0074×10^{2}	1.0083×10^{2}	1.0116×10^{2}
Mean		4.0354×10^{2}	3.2865×10^{2}	5.0434×10^{2}	1.0151×10^{2}	1.3285×10^{2}	1.4338×10^{2}	1.3793×10^{2}	1.0545×10^{2}
Std.		5.0860×10^{2}	4.0581×10^{2}	5.9561×10^{2}	7.8343×10^{-1}	5.7174×10^{1}	6.0623×10^{1}	5.7132×10^{1}	2.2901×10^{1}
Best		2.1954×10^{2}	1.0148×10^{2}	2.0806×10^{2}	1.3208×10^{2}	1.5100×10^{2}	5.1058×10^{-1}	6.9950×10^{-1}	0.0000×10^{0}
Worst		2.1551×10^{3}	1.3039×10^{3}	1.7796×10^{3}	1.7302×10^{2}	5.4487×10^{2}	6.2817×10^{2}	6.2871×10^{2}	4.0000×10^{2}
Median	11	5.2141×10^{2}	2.2664×10^{2}	5.4017×10^{2}	1.6178×10^{2}	1.6315×10^{2}	1.5094×10^{2}	1.5110×10^{2}	0.0000×10^{0}
Mean		6.8545×10^{2}	3.5514×10^{2}	6.3216×10^{2}	1.6057×10^{2}	1.7607×10^{2}	1.9771×10^{2}	1.8582×10^{2}	5.1681×10^{1}
Std.		4.4805×10^{2}	2.7022×10^{2}	3.8154×10^{2}	7.8473×10^{0}	6.8683×10^{1}	1.7882×10^{2}	1.6552×10^{2}	1.2145×10^{2}
Best		1.6850×10^{2}	1.6486×10^{2}	1.6783×10^{2}	1.6557×10^{2}	1.6517×10^{2}	1.6372×10^{2}	1.6140×10^{2}	1.6285×10^{2}
Worst		3.2758×10^{2}	2.7652×10^{2}	3.0532×10^{2}	1.7033×10^{2}	1.9065×10^{2}	2.3720×10^{2}	2.3853×10^{2}	1.6791×10^{2}
Median	12	2.1911×10^{2}	1.7433×10^{2}	2.2802×10^{2}	1.6764×10^{2}	1.6759×10^{2}	1.7224×10^{2}	1.6962×10^{2}	1.6494×10^{2}
Mean		2.2552×10^{2}	1.9148×10^{2}	2.2284×10^{2}	1.6772×10^{2}	1.6815×10^{2}	1.8182×10^{2}	1.7663×10^{2}	1.6487×10^{2}
Std.		4.5750×10^{1}	2.8811×10^{1}	3.7813×10^{1}	1.2837×10^{0}	4.2585×10^{0}	2.1797×10^{1}	1.9711×10^{1}	1.0370×10^{0}

Note: Values highlighted in gray represent the best results.

Table 8. The results of different algorithms on CEC 2022 test suite with dim = 20.

	Func.	SFO	MSFO1	MSFO2	SCA	ISCA	WOA	IWOA	MSSFO
Best		3.7293×10^{4}	9.8810×10^{3}	5.2651×10^{4}	2.1041×10^{3}	2.1712×10^{3}	1.6633×10^{0}	1.8143×10^{3}	3.0932×10^{-3}
Worst		9.4503×10^{4}	7.5677×10^{4}	1.8301×10^{5}	9.7667×10^{3}	8.0624×10^{3}	7.2329×10^{1}	3.6010×10^{4}	1.3539×10^{-2}
Median	1	6.1895×10^{4}	2.7820×10^{4}	9.0587×10^{4}	4.7091×10^{3}	5.0153×10^{3}	1.1429×10^{1}	8.7585×10^{3}	6.1565×10^{-3}
Mean		6.2829×10^{4}	3.1312×10^{4}	1.0184×10^{5}	5.3356×10^{3}	5.2665×10^{3}	2.1699×10^{1}	9.3791×10^{3}	6.7114×10^{-3}
Std.		1.3326×10^{4}	1.3139×10^{4}	3.3626×10^{4}	2.1205×10^{3}	1.4064×10^{3}	1.9790×10^{1}	6.6195×10^{3}	2.8183×10^{-3}

