

Article

A Hybrid JADE–Sine Cosine Approach for Advanced Metaheuristic Optimization

Abdelraouf Ishtaiwi ¹, Ahmad Sami Al-Shamayleh ^{2,*} and Hussam N. Fakhouri ¹¹ Data Science and Artificial Intelligence, Faculty of Information Technology, University of Petra, Amman 11932, Jordan; aishtaiwi@uop.edu.jo (A.I.); hussam.fakhouri@uop.edu.jo (H.N.F.)² Department of Data Science and Artificial Intelligence, Faculty of Information Technology, Al-Ahliyya Amman University, Amman 19328, Jordan

* Correspondence: a.alshamayleh@ammanu.edu.jo

Abstract: This paper presents the development and application of the JADESCA optimization algorithm for solving complex engineering design problems, including the welded beam, pressure vessel, spring, and speed reducer design problems. JADESCA, a hybrid algorithm that combines elements of JADE (differential evolution with adaptive parameters) and the sine cosine algorithm (SCA), is evaluated against a range of benchmark functions from the CEC2022 competition as well as specific engineering problems. The algorithm's performance is analyzed through convergence curves, search history diagrams, and statistical results. In engineering design problems, JADESCA consistently demonstrates superior performance by achieving optimal or near-optimal solutions with high precision and consistency. In particular, JADESCA outperforms 25 state-of-the-art optimizers over the CEC2022 benchmark functions, further proving its robustness and adaptability. Statistical comparisons and Wilcoxon rank-sum tests reinforce the superiority of JADESCA in achieving competitive results across various test cases, solidifying its effectiveness in handling complex, constrained optimization problems for engineering applications.

Keywords: optimization; metaheuristic; engineering design; differential evolution



Citation: Ishtaiwi, A.; Al-Shamayleh, A.S.; Fakhouri, H.N. A Hybrid JADE–Sine Cosine Approach for Advanced Metaheuristic Optimization. *Appl. Sci.* **2024**, *14*, 10248. <https://doi.org/10.3390/app142210248>

Received: 26 August 2024

Revised: 26 October 2024

Accepted: 31 October 2024

Published: 7 November 2024



Copyright: © 2024 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 (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Metaheuristic algorithms and artificial intelligence (AI) have revolutionized the field of optimization by providing powerful tools to solve complex problems that traditional methods struggle to address. Metaheuristics, inspired by natural processes such as evolution, swarm behavior, and annealing, offer a flexible framework for exploring vast search spaces and finding near-optimal solutions efficiently [1]. Algorithms like genetic algorithms, particle swarm optimization, and simulated annealing have demonstrated remarkable success in various domains by balancing exploration and exploitation to avoid local optima and enhance solution quality [2]. Meanwhile, AI techniques, including machine learning and deep learning, leverage vast amounts of data and sophisticated models to learn patterns and make predictions that guide optimization processes. Integrating AI with metaheuristics has further amplified their capabilities, enabling adaptive and intelligent search strategies that continuously improve over time. This synergy between metaheuristics and AI represents a significant advancement in optimization, offering robust and scalable solutions for increasingly complex real-world problems [3].

Metaheuristic algorithms and artificial intelligence (AI) have become essential tools in solving complex optimization problems that traditional methods struggle with. Optimization algorithms have achieved success across various fields, including energy, healthcare, construction, and cybersecurity. For instance, in energy systems, metaheuristics optimize resource allocation and improve renewable energy efficiency [4,5], while in healthcare, they enhance medical diagnosis and treatment planning [6]. In construction and cybersecurity, these algorithms are used for project scheduling, structural design, and threat detection [7].

Optimization, the pursuit of the best combination of decision variables, is crucial in solving a myriad of problems across diverse fields and applications. From designing efficient engineering systems and optimizing resource allocation to solving complex scientific computations and fine-tuning machine learning models, optimization lies at the heart of innovation and advancement [8]. In the realm of engineering, optimization is essential for creating designs that meet specific performance criteria while minimizing costs and maximizing efficiency. For instance, in structural engineering, optimization techniques are used to determine the best configuration of materials and structures to ensure safety and durability. Similarly, in aerospace engineering, optimization helps in designing aircraft and spacecraft that achieve the best possible performance with the least weight and fuel consumption [9].

Traditional optimization techniques, such as linear programming and non-linear programming, have long been the foundation of this field. These methods, grounded in mathematical rigor, rely heavily on gradient information and an appropriately chosen initial start vector. While powerful for well-behaved, small-scale problems, these methods often fall short when faced with the intricate landscapes of real-world problems [10]. The requirement for gradient information means they can struggle with non-differentiable or noisy objective functions. Moreover, their dependency on a good initial guess makes them prone to getting stuck in local optima, failing to find the global best solution in complex, high-dimensional spaces [11].

To overcome these limitations, hybrid meta-heuristics have emerged as powerful alternatives. Hybrid optimization methods amalgamate the strengths of various algorithms to address complex engineering design problems, often marked by high dimensionality, non-linearity, and multimodal functions [12]. These problems present significant challenges due to the vast and intricate nature of their solution spaces, where traditional optimization techniques might either fail to find optimal solutions or require excessive computational resources. By leveraging the complementary strengths of multiple algorithms, hybrid optimization methods aim to enhance the efficiency, accuracy, and robustness of the search process [13].

This research paper introduces a novel hybrid approach, integrating the sine cosine algorithm (SCA) with the advanced evolutionary algorithm known as JADE (adaptive differential evolution with an optional external archive), to effectively solve such intricate problems. The SCA is renowned for its simplicity and effectiveness in exploring the search space, utilizing mathematical sine and cosine functions to update the positions of solutions. This ensures a diverse and widespread exploration, making SCA adept at avoiding local optima and promoting global exploration in the initial stages of the optimization process [14].

Conversely, JADE incorporates adaptive mechanisms and an external archive to enhance its performance. It adjusts its mutation and crossover parameters dynamically, maintaining a balance between exploration and exploitation. JADE's external archive stores successful solutions, guiding future search directions and improving convergence speed. Its robust adaptive mechanisms and effective local search capabilities make it highly suitable for fine-tuning solutions and converging towards the global optimum in the later stages of the optimization process [15].

In recent years, hybrid optimization methods have gained considerable attention due to their effectiveness in solving complex engineering design problems. For instance, Dash et al. [16] introduced a hybrid approach combining the empirical wavelet transform with sine cosine and particle swarm optimization for enhanced solar power forecasting, achieving significant improvements in accuracy. Similarly, Zhao et al. [17] applied a hybrid grey wolf optimizer and sine cosine algorithm for construction project scheduling, demonstrating superior convergence and solution quality. These studies, along with others like Hussain et al. [18], who proposed a hybrid sine cosine algorithm with Harris hawks optimization for feature selection, showcase the growing interest in hybrid techniques. These examples highlight how hybrid approaches are increasingly utilized to

overcome the limitations of single algorithms in addressing high-dimensional, multimodal, and non-linear problems commonly encountered in engineering design. By leveraging the complementary strengths of different algorithms, hybrid methods provide improved exploration, exploitation, and convergence characteristics, making them well-suited for solving real-world engineering challenges.

The motivation for developing this hybrid approach lies in overcoming the inherent limitations of individual algorithms. The SCA is particularly adept at global exploration due to its mathematical design, which ensures a diverse set of candidate solutions and avoids premature convergence. However, it struggles with fine-tuning solutions during the later stages of optimization. Conversely, JADE excels in local search capabilities and adaptability but may require enhancement in its initial exploration phase. By combining these algorithms, we leverage their strengths and mitigate their weaknesses, leading to a more efficient and effective optimization process [19].