Table 8. *Cont.*

	Func.	SFO	MSFO1	MSFO2	SCA	ISCA	WOA	IWOA	MSSFO
Best		5.0222×10^{2}	2.6535×10^{2}	5.1638×10^{2}	1.3096×10^{2}	1.2702×10^{2}	5.2450×10^{0}	7.0481×10^{0}	4.1375×10^{0}
Worst		2.6767×10^{3}	8.5071×10^{2}	3.6395×10^{3}	3.1774×10^{2}	3.0106×10^{2}	1.3216×10^{2}	1.7209×10^{2}	7.3867×10^{1}
Median	2	1.2529×10^{3}	4.7642×10^{2}	1.7853×10^{3}	1.8110×10^{2}	1.8647×10^{2}	4.9203×10^{1}	6.5788×10^{1}	4.9090×10^{1}
Mean		1.3730×10^{3}	4.7838×10^{2}	1.9244×10^{3}	1.9012×10^{2}	1.9249×10^{2}	5.8233×10^{1}	8.3150×10^{1}	4.8277×10^{1}
Std.		5.5179×10^{2}	1.3741×10^{2}	7.2439×10^{2}	4.2703×10^{1}	3.2660×10^{1}	2.0208×10^{1}	4.5183×10^{1}	1.6892×10^{1}
Best		5.4917×10^{1}	3.3641×10^{1}	5.2358×10^{1}	2.2928×10^{1}	2.7114×10^{1}	2.5098×10^{1}	3.5982×10^{1}	1.8164×10^{1}
Worst		1.0601×10^{2}	9.3314×10^{1}	1.3581×10^{2}	3.8717×10^{1}	4.4901×10^{1}	8.0737×10^{1}	7.7349×10^{1}	6.7681×10^{1}
Median	3	7.8495×10^{1}	5.7476×10^{1}	9.3921×10^{1}	3.0721×10^{1}	3.1037×10^{1}	5.1420×10^{1}	5.3473×10^{1}	4.8746×10^{1}
Mean		7.9970×10^{1}	5.9389×10^{1}	9.3991×10^{1}	3.0864×10^{1}	3.2683×10^{1}	5.4247×10^{1}	5.4869×10^{1}	4.8712×10^{1}
Std.		1.3194×10^{1}	1.5066×10^{1}	1.7054×10^{1}	3.8135×10^{0}	4.3897×10^{0}	1.0349×10^{1}	9.5353×10^{0}	1.2209×10^{1}
Best		1.2502×10^{2}	8.4397×10^{1}	1.4900×10^{2}	9.6359×10^{1}	9.6206×10^{1}	5.0743×10^{1}	6.6665×10^{1}	4.8753×10^{1}
Worst		2.2570×10^{2}	1.5740×10^{2}	2.6506×10^{2}	1.3240×10^{2}	1.3374×10^{2}	1.9402×10^{2}	1.9402×10^{2}	1.3332×10^{2}
Median	4	1.7970×10^{2}	1.2462×10^{2}	1.9609×10^{2}	1.1994×10^{2}	1.1424×10^{2}	1.0795×10^{2}	1.0049×10^{2}	7.6612×10^{1}
Mean		1.8200×10^{2}	1.2203×10^{2}	1.9325×10^{2}	1.1889×10^{2}	1.1285×10^{2}	1.1240×10^{2}	1.1092×10^{2}	7.7607×10^{1}
Std.		2.5016×10^{1}	2.0511×10^{1}	2.5933×10^{1}	8.4399×10^{0}	9.4536×10^{0}	3.3278×10^{1}	3.5591×10^{1}	1.8310×10^{1}
Best		2.1957×10^{3}	1.0196×10^{3}	2.6940×10^{3}	4.2817×10^{2}	4.1414×10^{2}	7.6928×10^{2}	5.7352×10^{2}	3.9719×10^{2}
Worst		5.1042×10^{3}	3.1661×10^{3}	9.0338×10^{3}	1.1778×10^{3}	1.3208×10^{3}	6.0255×10^{3}	4.8438×10^{3}	2.0443×10^{3}
Median	5	3.2329×10^{3}	2.1182×10^{3}	3.9763×10^{3}	6.6459×10^{2}	6.7760×10^{2}	1.7427×10^{3}	1.5859×10^{3}	1.2821×10^{3}
Mean		3.2890×10^{3}	2.1534×10^{3}	4.1722×10^{3}	7.0466×10^{2}	7.4552×10^{2}	2.1463×10^{3}	1.8227×10^{3}	1.2696×10^{3}
Std.		7.1094×10^{2}	4.5235×10^{2}	1.2238×10^{3}	1.9172×10^{2}	2.2563×10^{2}	1.1883×10^{3}	9.2280×10^{2}	3.5878×10^{2}
Best		2.2958×10^{7}	2.0585×10^{5}	1.0181×10^{8}	3.8309×10^{6}	3.1969×10^{6}	1.8848×10^{2}	1.6012×10^{3}	1.5133×10^{2}
Worst		2.3583×10^{9}	3.7976×10^{8}	2.5228×10^{9}	1.0917×10^{8}	8.6440×10^{7}	1.8672×10^{4}	2.8832×10^{7}	1.8597×10^{4}
Median	6	5.5534×10^{8}	1.5998×10^{7}	7.2025×10^{8}	4.8540×10^{7}	1.8407×10^{7}	3.3104×10^{3}	1.8445×10^{5}	2.1276×10^{3}
Mean		7.4108×10^{8}	3.0074×10^{7}	8.7831×10^{8}	4.6179×10^{7}	2.5504×10^{7}	5.2912×10^{3}	3.1671×10^{6}	4.1926×10^{3}
Std.		6.1230×10^{8}	6.6807×10^{7}	6.2903×10^{8}	2.7474×10^{7}	2.1917×10^{7}	5.3669×10^{3}	7.1001×10^{6}	4.4813×10^{3}
Best		1.5841×10^{2}	7.5627×10^{1}	1.8871×10^{2}	6.7648×10^{1}	7.1328×10^{1}	7.9205×10^{1}	4.6775×10^{1}	7.2849×10^{1}
Worst		4.7869×10^{2}	4.2448×10^{2}	6.4841×10^{2}	1.3355×10^{2}	1.3126×10^{2}	3.1242×10^{2}	2.5819×10^{2}	2.1709×10^{2}
Median	7	2.8310×10^{2}	2.0606×10^{2}	3.6929×10^{2}	9.4172×10^{1}	1.0635×10^{2}	1.5842×10^{2}	1.4619×10^{2}	1.2619×10^{2}
Mean		2.8496×10^{2}	2.2408×10^{2}	3.6688×10^{2}	9.3684×10^{1}	1.0439×10^{2}	1.5533×10^{2}	1.4230×10^{2}	1.2731×10^{2}
Std.		8.2863×10^{1}	8.8401×10^{1}	1.0009×10^{2}	1.3277×10^{1}	1.3934×10^{1}	5.1887×10^{1}	5.0619×10^{1}	3.6106×10^{1}
Best		7.2435×10^{1}	3.0949×10^{1}	8.1869×10^{1}	3.4470×10^{1}	3.3475×10^{1}	2.6779×10^{1}	2.9448×10^{1}	2.6091×10^{1}
Worst		7.4573×10^{2}	4.6507×10^{2}	1.3551×10^{3}	5.2388×10^{1}	7.1399×10^{1}	6.3284×10^{1}	1.5219×10^{2}	1.7276×10^{2}
Median	8	3.4776×10^{2}	1.7957×10^{2}	4.0887×10^{2}	4.2174×10^{1}	4.1344×10^{1}	3.8834×10^{1}	3.7193×10^{1}	3.6562×10^{1}
Mean		3.9646×10^{2}	2.0627×10^{2}	4.6782×10^{2}	4.2086×10^{1}	4.2311×10^{1}	4.3405×10^{1}	4.6841×10^{1}	5.9428×10^{1}
Std.		1.8740×10^{2}	1.2637×10^{2}	2.4582×10^{2}	4.2581×10^{0}	6.6141×10^{0}	1.1416×10^{1}	2.9014×10^{1}	4.9121×10^{1}
Best		4.0045×10^{2}	2.4556×10^{2}	5.2900×10^{2}	2.0717×10^{2}	2.0084×10^{2}	1.8079×10^{2}	1.8079×10^{2}	1.8081×10^{2}
Worst		1.2263×10^{3}	8.1446×10^{2}	1.4927×10^{3}	2.5769×10^{2}	2.6818×10^{2}	1.8285×10^{2}	1.8536×10^{2}	1.8096×10^{2}
Median	9	6.9719×10^{2}	4.5081×10^{2}	8.5790×10^{2}	2.2651×10^{2}	2.2649×10^{2}	1.8091×10^{2}	1.8114×10^{2}	1.8092×10^{2}
Mean		7.4632×10^{2}	4.7206×10^{2}	8.6874×10^{2}	2.2801×10^{2}	2.2908×10^{2}	1.8126×10^{2}	1.8142×10^{2}	1.8091×10^{2}
Std.		2.0319×10^{2}	1.3051×10^{2}	2.3719×10^{2}	1.2993×10^{1}	1.6297×10^{1}	5.4487×10^{-1}	9.0127×10^{-1}	4.9595×10^{-2}
Best		1.8624×10^{2}	1.1989×10^{2}	3.8011×10^{3}	1.0905×10^{2}	1.0609×10^{2}	1.0073×10^{2}	1.0067×10^{2}	1.0059×10^{2}
Worst		5.6310×10^{3}	4.9434×10^{3}	5.9809×10^{3}	1.2091×10^{2}	3.7133×10^{2}	3.4504×10^{3}	3.3960×10^{3}	2.9553×10^{3}
Median	10	4.6370×10^{3}	3.9372×10^{3}	5.1800×10^{3}	1.1467×10^{2}	1.1723×10^{2}	1.8638×10^{3}	1.6848×10^{3}	1.0176×10^{2}
Mean		4.5244×10^{3}	3.6328×10^{3}	5.1546×10^{3}	1.1441×10^{2}	1.5586×10^{2}	1.5138×10^{3}	1.6373×10^{3}	3.5898×10^{2}
Std.		9.3340×10^{2}	1.0860×10^{3}	5.1119×10^{2}	3.4235×10^{0}	8.9533×10^{1}	1.0543×10^{3}	9.6843×10^{2}	7.0559×10^{2}
Best		2.7021×10^{3}	1.2308×10^{3}	3.2246×10^{3}	8.6184×10^{2}	9.5055×10^{2}	7.4391×10^{-1}	6.4822×10^{0}	3.0000×10^{2}
Worst		6.1848×10^{3}	4.3307×10^{3}	7.4610×10^{3}	2.0828×10^{3}	2.3211×10^{3}	5.0784×10^{3}	6.9199×10^{2}	7.5338×10^{2}
Median	11	5.1510×10^{3}	2.4268×10^{3}	5.7144×10^{3}	1.5746×10^{3}	1.7457×10^{3}	3.0234×10^{2}	3.2175×10^{2}	3.0000×10^{2}
Mean		4.9145×10^{3}	2.4752×10^{3}	5.5683×10^{3}	1.5269×10^{3}	1.7126×10^{3}	6.0039×10^{2}	3.7004×10^{2}	3.4511×10^{2}
Std.		9.2900×10^{2}	6.6742×10^{2}	1.0153×10^{3}	3.5644×10^{2}	3.4058×10^{2}	1.0648×10^{3}	1.2253×10^{2}	8.8412×10^{1}

Table 8. *Cont.*

	Func.	SFO	MSFO1	MSFO2	SCA	ISCA	WOA	IWOA	MSSFO
Best		3.9376×10^2	2.8920×10^2	3.4685×10^2	2.7456×10^2	2.7246×10^2	2.5385×10^2	2.4973×10^2	2.4511×10^2
Worst		1.1621×10^3	6.3435×10^2	9.7937×10^2	3.2521×10^2	3.8057×10^2	3.9206×10^2	3.6712×10^2	5.1735×10^2
Median	12	6.0362×10^2	4.2483×10^2	6.4637×10^2	2.9461×10^2	3.0160×10^2	2.8710×10^2	2.9580×10^2	3.2777×10^2
Mean		6.1795×10^2	4.2866×10^2	6.3861×10^2	2.9470×10^2	3.0546×10^2	2.9621×10^2	2.9773×10^2	3.4027×10^2
Std.		1.6695×10^2	7.7817×10^1	1.6645×10^2	1.0803×10^1	1.8911×10^1	3.5517×10^1	2.8193×10^1	7.1874×10^1

Note: Values highlighted in gray represent the best results.

4.5. Statistical Analysis

Studies have shown the effectiveness and validity of various statistical techniques on a set of mean values obtained by running different metaheuristics on similar problems, and recommendations about the test to use in specific situations are also available for researchers [82]. For this study, the Friedman test (multiple comparison), followed by the Wilcoxon signed-rank test (one-to-one comparison), is used. In this section, first, the ranking is performed based on the mean error values and then average and total ranks are calculated and presented in Tables 9 and 10. It is observed that the MSSFO outperforms the other algorithms in eleven functions out of twelve for 10 dimensional problems, hence achieving an overall rank of one. For the 20 dimensional problems, the SCA and ISCA are the competitors; however, the overall rank of the MSSFO is still first in these problems.