2. Literature Review

In recent years, numerous hybrid optimization methods have been proposed to tackle complex engineering design problems, leveraging the strengths of different algorithms to enhance performance in terms of efficiency, accuracy, and robustness. Dash et al. [16] introduced a hybrid approach integrating empirical wavelet transform (EWT) with a robust minimum variance random vector functional link network (RRVFLN), optimized using a sine cosine and Levy flight-based particle swarm optimization (PSOLVSC). This method demonstrated significant improvements in short-term solar power forecasting accuracy, showcasing the effectiveness of combining multiple optimization strategies to handle large and time-varying databases. Similarly, Zhao et al. [17] explored the optimization of construction duration and schedule robustness using a hybrid grey wolf optimizer (GWO) with the sine cosine algorithm (SCA). Their model improved convergence, diversity, accuracy, and distribution, proving its efficacy through a case study of an underground power station in China.

Another notable study by Hussain et al. [18] proposed a hybrid optimization method combining SCA with Harris hawks optimization (HHO) to enhance exploration and exploitation capabilities, particularly for low and high-dimensional feature selection tasks. Their extensive experimental and statistical analyses confirmed the superior performance of the method in terms of search efficiency and convergence speed. Further extending the application of hybrid methods, Zhao et al. [20] developed a hybrid algorithm integrating GWO, opposition-based learning, SCA, and reinforcement learning for reliable scheduling and resource allocation in space debris tracking. This fusion algorithm demonstrated improved performance in managing allocation and scheduling problems, validated through real-world data.

In the domain of biometric identification, Kamble et al. [21] proposed a hybrid model combining the discrete cosine transform (DCT) and deep learning for children's fingerprint identification. Their approach achieved a remarkable accuracy of 99% by integrating traditional and modern computational techniques. Similarly, Yousef et al. [22] introduced an adaptive sine cosine algorithm (ASCA) for optimizing PID controllers in hybrid renewable energy systems (HRES). By enhancing the traditional SCA with Levy flight distribution and adaptive operators, their method significantly improved the performance of HRES controllers under various operating conditions.

Bansal and Wadhawan [23] addressed the heterogeneous fixed fleet vehicle routing problem using a hybrid approach combining SCA and particle swarm optimization (PSO). Their HSPS algorithm exhibited faster convergence rates and better solution quality compared to other hybrid algorithms. Li et al. [24] proposed an improved hybrid particle swarm optimizer with sine cosine acceleration coefficients (IH-PSO-SCAC) for transient electromagnetic inversion. Employing reverse learning strategies and sine cosine mapping, their method enhanced population diversity and global convergence, demonstrating superior stability and accuracy in geophysical applications.

Begum et al. [25] presented a robust digital watermarking system based on a combination of DCT, discrete wavelet transform (DWT), and singular value decomposition (SVD). This hybrid approach ensured improved robustness, high imperceptibility, and enhanced security, making it effective against various attacks. Khajehzadeh et al. [26] introduced a particle swarm sine cosine algorithm (PSSCA) for optimizing retaining structures under seismic loads. Their hybrid algorithm leveraged the dynamic behavior of sine and cosine functions to achieve faster convergence and better accuracy, outperforming other competitive algorithms in seismic optimization tasks.

Hassan et al. [27] proposed an adaptive load frequency control (LFC) for isolated microgrids (μ Gs) that includes PV systems and electric vehicles (EVs). Utilizing a combination of sine cosine optimization (SCO) and balloon effect identifier (BEI) algorithms, their method ensured stability under various conditions, demonstrating significant improvements in regulating the frequency and power of an islanded μ G under variable load and system parameters uncertainties. Talebi et al. [28] investigated the effect of hybrid nanofluids on heat transfer around a vertical rod with cosine heat flux. By adding TiO_2 and Al_2O_3 nanoparticles to deionized water, they achieved significant temperature reduction and improved thermal performance in both turbulence and laminar flow regimes, particularly under varying pressures.