The Friedman test is further used to statistically analyze the ranking (especially when the number of comparative algorithms are equal to or greater than six [82]). If the results of the test are significant, then it means that there exists at least one algorithm that is significantly different. The Friedman statistics F can be computed in more than one way, as presented in the literature. In all the cases, the computed statistical value (computed F) should be greater than the tabled critical value to prove that at least one algorithm is statistically better than the others.

Table 9. Average and total ranks of utilized algorithms on 10 dimensional problems.

Function	MSSFO	SFO	MSFO1	MSFO2	SCA	ISCA	WOA	IWOA
1	1	7	4	8	2	3	5	6
2	1	7	6	8	5	4	3	2
3	1	7	6	8	2	3	5	4
4	1	7	2	8	4	3	6	5
5	1	7	4	8	2	3	6	5
6	1	7	4	8	6	5	3	2
7	1	7	6	8	2	3	4	5
8	1	7	6	8	2	3	5	4
9	1	7	6	8	4	5	3	2
10	2	7	6	8	1	3	5	4
11	1	8	6	7	2	3	5	4
12	1	8	6	7	2	3	5	4
Avg Rank	1.08	7.17	5.17	7.83	2.83	3.42	4.58	3.92
Total Rank	1	7	6	8	2	3	5	4

Table 10. Average and total ranks of different algorithms on 20 dimensional problems.

Function	MSSFO	SFO	MSFO1	MSFO2	SCA	ISCA	WOA	IWOA
1	1	7	6	8	4	3	2	5
2	1	7	6	8	4	5	2	3
3	3	7	6	8	1	2	4	5
4	1	7	6	8	5	4	3	2
5	3	7	6	8	1	2	5	4
6	1	7	5	8	6	4	2	3
7	3	7	6	8	1	2	5	4
8	5	7	6	8	1	2	3	4
9	1	7	6	8	4	5	2	3
10	3	7	6	8	1	2	4	5
11	1	7	6	8	4	5	3	2
12	5	7	6	8	1	4	2	3
Avg Rank	2.33	7	5.92	8	2.75	3.33	3.08	3.58
Total Rank	1	7	6	8	2	4	3	5

The basic Friedman test uses chi-square distribution to approximate the F statistic. The computed F value is then compared to the tabled chi-square distribution [83]. The variation of this statistical test proposed by Iman and Davenport [84] is also employed in this study. The tabled critical values and the computed values with a 95% significance level (α = 0.05) are presented in Table 11. It is observed from the table that the computed F value is greater than the tabled critical value in both cases, which means that at least one algorithm is performing better than the others. The computed *p*-value is also lower than the significance level.

Table 11. Friedman test results on CEC 2022 functions.

Dim.	Test Method	Tabled Critical Value at $\alpha = 0.05$	Computed Value (F)	*p*-Value
10D	Chi-square distribution	14.06714045	69.27778	2.06729×10^{-12}
	Iman and Davenport method	2.130990456	51.76226	1.47587×10^{-26}
20D	Chi-square distribution	14.06714045	64.9444	1.5423×10^{-11}
	Iman and Davenport method	2.130990456	37.4898	2.6002×10^{-22}

Since the Friedman is a multiple comparison test, the rejection of the null hypothesis (null hypothesis here = all algorithms perform equally) leads to the one-to-one comparison [85] that will highlight the algorithm which performs better than the others. The Wilcoxon signed-rank test is used to compare the MSSFO with all the other algorithms to verify the significance of the study. The Wilcoxon signed-rank test is employed on the mean error values, and the results are presented in Table 12. It is evident from the table that the MSSFO is significantly better than all the other algorithms with a *p*-value lower than the threshold (0.05) for 10 dimensional problems. In the 20 dimensional problems, the MSSFO is competitive with respect to the SCA and the ISCA and shows superior performance than the SFO, MSFO-1, MSFO-2, the WOA, and the IWOA, with a *p*-value < 0.05.

Table 12. Wilcoxon signed-rank test results of MSSFO versus other algorithms.

MSSFO vs.	z-Statistic	*p*-Value	MSSFO vs.	z-Statistic	*p*-Value
SFO	3.05941	0.0022			
SCA	2.74563	0.0060	ISCA	3.05941	0.0022
WOA	2.74563	0.0060	IWOA	2.82407	0.0047
MSFO-1	3.05941	0.0022	MSFO-2	3.05941	0.0022

The convergence graphs for the selected CEC 2022 problems are presented in Figure 5. In the unimodal problem (F1) and multimodal problem (F3), the MSSFO is the fastest and converges very quickly to the optimum/near-optimum solutions. F8 is the hybrid problem, and F11 is composite in nature, and in both the cases, the MSSFO exhibits the best convergence performance. A similar trend is observed in other test problems. Hence, this analysis confirms that the MSSFO exhibits faster convergence and does not stuck in the local optimum. Statistical analysis proves that the MSSFO is not only better than the SFO but also superior to the existing variants of the SFO and other optimization algorithms. Next, the MSSFO is applied to solve real-world problems.

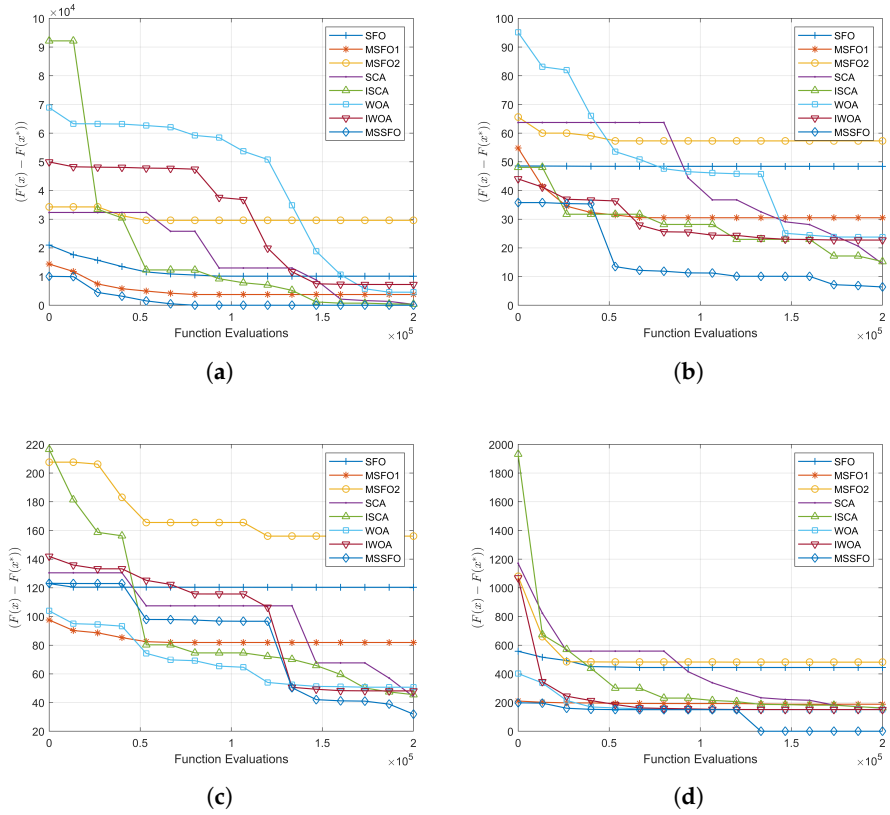


Figure 5. Comparative convergence graphs for selected CEC functions: (**a**) F1. (**b**) F3. (**c**) F7. (**d**) F11.

4.6. Performance Evaluation on Compression Spring Design Problem

The tension/compression spring design (CSD) problem is a continuous constrained engineering design problem [24]. The objectives of this problem are to minimize the weight of the coil spring subjected to the constraints of shear stress, surge frequency, and minimum deflection. Decision variables include wire diameter (d), average coil diameter (D), effective coil number (P), and f(x) to minimize the spring weight. The objective functions and constraints of these three optimization variables are as follows [86]:

Minimize

$$f(\mathbf{x}) = x_1^2 \cdot x_2 \cdot (2 + x_3)$$
$$\mathbf{x} = [x_1\ x_2\ x_3] = [d\ D\ P]$$

subject to

$$g_1(x) = 1 - \frac{x_2^3 x_3}{71{,}785\, x_1^4} \leq 0$$
$$g_2(x) = \frac{4x_2^2 - x_1 x_2}{12{,}566(x_2 x_1^3 - x_1^4)} + \frac{1}{5108\, x_1^2} \leq 0$$
$$g_3(x) = 1 - \frac{140.45\, x_1}{x_2^2 x_3} \leq 0$$
$$g_4(x) = \frac{x_1 + x_2}{1.5} - 1 \leq 0$$

with

$$0.05 \leq x_1 \leq 2.0, \quad 0.25 \leq x_2 \leq 1.3, \quad 2.0 \leq x_3 \leq 15.0$$

The optimization results of the CSD problem for the MSSFO are compared with the GA, PSO, weighted chimp optimization (WCHO), the SFO, the SSA, and the IWOA, and the results are presented in Table 13. The GA, PSO, and WCHO results are taken from the literature [86], and the rest of the results have been generated. Each algorithm is independently run 30 times, and the maximum number of iterations is set to 1000. These settings are consistent with the literature [86]. From Table 13, it is evident that the performance of the MSSFO is better than all other nature-inspired optimization algorithms, as it provides the best solutions among all. Further analysis using the convergence curve of the median values is presented in Figure 6.

It can be observed from Table 13 and Figures 6 and 7 that the MSSFO shows better performance compared with all the other algorithms applied in the CSD problem and improves the quality of solutions for the selected problem.

Table 13. Comparison of the MSSFO results with other algorithms for tension/compression spring design problem.

Algorithm	d	D	P	Mean Fitness
GA [86]	0.0524	0.3521	11.5980	0.032
PSO [86]	0.0500	0.3104	14.998	0.0131
WCHO [86]	0.0508	0.3142	14.4314	0.0134
MSSFO	0.0556	0.4594	7.5262	0.0130
SFO	0.0572	0.5124	6.7111	0.0135
SSA	0.0530	0.3988	11.2036	2.2940
IWOA	0.0584	0.5483	5.8836	0.0137

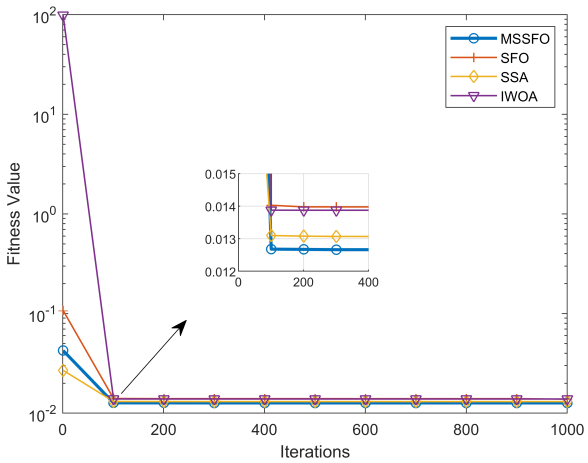


Figure 6. Convergence curve of CSD problem using median values.

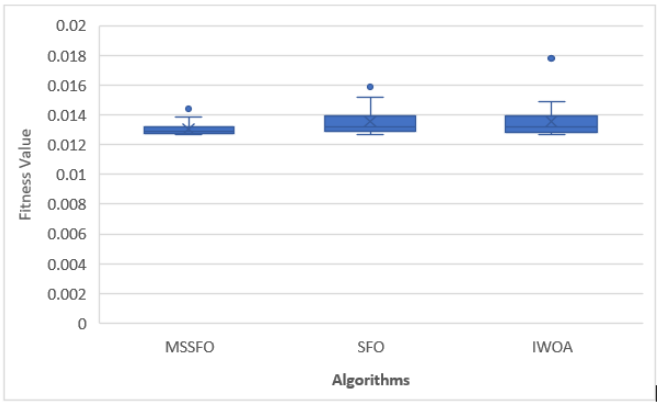


Figure 7. Optimal value box plot for CSD problem.

4.7. Performance Evaluation on Mobile Robot Path Planning Problem

Path planning is a fundamental aspect of mobile robot technology, with the primary goal of determining a safe and optimal route from a given starting point to the target destination [87]. The objective here is to minimize the path length in the presence of various obstacles. Given a solution that represents a path consisting of a series of nodes in a graph, the fitness value can be computed as the total sum of distances between consecutive nodes, from the start node to the destination node, presented by the following equation:

$$\text{Fitness}(F) = \sum_{i=1}^{N-1} d(P_i, P_{i+1})$$

The Euclidean distance between two consecutive points $P_i(x_i, y_i)$ and $P_{i+1}(x_{i+1}, y_{i+1})$ is given by

$$d(P_i, P_{i+1}) = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}.$$

Two scenarios are considered for this problem, where scenario one uses one circular obstacle and scenario two uses multiple regularly shaped obstacles in a 100 × 100 2D environment. Multiple path points are setup for the simulation starting from (5, 5) and ending at (99, 99). The start and end points are determined and kept constant before running the simulation. The path points are randomly generated within the boundary of the environment. The initial population is set to 20, and the number of runs is 30, whereas the number of iterations for each run is set to 200. For a fair comparison, these settings are kept consistent for all the algorithms. Other algorithms used for comparison are the PSO, the SFO, and the WOA. The optimal results for the two scenarios (after 30 independent runs) are reported in Tables 14 and 15. Convergence behavior for all the algorithms utilized in scenario 1 and scenario 2 are visualized in Figure 8 and Figure 9, respectively. Optimal paths selected by various algorithms using two scenarios are presented in Figures 10 and 11.

Table 14. Comparison of the MSSFO with other algorithms on the mobile robot path planning problem using one circular obstacle.

Algorithm	Best Path Length	Average Path Length	Std. Dev.
MSSFO	139.5648	176.0337	39.8151
WOA	139.5343	1381.7761	6642.2654
PSO	183.7413	371.2224	118.4736
SFO	155.5843	1955.6175	6763.5753

Table 15. Comparison of the MSSFO with other algorithms on the mobile robot path planning problem using multiple obstacles.

Algorithm	Best Path Length	Average Path Length	Std. Dev.
MSSFO	136.1217	179.8756	74.6789
WOA	138.2446	933,488.3621	1,337,344.7480
PSO	164.2967	233,660.4280	430,180.6187
SFO	140.1128	733,551.9436	1,014,830.7897

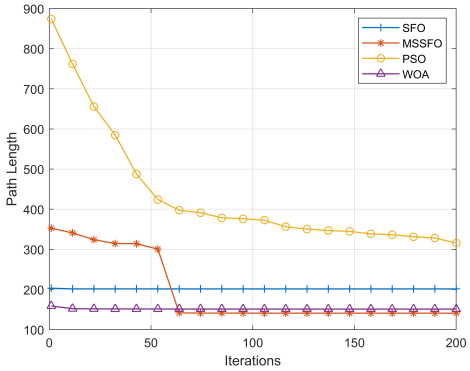


Figure 8. Convergence curve of robot path planning with one circular obstacle.

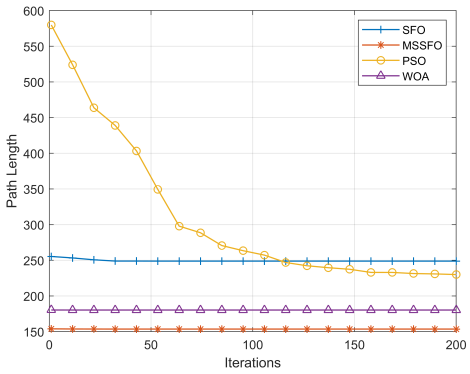
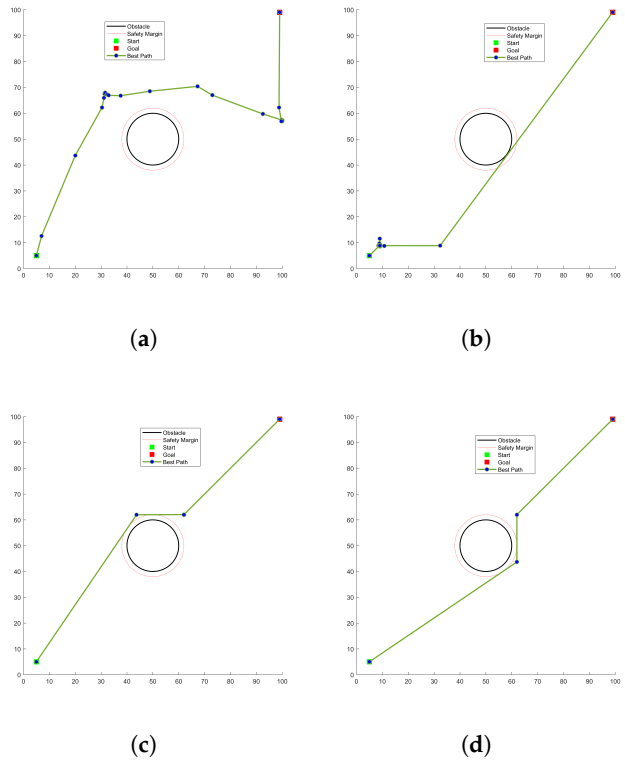


Figure 9. Convergence curve of robot path planning with multiple regularly shaped obstacles.