Hans and Kaur [29] proposed a hybrid binary SCA combined with ant-lion optimization (ALO) for feature selection problems. Their hybrid SCALO algorithm showed superior performance in eliminating inappropriate features and enhancing classification accuracy across diverse datasets. Song et al. [30] addressed mixed-integer nonlinear programming problems by presenting a hybrid improved sine and cosine algorithm. Their improved position update formula and combined mutation strategies significantly enhanced global and local search abilities, providing effective solutions for high-dimensional optimization problems. Gu et al. [31] developed a hybrid compression method for NMR data using window averaging and DCT. Their WAC-DCT method effectively reduced inversion time while maintaining high compression rates, demonstrating superior performance compared to other compression methods.

Rawa [32] proposed a transmission expansion planning model to avoid cascading failures in modern active power systems. By integrating the snake optimizer and sine cosine algorithm (SO-SCA), the model effectively minimized cascading outages and improved system resilience under different failure scenarios. Rosli et al. [33] introduced a hybrid optimization method combining SCA with the cuckoo search algorithm (CSA). Their hybrid modified sine cosine algorithm cuckoo search algorithm (HMSCACSA) demonstrated faster convergence and reduced computational time, outperforming other hybrid metaheuristics in various benchmark tests.

Naik et al. [34] proposed an optimized droop controller for energy management in stand-alone microgrids using a hybrid monarch butterfly and sine cosine algorithm. Their SCMBO method achieved optimal energy management solutions and maintained microgrid stability under various load conditions. Gupta and Deep [35] hybridized the sine cosine algorithm (SCA) with an artificial bee colony (ABC) algorithm to address global optimization and image segmentation problems. Their SCABC algorithm enhanced exploration and exploitation capabilities, achieving better search results and image segmentation performance compared to classical ABC and other optimization algorithms.

In recent years, metaheuristic algorithms have gained popularity in tackling optimization problems due to their flexibility and effectiveness in handling complex, nonlinear systems. The carnivorous plant algorithm (CPA), recently proposed as a population-based optimization approach, mimics the natural strategies of carnivorous plants to balance exploration and exploitation phases. An improved version, termed I-CPA, integrates a teaching factor strategy to enhance exploitation capabilities and stabilize results. The I-CPA has shown competitive performance in solving solar photovoltaic (PV) parameter identification problems, achieving lower root mean square error (RMSE) values compared to other metaheuristics [36].

Similarly, the tree seed algorithm (TSA) has been proposed to address continuous optimization issues by modeling the relationship between trees and seeds. To address limitations like local minima entrapment and loss of diversity, a multi-strategy tree seed algorithm (MS-TSA) was introduced, incorporating self-adaptive weighting, chaotic elite learning, and experience-based learning. When applied to PV parameter estimation, MS-TSA demonstrated superior performance over traditional TSA and other algorithms, achieving lower RMSE values and proving its effectiveness in real-world applications [37].

2.1. Overview of JADE

JADE (adaptive differential evolution with an optional external archive) is an advanced evolutionary algorithm designed to enhance the traditional differential evolution (DE) framework. It introduces adaptive mechanisms and an optional external archive to improve performance in terms of convergence speed, robustness, and solution quality. A key innovation in JADE is its dynamic adjustment of mutation and crossover parameters, which balances exploration and exploitation throughout the optimization process. Unlike traditional DE, JADE continuously adapts these parameters based on the current state of the search, ensuring effective performance across different optimization stages.

Another significant feature is the external archive, which stores successful solutions from previous generations. This archive serves as a memory, guiding future searches and helping to avoid premature convergence to local optima. JADE employs a modified DE/current-to-p-best/1 mutation strategy, generating difference vectors based on the current population and a subset of the best solutions. This enhances the algorithm's ability to exploit promising search space regions while maintaining diversity. Additionally, JADE uses a binomial crossover mechanism with a dynamically adjusted crossover rate, promoting diversity and preventing stagnation by ensuring offspring inherit a mix of traits from parent solutions.