It is observed from Table 14 that the MSSFO and WOA perform similarly in finding a minimum (best) distance; however, there is a clear difference in average performance, which confirms the robustness of the proposed MSSFO. As a result, the MSSFO exhibits the best average path length with a minimal deviation. This is further verified when the MSSFO is applied to the second scenario (with multiple obstacles), where the MSSFO performs better than all the other algorithms, with the minimum best path length, the minimum average path length, and a minimal deviation from the results.

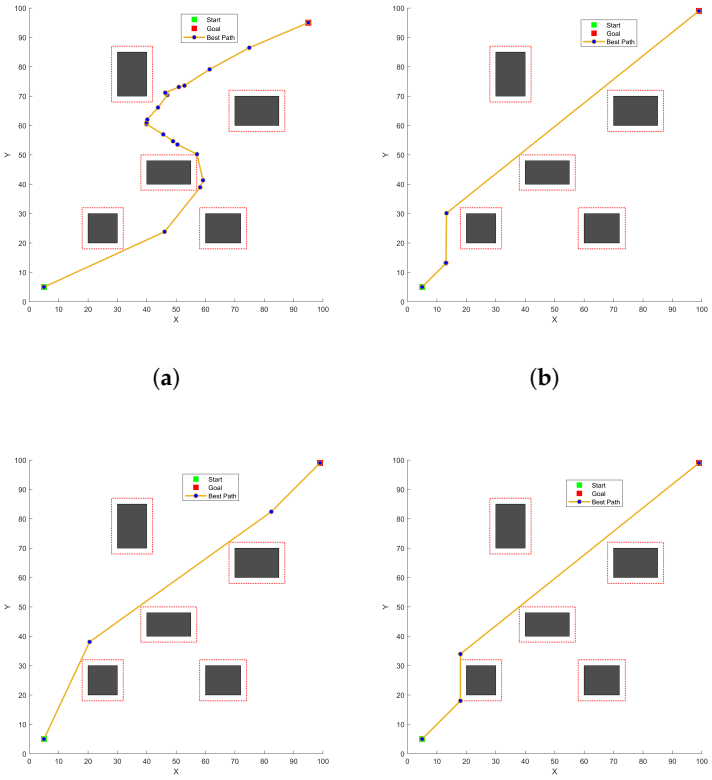
The superiority of the MSSFO can be further verified by the convergence curve presented in Figures 8 and 9. For the single-obstacle scenario, it can be seen that the WOA and the MSSFO are competitive, whereas the MSSFO outperforms all the other algorithms when the multiple-obstacle environment is created. This verifies the effectiveness and efficiency of the MSSFO in solving the mobile robot path planning problem, which can be further explored in complex environments with multiple irregularly shaped obstacles, more path points, or multiple robots.



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

(**c**) (**d**)

Figure 10. Optimal path for single obstacle using (**a**) PSO, (**b**) SFO, (**c**) MSSFO, and (**d**) WOA.



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

(**c**) (**d**)

Figure 11. Optimal path for multiple-obstacle scenario from start to end using (**a**) PSO, (**b**) SFO, (**c**) MSSFO, and (**d**) WOA. (Obstacles are black objects, and red lines around obstacles are safety margins.)

5. Discussion

The sailfish optimizer (SFO) has gained considerable attention since its introduction, with numerous improvements and applications proposed in the literature. In this study, a new enhanced variant of the SFO, referred to as the Multi-Strategy Sailfish Optimizer (MSSFO), has been developed by integrating multiple strategies, including an improved search mechanism, a non-linear attack power (AP) parameter, a modified hunting process, and a novel dynamic sardine population technique. The experimental results have been systematically presented in the previous section.

Parameter sensitivity analysis is a crucial step in laying down the basis for further experimentation. Three internal parameters, α, β, and a were experimentally evaluated with different combinations to find suitable values for further experimentation. The mean results and a Friedman mean rank were used to select the best combinations, and the final values of parameter a, α, and β turned out to be 2, 0.01, and 0.2, respectively.

Next, the impact of each modification individually and in combination was studied and applied to the original SFO. Twelve minimization problems from the CEC 2022 test suite were utilized for the experimentation purposes. F1 is the only unimodal problem, and interestingly, none of the techniques individually impacted the solution significantly. However, when the modified search strategy was combined with a dynamic sardine population (MSFO-AD), the result for F1 was significantly improved. Furthermore, when all the techniques were combined together, the algorithm successfully found the optimal solution, as the error value reached zero for F1. In the multimodal problems (F2 to F5), the dynamic sardine population (MSFO-D) strategy improved the original SFO and achieved better performance than the SFO in all the problems. The hybrid functions (F6 to F8) seemed more challenging, and a very slight improvement was observed with the MSFO-A, MSFO-B, and MSFO-D strategies in all the three problems. The last category is that of the composite functions (F9 to F12), and the MSFO-D strategy showed improvement and achieved the best rank for these functions.

It is important to note that modifying the search strategy (MSFO-A) alone provided a slight improvement, less than 2% overall. However, when the MSFO-A was combined with the MSFO-C or MSFO-D strategy, the performance improved significantly. Similarly, the modified hunting technique, MSFO-C, alone has limited or no impact, but when integrated with the modified search strategy (MSFO-AC), it yielded notable performance, gaining an overall 33% improvement. The two most effective techniques, MSFO-A and MSFO-D, combined together as the MSFO-AD, achieved the best results, providing an overall improvement of 76%, obtaining a higher rank than any other modification. The final version, the MSSFO, combining all the proposed strategies, showed 80% improvement over the original SFO, making it the best choice among all the tested variants. To further analyze the significance of the MSSFO, the Wilcoxon signed-rank test was performed at a 95% significance level, which again resulted in a p-value lower than 0.05. This validates that the MSSFO significantly improves the SFO and can be further evaluated with other optimization algorithms.

After validation, the performance of the MSSFO was compared with the other well-known optimization algorithms, including the SFO, two other variants of the SFO, the SCA, the ISCA, the WOA, and the IWOA, as discussed in Section 4.4. The MSSFO is evidently the best optimizer for 10-dimensional problems, except for F10, where the MSSFO achieved a second rank. However, it ranked at the top when the average was calculated. For 20-dimensional problems, the MSSFO was competitive with respect to the SCA and the ISCA and significantly better than all other algorithms and remained on top in the average ranking. The convergence curves were also presented to show the fast convergence capabilities of the MSSFO.

A statistical analysis is essential to identifying the significant differences among the optimization methods. Since the non-parametric tests do not rely on strict data assumptions [88], they are appropriate for comparing the stochastic optimizers. In this study, the Friedman test was used to analyze rankings for both the 10- and 20-dimensional problems. The results confirmed the statistical significance of the MSSFO's superior performance. A pairwise Wilcoxon signed-rank test was also conducted to compare the MSSFO against each algorithm individually, and again, the MSSFO showed statistically significant improvements with p-values below the 0.05 threshold.

Finally, the MSSFO was applied to two real-world problems. The first application was to minimize the spring weight for the compression spring design (CSD) problem. The results showed that the MSSFO is robust when applied to the CSD problem. It is also evident from the convergence graph that the MSSFO has the capability of performing better exploitation of the search space and that it converges quickly towards the optimal/near-optimal solution. Furthermore, the box plot for 30 optimal values of the CSD problem validates the efficacy and accuracy of the MSSFO in finding optimal solutions with robustness.

The second real-world problem evaluated in this study was the mobile robot path planning problem. The objective is to minimize the path length of a mobile robot while avoiding the obstacles encountered during the journey. Two environments were set up, the first with one obstacle and the second one with five multi-shaped obstacles. In the first scenario, although the MSSFO was competitive with respect to the WOA in finding the minimum path length, the average path length was significantly better than that of the WOA. This confirms the robustness of the proposed algorithm. In the second scenario, where five obstacles were placed, the MSSFO was the best optimizer with the lowest average path length and minimum deviation. The convergence curves of both the scenarios verified the effectiveness of the MSSFO in solving the robot path planning problem.

Overall, the performance of the MSSFO was significantly better than the comparison algorithms in solving the CEC 2022 SO-BO problems and the two real-world problems considered in this study, hence confirming the significance of the MSSFO as a potential candidate for solving real-world optimization problems with efficacy and robustness.

Future Research Directions

The superiority of the MSSFO raises several important open research directions that warrant further investigation. One key question is its ability to solve more complex real-world problems, like the robot path planning with various irregularly shaped obstacles, multi-robot path planning, and feature selection in machine learning, among others. Another open challenge is the scalability of the MSSFO to high-dimensional, real-time, or industrial-scale optimization problems, raising the question of whether hybridized or parallel implementations could enhance its efficiency. The modifications proposed in this study open several avenues for further investigation. The observed standalone effects of the proposed strategies indicate that the modified hunting mechanism merits further independent investigation to achieve maximum benefit and improve solution quality. Furthermore, multi-objective and many-objective variants of the MSSFO can also be developed and tested on real-world problems. The motivation for the development of such algorithms is the demand of recent complex problems in the areas of engineering and science. These open research directions provide a basis for advancing both theoretical development and real-world applicability in future research.