The mutation strategy in JADE is given as shown in Equation (1):

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F \cdot (\mathbf{x}_{pbest,g} - \mathbf{x}_{i,g}) + F \cdot (\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}), \quad (1)$$

where $\mathbf{x}_{i,g}$ is the current vector, $\mathbf{x}_{pbest,g}$ is the p-best individual, and $\mathbf{x}_{r1,g}$ and $\mathbf{x}_{r2,g}$ are two randomly selected vectors from the current population.

The crossover operation is defined as follows:

$$\mathbf{u}_{i,j,g} = \begin{cases} \mathbf{v}_{i,j,g} & \text{if } \text{rand}_j[0,1) \leq CR \text{ or } j = j_{rand} \\ \mathbf{x}_{i,j,g} & \text{otherwise} \end{cases},$$

where $\mathbf{u}_{i,j,g}$ is the trial vector, CR is the crossover rate, and j_{rand} is a randomly chosen index to ensure that $\mathbf{u}_{i,g}$ gets at least one component from $\mathbf{v}_{i,g}$.

The adaptive parameter control and use of the external archive contribute to JADE's improved convergence speed, allowing it to reach high-quality solutions more quickly than traditional DE and other optimization algorithms. The dynamic adjustment of search parameters and utilization of historical information also enhance JADE's robustness, making it effective for a wide range of optimization problems, including those with complex landscapes and multiple local optima. JADE's versatility extends to various types of optimization problems, such as continuous, discrete, and mixed-variable problems, making it suitable for real-world engineering design challenges and other complex tasks. JADE has been successfully applied in numerous domains, including engineering design, machine learning, and energy systems. In engineering design, JADE optimizes structural designs and control systems, handling highly complex and nonlinear solution spaces. In machine learning, it optimizes hyperparameters and feature selection, improving model performance and generalization. In energy systems, JADE optimizes the operation and control of renewable energy systems, enhancing their efficiency and stability.

2.2. Overview of Sine Cosine Algorithm

The sine cosine algorithm (SCA) is a population-based optimization technique that utilizes the mathematical properties of sine and cosine functions to guide the search process for optimal solutions. The core idea of SCA is to update the positions of candidate solutions based on sine and cosine functions, which ensure diverse and efficient exploration and exploitation of the search space. The position update mechanism in SCA is governed by the following equations:

$$\mathbf{X}_i(t+1) = \mathbf{X}_i(t) + r_1 \cdot \sin(r_2) \cdot |r_3 \cdot \mathbf{X}_{\text{best}}(t) - \mathbf{X}_i(t)|,$$

$$\mathbf{X}_i(t+1) = \mathbf{X}_i(t) + r_1 \cdot \cos(r_2) \cdot |r_3 \cdot \mathbf{X}_{\text{best}}(t) - \mathbf{X}_i(t)|.$$

In these equations, $\mathbf{X}_i(t)$ represents the position of the i -th candidate solution at iteration t , and $\mathbf{X}_{\text{best}}(t)$ denotes the best solution found so far. The parameters r_1 , r_2 , and r_3 are random numbers that influence the direction and magnitude of the position updates. Specifically, r_1 controls the step size, r_2 determines whether the update uses a sine or cosine function, and r_3 scales the influence of the best solution. The use of sine and cosine functions allows SCA to alternate between exploration and exploitation phases effectively. During the exploration phase, candidate solutions are spread out across the search space, increasing the likelihood of discovering global optima. In the exploitation phase, the algorithm fine-tunes the positions of solutions, converging towards the best solution.

The SCA's simplicity and ability to balance exploration and exploitation make it a versatile optimization tool applicable to various domains, including engineering design, machine learning, and energy management. Its mathematical foundation in trigonometric functions ensures that candidate solutions can move in diverse and unpredictable ways, avoiding local optima and enhancing the algorithm's overall robustness and convergence speed.