6. Conclusions

In this study, an improved variant of the sailfish optimizer (SFO), named the Multi-Strategy Sailfish Optimizer (MSSFO), is developed by integrating several improvement strategies, including an improved search strategy, a non-linear AP parameter, a modified hunting mechanism, and dynamic sardine population. The proposed modifications were designed to strengthen the algorithm's search capability, balance exploration and exploitation, and improve the overall solution accuracy. Based on the experimental evaluation, the MSSFO demonstrated strong optimization performance and reliable convergence behavior. The findings confirm that the MSSFO is a robust and effective optimization method suitable for a wide range of benchmark and real-world problems. The integrated improvements enable it to achieve competitive results across diverse problem types. Overall, the MSSFO represents a significant improved variant of the original SFO and provides a solid foundation for future extensions and applications.

Author Contributions: Conceptualization, S.N.A., M.R.T., R.C.V. and M.H.T.; methodology, S.N.A., M.R.T. and S.A.K.; software, S.N.A. and M.R.T.; validation, S.A.K. and A.A.S.; formal analysis, M.R.T. and S.A.K.; investigation, M.R.T. and S.A.K.; resources, S.A.K.; data curation, S.N.A. and M.R.T.; writing—original draft preparation, S.N.A. and M.R.T.; writing—review and editing, A.A.S., S.A.K., R.C.V. and M.H.T.; visualization, S.N.A., M.R.T. and S.A.K.; supervision, M.R.T., A.A.S. and S.A.K.; project administration, S.A.K.; funding acquisition, R.C.V. and M.H.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research study received no external funding.

Data Availability Statement: The data used for this research study are included in the article.

Acknowledgments: The authors are thankful to the Karachi Institute of Economics and Technology, Karachi, Pakistan, for providing the simulation environment and to Narmeen Z. Bawany, whose selfless contribution and unwavering support played a pivotal role in the completion of this work.

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

Abbreviations

The following abbreviations are used in this manuscript:

ACO	Ant colony optimization
AGPSO	Autonomous Group Particle Swarm Optimization
AOA	Arithmetic optimization algorithm
AP	Attack power
BA	Bat algorithm
BLTLBO	Balanced teaching–learning-based optimizer
CHO	Chimp optimization algorithm
CUDA	Compute Unified Device Architecture
CSD	Compression spring design
DE	Differential evolution
ELM	Extreme machine learning model
ES	Evolutionary strategy
ET	Execution Time
FTID	Fuzzy logic-based Tilt–Integral–Derivative
GA	Genetic algorithm
GAN	Generative adversarial network
GPU	Graphics Processing Unit
GWO	Gray wolf optimizer

ISCA	Improved sine–cosine algorithm
ISSA	Improved salp swarm algorithm
ITS	Intelligent transportation system
IWOA	Improved whale optimization algorithm
MaxFEs	Maximum number of Function Evaluations
MSFO	Modified Sailfish Optimizer
MSSFO	Multi-Strategy Sailfish Optimizer
NFL	No Free Lunch
NIOAs	Nature-inspired optimization algorithms
PD	Prey density
PP	Percentage Population
PSO	Particle Swarm Optimization
RTH	Red-Tailed Hawk algorithm
SCA	Sine–cosine algorithm
SFO	Sailfish optimizer
SIA	Swarm-intelligence-based algorithm
SO-BO	Single-objective bound-constrained optimization
TLBO	Teaching–learning-based optimizer
TSA	Tunicate Swarm Algorithm
TSP	Traveling salesman problem
VMD	Variational mode decomposition
WCHO	Weighted chimp optimization
WOA	Whale optimization algorithm
WSN	Wireless sensor network
WSO	White shark optimization algorithm

References

1. Laporte, G. The traveling salesman problem: An overview of exact and approximate algorithms. *Eur. J. Oper. Res.* **1992**, *59*, 231–247. [CrossRef]
2. Zhou, T.; Wei, W. Mobile robot path planning based on an improved ACO algorithm and path optimization. *Multimed. Tools Appl.* **2025**, *84*, 10899–10922. [CrossRef]
3. Larabi-Marie-Sainte, S.; Bin Alamir, M.; Alameer, A. Arabic Text Clustering Using Self-Organizing Maps and Grey Wolf Optimization. *Appl. Sci.* **2023**, *13*, 10168. [CrossRef]
4. Kumar, M.; Suman, S. Scheduling in IaaS Cloud Computing Environment using Sailfish Optimization Algorithm. *Trends Sci.* **2022**, *19*, 4204. [CrossRef]
5. Rajamoorthy, R.; Arunachalam, G.; Kasinathan, P.; Devendiran, R.; Ahmadi, P.; Pandiyan, S.; Muthusamy, S.; Panchal, H.; Kazem, H.A.; Sharma, P. A novel intelligent transport system charging scheduling for electric vehicles using Grey Wolf Optimizer and Sail Fish Optimization algorithms. *Energy Sources Part A Recover. Util. Environ. Eff.* **2022**, *44*, 3555–3575. [CrossRef]
6. Zhang, Y.; Mo, Y. Dynamic optimization of chemical processes based on modified sailfish optimizer combined with an equal division method. *Processes* **2021**, *9*, 1806. [CrossRef]
7. Ren, X.; Ma, L. Main Steam Temperature Load Tracking Control Based on Improved Sailfish Optimization. In Proceedings of the Energy Proceedings (International Conference on Applied Energy—ICAE 2020), Bangkok, Thailand, 1–10 December 2020. [CrossRef]
8. Li, L.L.; Shen, Q.; Tseng, M.L.; Luo, S. Power system hybrid dynamic economic emission dispatch with wind energy based on improved sailfish algorithm. *J. Clean. Prod.* **2021**, *316*, 128318. [CrossRef]
9. Antwi, R.; Gadze, J.D.; Tchao, E.T.; Sikora, A.; Obour Agyekum, K.O.B.; Nunoo-Mensah, H.; Agbemenu, A.S.; Agyemang, J.O.; Welte, D.; Ahmed, A.R.; et al. Optimising peer-to-peer topology for blockchain-based industrial internet of things networks using particle swarm optimisation. *Clust. Comput.* **2025**, *28*, 204. [CrossRef]
10. Tounsi, M.; Ali, H.; Azar, A.T.; Al-Khayyat, A.; Ibraheem, I.K. Comprehensive Learning Salp Swarm Algorithm with Ensemble Deep Learning-based ECG Signal Classification on Internet of Things Environment. *Eng. Technol. Appl. Sci. Res.* **2025**, *15*, 19492–19500. [CrossRef]
11. Wu, M. Application of particle swarm optimisation algorithm incorporating frog-leaping algorithm in optimal scheduling for production management in manufacturing plant. *Int. J. Interact. Des. Manuf. (IJIDeM)* **2025**, *19*, 1721–1735. [CrossRef]