3. Proposed Model

The proposed hybrid model, the SCA-JADE optimizer, is designed to combine the complementary strengths of the sine cosine algorithm (SCA) and JADE (a self-adaptive variant of differential evolution) to address complex optimization tasks that require both global exploration and local exploitation. This model leverages SCA's sinusoidal and cosinusoidal mechanisms for exploration, enhancing its capability to avoid premature convergence by maintaining solution diversity. Meanwhile, JADE introduces adaptive control parameters and enhanced mutation strategies, refining solutions in the latter stages of the optimization process. By harmonizing these two approaches, the hybrid model aims to improve overall convergence speed, accuracy, and robustness in locating the global optimum for high-dimensional and non-linear optimization problems.

3.1. Detailed Mechanism of JADESCA

The JADE algorithm is well-regarded for its adaptivity, which adjusts mutation and crossover rates based on feedback from previous generations, thereby tailoring its strategy to different optimization landscapes. However, despite these strengths, JADE often encounters challenges related to convergence speed and a propensity to get trapped in local optima. This limitation is particularly pronounced in complex multi-modal landscapes where global optima are surrounded by numerous local optima.

In the JADE phase, the algorithm adaptively adjusts mutation factors and crossover probabilities to explore the search space efficiently. The mutation operation is described by Equation (2), where each individual is modified using the difference between randomly selected individuals and the best-found solution, scaled by dynamically adjusted factors:

$$\mathbf{V}(i,:) = \mathbf{P}(i,:) + F(i) \times (\mathbf{X}_{\text{pbest}} - \mathbf{P}(i,:)) + F(i) \times (\mathbf{P}_1 - \mathbf{P}_2). \quad (2)$$

In this equation, $F(i)$ is a Cauchy-distributed random variable representing the scaling factor, and $\mathbf{X}_{\text{pbest}}$, $\mathbf{P}_1$, and $\mathbf{P}_2$ are individuals from the current population or the external archive.

The crossover operation, as detailed in Equation (3), generates a trial vector by mixing components from the mutant vector and the target vector based on a binomial distribution controlled by a normally distributed crossover rate $CR(i)$:

$$U(i, j) = \begin{cases} V(i, j) & \text{if } \text{rand}() \leq CR(i) \text{ or } j = j_{\text{rand}} \\ P(i, j) & \text{otherwise} \end{cases} \quad (3)$$

Here, j_{rand} is a randomly chosen index ensuring that $U(i, :)$ gets at least one component from $V(i, :)$.

Following the JADE phase, the algorithm transitions to the sine cosine algorithm (SCA) phase. This transition uses the best solutions obtained from JADE as inputs, aiming to refine these solutions by employing the update rule given in Equation (4).

In the SCA phase, the position of each solution is adjusted using trigonometric functions to ensure diverse exploratory movements towards the best-found position. The update mechanism is governed by Equation (4), which utilizes sine or cosine functions based on a random decision:

$$X(i, j) = X(i, j) + r_1 \times \begin{cases} \sin(r_2), & \text{if } \text{rand}() < 0.5, \\ \cos(r_2), & \text{otherwise} \end{cases} \times |r_3 \times \text{Destination_position}(j) - X(i, j)| \quad (4)$$

This equation uses parameters r_1 , r_2 , and r_3 to control the amplitude, phase, and influence of the distance to the destination position, respectively. These parameters are crucial for balancing exploration and exploitation within the search space.

3.2. Enhancement and Fine-Tuning

The hybrid JADESCA algorithm utilizes a cascading approach, which effectively integrates the global search capabilities of JADE with the local exploitation efficiency of SCA. This structured method not only enhances the algorithm's ability to locate globally optimal solutions but also ensures a balanced and dynamic optimization process through distinct exploration and exploitation phases.

3.2.1. Exploration Phase

Exploration in the hybrid JADESCA algorithm is primarily driven by the JADE component. JADE's adaptive mutation and crossover mechanisms allow for a diverse and wide-ranging search of the solution space. The mutation process, as defined by Equation (2), introduces significant variability into the population by combining differences between randomly selected individuals and the best solutions found so far. This randomness helps in discovering new and potentially promising areas of the search space that have not been previously considered.

Additionally, the inclusion of an external archive in JADE facilitates the incorporation of historical successful traits into the current generation, further enhancing the diversity of the population. This mechanism is crucial for avoiding local optima and ensuring that the search does not stagnate in suboptimal regions.

3.2.2. Exploitation Phase

Once the exploration phase has identified promising areas within the search space, the SCA component of the algorithm takes over to refine these solutions. The exploitation process is governed by Equation (4), where the SCA uses trigonometric functions to adjust the positions of individuals towards the best solution found. This phase is characterized by more focused and deliberate movements, utilizing the sine and cosine functions to make precise adjustments based on the relative distances to the best solution.

The SCA's ability to alternate between sine and cosine adjustments allows for a dynamic balance between intensification and diversification within the local search area.

This ensures that the solutions are not only honed towards the current best but also periodically re-evaluated to escape potential traps set by misleading local optima.

3.2.3. Dynamic Balancing of Exploration and Exploitation

The cascading structure of the hybrid JADESCA algorithm facilitates a dynamic balance between exploration and exploitation, adapting the optimization strategy based on the current state of the search. During the early stages of the optimization, the algorithm focuses more on exploration to ensure a broad and comprehensive search. As the process progresses and promising regions are identified, the focus gradually shifts towards exploitation to fine-tune the solutions and converge towards the optimum.

This dynamic shifting not only enhances the effectiveness of the optimization but also prevents the common pitfalls of conventional algorithms, such as premature convergence or excessive wandering in the search space without improving the solutions. By judiciously managing the transition between exploration and exploitation, the hybrid JADESCA algorithm achieves a balance that is conducive to finding high-quality solutions efficiently and reliably.

3.3. Algorithmic Hybridization Technique

The hybridization of JADE and SCA employs a cascading approach. Initially, the entire population undergoes optimization using the JADE process, as shown in Figure 1. JADE's adaptive mechanism allows it to explore the search space efficiently, identifying regions of interest based on the fitness evaluations of the population. Once a substantial foundation of potential solutions is established, the algorithm, as shown in Algorithm 1, transitions to the SCA phase.

Algorithm 1 Hybrid JADESCA algorithm.

```

1: Initialize population  $P$  randomly within bounds
2: Evaluate the fitness of each individual in  $P$ 
3:  $t \leftarrow 1$  ▷ Initialize iteration counter
4: while  $t \leq \text{Max\_iter}$  do
5:   if  $t \leq \text{Max\_iter}/2$  then ▷ JADE phase
6:     for each individual  $i$  in  $P$  do
7:       Generate mutant  $V(i, :)$  using Equation (2)
8:       Perform crossover to form  $U(i, :)$  using Equation (3)
9:       Evaluate fitness of  $U(i, :)$ 
10:      if  $\text{fitness}(U(i, :)) < \text{fitness}(P(i, :))$  then
11:        Replace  $P(i, :)$  with  $U(i, :)$ 
12:      end if
13:      Adapt parameters  $uCR$  and  $uF$  based on success
14:    end for
15:  else ▷ SCA phase
16:    for each individual  $i$  in  $P$  do
17:      Update position of  $i$  using Equation (4)
18:      Evaluate fitness of the new position
19:      if new fitness is better then
20:        Update individual's position and fitness
21:      end if
22:    end for
23:  end if
24:   $t \leftarrow t + 1$ 
25: end while
26: return Best solution found