12. Kumar, L.; Goyal, A.; Pathak, V.K.; Bhowmik, A. A state-of-the-art review of soft computing-based monitoring and control in the machining of hard alloys. *Discov. Appl. Sci.* **2025**, *7*, 681. [CrossRef]
13. Zambou, M.C.Z.; Kammogne, A.S.T.; Siewe, M.S.; Azar, A.T.; Ahmed, S.; Hameed, I.A. Optimized Nonlinear PID Control for Maximum Power Point Tracking in PV Systems Using Particle Swarm Optimization. *Math. Comput. Appl.* **2024**, *29*, 88. [CrossRef]
14. Mahmood, S.; Bawany, N.Z.; Tanweer, M.R. A comprehensive survey of Whale Optimization Algorithm: Modifications and classification. *Indones. J. Electr. Eng. Comput. Sci.* **2023**, *29*, 899–910. [CrossRef]
15. Yang, X.S. Nature-inspired optimization algorithms: Challenges and open problems. *J. Comput. Sci.* **2020**, *46*, 101104. [CrossRef]
16. Moshtaghi, H.R.; Eshlaghy, A.T.; Motadel, M.R. A Comprehensive Review on Meta-Heuristic Algorithms and their Classification with Novel Approach. *J. Appl. Res. Ind. Eng.* **2021**, *8*, 63–89. [CrossRef]
17. Holland, J.H. Genetic Algorithms. *Sci. Am.* **1992**, *267*, 66–72. [CrossRef]
18. Beyer, H.G. Toward a Theory of Evolution Strategies: The (μ, λ)-Theory. *Evol. Comput.* **1994**, *2*, 381–407. [CrossRef]
19. Storn, R.; Price, K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
20. Dorigo, M.; Maniezzo, V.; Colorni, A. Ant system: Optimization by a colony of cooperating agents. *IEEE Trans. Syst. Man, Cybern. Part B Cybern.* **1996**, *26*, 29–41. [CrossRef]
21. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95—International Conference on Neural Networks, Perth, WA, Australia, 27 November 1995–1 December 1995; Volume 26, pp. 1942–1948. [CrossRef]
22. Yang, X.S.; Gandomi, A.H. Bat algorithm: A novel approach for global engineering optimization. *Eng. Comput.* **2012**, *29*, 464–483. [CrossRef]
23. Mirjalili, S. SCA: A sine cosine algorithm for solving optimization problems. *Knowl.-Based Syst.* **2016**, *96*, 120–133. [CrossRef]
24. Mirjalili, S.; Lewis, A. Advances in Engineering Software The Whale Optimization Algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
25. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
26. Braik, M.; Hammouri, A.; Atwan, J.; Al-Betar, M.A.; Awadallah, M.A. White Shark Optimizer: A novel bio-inspired meta-heuristic algorithm for global optimization problems. *Knowl.-Based Syst.* **2022**, *243*, 108457. [CrossRef]
27. Rao, R.V.; Savsani, V.J.; Vakharia, D.P. Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems. *CAD Comput. Aided Des.* **2011**, *43*, 303–315. [CrossRef]
28. Shadravan, S.; Naji, H.R.; Bardsiri, V.K. The Sailfish Optimizer: A novel nature-inspired metaheuristic algorithm for solving constrained engineering optimization problems. *Eng. Appl. Artif. Intell.* **2019**, *80*, 20–34. [CrossRef]
29. Abualigah, L.; Diabat, A.; Mirjalili, S.; Elaziz, M.A.; Gandomi, A.H. The Arithmetic Optimization Algorithm. *Comput. Methods Appl. Mech. Eng.* **2021**, *376*, 113609. [CrossRef]
30. Khishe, M.; Mosavi, M.R. Chimp optimization algorithm. *Expert Syst. Appl.* **2020**, *149*, 113338. [CrossRef]
31. Li, H.; Liu, X.; Huang, Z.; Zeng, C.; Zou, P.; Chu, Z.; Yi, J. Newly Emerging Nature-Inspired Optimization-Algorithm Review, Unified Framework, Evaluation, and Behavioural Parameter Optimization. *IEEE Access* **2020**, *8*, 72620–72649. [CrossRef]
32. Siddique, N.; Adeli, H. Nature Inspired Computing: An Overview and Some Future Directions. *Cogn. Comput.* **2015**, *7*, 706–714. [CrossRef]
33. Tzanetos, A.; Dounias, G. Nature inspired optimization algorithms or simply variations of metaheuristics? *Artif. Intell. Rev.* **2021**, *54*, 1841–1862. [CrossRef]
34. Suid, M.H.; Ahmad, M.A.; Ismail, M.R.T.R.; Ghazali, M.R.; Irawan, A.; Tumari, M.Z. An Improved Sine Cosine Algorithm for Solving Optimization Problems. In Proceedings of the 2018 IEEE Symposium on Computer Applications & Industrial Electronics (SPC), Melaka, Malaysia, 14–15 December 2018 [CrossRef]
35. Tanweer, M.R.; Auditya, R.; Suresh, S.; Sundararajan, N.; Srikanth, N. Directionally Driven Self-Regulating Particle Swarm Optimization algorithm. *Swarm Evol. Comput.* **2016**, *28*, 98–116. [CrossRef]
36. Tudose, A.M.; Picioroaga, I.I.; Sidea, D.O.; Bulac, C. Solving single-and multi-objective optimal reactive power dispatch problems using an improved salp swarm algorithm. *Energies* **2021**, *14*, 1222. [CrossRef]
37. Taheri, A.; RahimiZadeh, K.; Rao, R.V. An efficient Balanced Teaching-Learning-Based optimization algorithm with Individual restarting strategy for solving global optimization problems. *Inf. Sci.* **2021**, *576*, 68–104. [CrossRef]
38. Pandey, A.C.; Tikkiwal, V.A. Stance detection using improved whale optimization algorithm. *Complex Intell. Syst.* **2021**, *7*, 1649–1672. [CrossRef]
39. Wolpert, D.H.; Macready, W.G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [CrossRef]
40. Issam, E.H.; Lajjam, A.; Merouani, M.E.; Tabaa, Y. A modified sailfish optimizer to solve dynamic berth allocation problem in conventional container terminal. *Int. J. Ind. Eng. Comput.* **2019**, *10*, 491–504. [CrossRef]
41. Ghosh, K.K.; Ahmed, S.; Singh, P.K.; Geem, Z.W.; Sarkar, R. Improved binary sailfish optimizer based on adaptive B-Hill climbing for feature selection. *IEEE Access* **2020**, *8*, 83548–83560. [CrossRef]

42. Zhang, Y.; Mo, Y. Chaotic adaptive sailfish optimizer with genetic characteristics for global optimization. *J. Supercomput.* **2022**, *78*, 10950–10996. [CrossRef]
43. Mohammadi, R.; Akleylek, S.; Ghaffari, A. SDN-IoT: SDN-based efficient clustering scheme for IoT using improved Sailfish optimization algorithm. *PeerJ Comput. Sci.* **2023**, *9*, e1424. [CrossRef] [PubMed]
44. Hongfei, L.; Guohui, Z.; Huang, B.; Chi, M.A.; Gong, C.; Jinbin, Z. Optimal Control Virtual Inertia of Optical Storage Microgrid Based on Improved Sailfish Algorithm. *Wuhan Univ. J. Nat. Sci.* **2022**, *27*, 218–230. [CrossRef]
45. Shajin, F.H.; Devi, B.A.; Prakash, N.B.; Sreekanth, G.R.; Rajesh, P. Sailfish optimizer with Levy flight, chaotic and opposition-based multi-level thresholding for medical image segmentation. *Soft Comput.* **2023**, *27*, 12457–12482. [CrossRef]
46. Cao, L.; Yue, Y.; Chen, Y.; Chen, C.; Chen, B. Sailfish optimization algorithm integrated with the osprey optimization algorithm and cauchy mutation and its engineering applications. *Symmetry* **2025**, *17*, 938. [CrossRef]
47. Amin, S.A.; Alqudah, M.K.S.; Almutairi, S.A.; Almajed, R.; Al Nasar, M.R.; Alkhazaleh, H.A. Optimal extreme learning machine for diagnosing brain tumor based on modified sailfish optimizer. *Heliyon* **2024**, *10*, e34050. [CrossRef]
48. Babu, J.S.; Niranjana, G.; Ramana, K. Adaptive sail fish optimizer based optimized region selection for robust video steganography. *Int. J. Inf. Technol.* **2025**, *17*, 2417–2427. [CrossRef]
49. Sedak, M.; Rosić Vitas, M.; Rosić, B. Multi-objective hybrid sailfish optimization algorithm for planetary gearbox and mechanical engineering design optimization problems. *Comput. Model. Eng. Sci.* **2025**, 142, 2111–2145. [CrossRef]
50. Azzam, S.M.; Emam, O.; Abolaban, A.S. An improved Differential evolution with Sailfish optimizer (DESFO) for handling feature selection problem. *Sci. Rep.* **2024**, *14*, 13517. [CrossRef]
51. Li, M.; Li, Y.; Chen, Y.; Xu, Y. Batch recommendation of experts to questions in community-based question-answering with a sailfish optimizer. *Expert Syst. Appl.* **2021**, *169*, 114484. [CrossRef]
52. Togacar, M.; Ergen, B. Classification of cloud images by using super resolution, semantic segmentation approaches and binary sailfish optimization method with deep learning model. *Comput. Electron. Agric.* **2022**, *193*, 106724. [CrossRef]
53. Naji, H.R.; Shadravan, S.; Jafarabadi, H.M.; Momeni, H. Accelerating sailfish optimization applied to unconstrained optimization problems on graphical processing unit. *Eng. Sci. Technol. Int. J.* **2022**, *32*, 101077. [CrossRef]
54. Shadravan, S.; Naji, H.R.; Khatibi, V. A Distributed Sailfish Optimizer Based on Multi-Agent Systems for Solving Non-Convex and Scalable Optimization Problems Implemented on GPU. *Technol. J. Artif. Intell. Data Min.* **2021**, *9*, 59–71. [CrossRef]
55. Geetha, P.; Nanda, S.J.; Yadav, R.P. A parallel chaotic sailfish optimization algorithm for estimation of DOA in wireless sensor array. *Phys. Commun.* **2022**, *51*, 101536. [CrossRef]
56. Hammouti, I.E.; Lajjam, A.; Merouani, M.E. Comparison of planning models for dynamic berth allocation problem using a sailfish-based algorithm. *Procedia Comput. Sci.* **2020**, *176*, 3112–3120. [CrossRef]
57. Sankar, S.; Ramasubbareddy, S.; Chen, F.; Gandomi, A.H. Energy-Efficient Cluster-Based Routing Protocol in Internet of Things Using Swarm Intelligence. In Proceedings of the 2020 IEEE Symposium Series on Computational Intelligence (SSCI), Canberra, ACT, Australia, 1–4 December 2020; pp. 219–224. [CrossRef]
58. Amutha, R.; Sivasankari, G.G.; Venugopal, K.R. Node clustering and data aggregation in wireless sensor network using sailfish optimization. *Multimed. Tools Appl.* **2023**, *82*, 44107–44122. [CrossRef]
59. Mehta, D.; Saxena, S. MCH-EOR: Multi-objective Cluster Head Based Energy-aware Optimized Routing algorithm in Wireless Sensor Networks. *Sustain. Comput. Inform. Syst.* **2020**, *28*, 100406. [CrossRef]
60. Kumar, M.; Ali, J. Taylor Sailfish Optimizer-Based Deep Stacked Auto Encoder for Blackhole Attack Detection in Wireless Sensor Network. *J. Web Eng.* **2022**, *21*, 911–940. [CrossRef]
61. Kumar, B.S.; Santhi, S.G.; Narayana, S. Sailfish optimizer algorithm (SFO) for optimized clustering in wireless sensor network (WSN). *J. Eng. Des. Technol.* **2021**, 20, 1449–1467. [CrossRef]
62. Rajoriya, M.K.; Gupta, C.P. Sailfish optimization-based controller selection (SFO-CS) for energy-aware multi-hop routing in software defined wireless sensor network (SDWSN). *Int. J. Inf. Technol.* **2023**, *15*, 3935–3948. [CrossRef]
63. Samal, P.; Roshan, R. Optimal STATCOM allocation and sizing using the sailfish optimizer algorithm. In Proceedings of the 2020 IEEE 7th Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON), Prayagraj, India, 27–29 November 2020; pp. 1–6. [CrossRef]
64. Duhayyim, M.A.; Malibari, A.A.; Dhahbi, S.; Nour, M.K.; Al-Turaiki, I.; Obayya, M.; Mohamed, A. Sailfish Optimization with Deep Learning Based Oral Cancer Classification Model. *Comput. Syst. Sci. Eng.* **2023**, *45*, 753–767. [CrossRef]
65. Hamza, M.A.; Hassine, S.B.H.; Abunadi, I.; Al-Wesabi, F.N.; Alsolai, H.; Hilal, A.M.; Yaseen, I.; Motwakel, A. Feature Selection with Optimal Stacked Sparse Autoencoder for Data Mining. *Comput. Mater. Contin.* **2022**, *72*, 2581–2596. [CrossRef]
66. Selvapandian, A.; N. Prabhu, S.; Sivakumar, P.; J. Rao, D.B. Lung cancer detection and severity level classification using sine cosine sail fish optimization based generative adversarial network with CT images. *Comput. J.* **2022**, *65*, 1611–1630. [CrossRef]
67. Khan, N.M.; Khan, U.A.; Zafar, M.H. Maximum power point tracking of PV system under uniform irradiance and partial shading conditions using machine learning algorithm trained by sailfish optimizer. In Proceedings of the 2021 4th International Conference on Energy Conservation and Efficiency (ICECE), Lahore, Pakistan, 16–17 March 2021; pp. 1–6. [CrossRef]

68. Alqahtani, M.M.; Dutta, A.K.; Almotairi, S.; Ilayaraja, M.; Albraikan, A.A.; Al-Wesabi, F.N.; Duhayyim, M.A. Sailfish Optimizer with EfficientNet Model for Apple Leaf Disease Detection. *Comput. Mater. Contin.* **2023**, *74*, 217–233. [CrossRef]
69. Nagapavithra, S.; Umamaheswari, S. Detection and classification of sugarcane billet damage using Aquila Sailfish Optimizer based deep learning. *Artif. Intell. Rev.* **2023**, *56*, 2183–2206. [CrossRef]
70. Gutte, V.S.; Parasar, D. Sailfish invasive weed optimization algorithm for multiple image sharing in cloud computing. *Int. J. Intell. Syst.* **2022**, *37*, 4190–4213. [CrossRef]
71. Kalpana, P. Chronological sailfish optimizer for preserving privacy in cloud based on khatri-rao product. *Comput. J.* **2021**, *66*, 101–113. [CrossRef]
72. Nassef, M.G.A.; Hussein, T.M.; Mokhiamar, O. An adaptive variational mode decomposition based on sailfish optimization algorithm and Gini index for fault identification in rolling bearings. *Measurement* **2021**, *173*, 108514. [CrossRef]
73. Rai, A.; Das, D.K. The development of a fuzzy tilt integral derivative controller based on the sailfish optimizer to solve load frequency control in a microgrid, incorporating energy storage systems. *J. Energy Storage* **2022**, *48*, 103887. [CrossRef]
74. Cheng, X.; Li, J.; Zheng, C.; Zhang, J.; Zhao, M. An improved PSO-GWO algorithm with chaos and adaptive inertial weight for robot path planning. *Front. Neurorobotics* **2021**, *15*, 770361. [CrossRef] [PubMed]
75. Ab Wahab, M.N.; Nazir, A.; Khalil, A.; Ho, W.J.; Akbar, M.F.; Noor, M.H.M.; Mohamed, A.S.A. Improved genetic algorithm for mobile robot path planning in static environments. *Expert Syst. Appl.* **2024**, *249*, 123762. [CrossRef]
76. Cui, J.; Wu, L.; Huang, X.; Xu, D.; Liu, C.; Xiao, W. Multi-strategy adaptable ant colony optimization algorithm and its application in robot path planning. *Knowledge-Based Syst.* **2024**, *288*, 111459. [CrossRef]
77. Sun, J.; Feng, B.; Xu, W. Particle swarm optimization with particles having quantum behavior. In Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753), Portland, OR, USA, 19–23 June 2004; IEEE: Piscataway, NJ, USA, 2004, Volume 1, pp. 325–331. [CrossRef]
78. Sun, J.; Xu, W.; Feng, B. A global search strategy of quantum-behaved particle swarm optimization. In Proceedings of the IEEE Conference on Cybernetics and Intelligent Systems, Singapore,1–3 December 2004; IEEE: Piscataway, NJ, USA, 2004; Volume 1, pp. 111–116. [CrossRef]
79. Kononova, A.V.; Corne, D.W.; De Wilde, P.; Shneer, V.; Caraffini, F. Structural bias in population-based algorithms. *Inf. Sci.* **2015**, *298*, 468–490. [CrossRef]
80. Long, W.; Wu, T.; Liang, X.; Xu, S. Solving high-dimensional global optimization problems using an improved sine cosine algorithm. *Expert Syst. Appl.* **2019**, *123*, 108–126. [CrossRef]
81. Kumar, A.; Price, K.; Mohamed, A.W.; Hadi, A.A.; Suganthan, P.N. Problem definitions and evaluation criteria for the CEC 2022 special session and competition on Single Objective Bound Constrained Numerical Optimization. *Tech. Rep.* 2021. [CrossRef]
82. Derrac, J.; García, S.; Molina, D.; Herrera, F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm Evol. Comput.* **2011**, *1*, 3–18. [CrossRef]
83. Sheskin, D.J. *Handbook of Parametric and Nonparametric Statistical Procedures*, 3rd ed.; CRC Press: Boca Raton, FL, USA, 2003. [CrossRef]
84. Davenport, J.M. Approximations of the critical region of the Friedman statistic. *Commun. Stat. Theory Methods* **1980**, *9*, 571–595. [CrossRef]
85. García, S.; Fernández, A.; Luengo, J.; Herrera, F. A study of statistical techniques and performance measures for genetics-based machine learning: Accuracy and interpretability. *Soft Comput.* **2009**, *13*, 959–977. [CrossRef]
86. Zhang, Q.; Du, S.; Zhang, Y.; Wu, H.; Duan, K.; Lin, Y. A Novel Chimp Optimization Algorithm with Refraction Learning and Its Engineering Applications. *Algorithms* **2022**, *15*, 189. [CrossRef]
87. Massoud, M.M.; Abdellatif, A.; Atia, M.R. Different path planning techniques for an indoor omni-wheeled mobile robot: experimental implementation, comparison and optimization. *Appl. Sci.* **2022**, *12*, 12951. [CrossRef]
88. Siegel, S. Nonparametric Statistics. *Am. Stat.* **1957**, *11*, 13–19. [CrossRef]

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