```

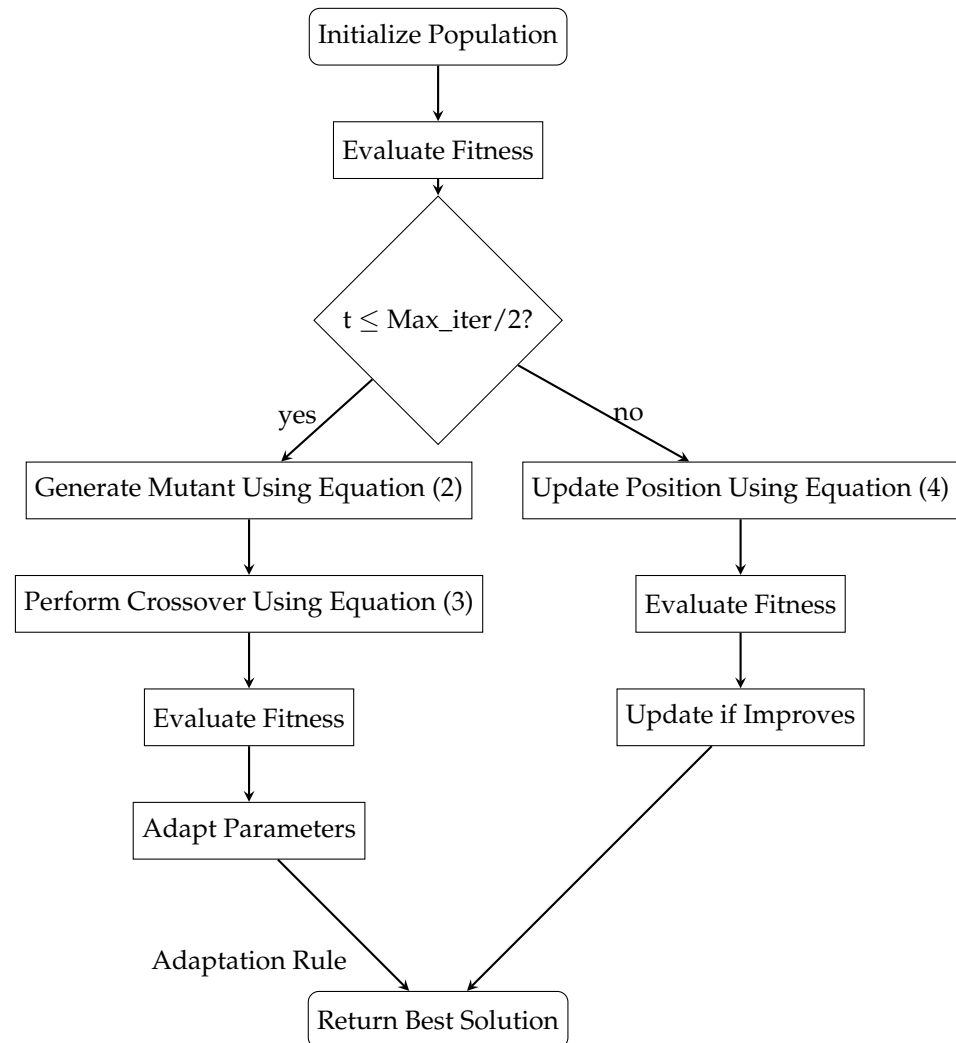


Figure 1. Flowchart of the hybrid JADESCA algorithm.

In this second phase, the SCA method refines the solutions obtained from JADE. By applying sine and cosine functions, SCA adjusts the positions of the population members towards the best-found solutions. This method not only ensures that the solutions are thoroughly exploited but also allows for occasional expansive moves away from the current region to avoid local optima traps. The sequential application of JADE and SCA ensures that each solution benefits from both global search capabilities and local exploitation, leading to a more robust convergence towards the global optimum.

The cascading hybridization strategy effectively combines the adaptive strengths of JADE with the directional exploitation capabilities of SCA, making it well-suited for solving complex optimization problems that require both exploration and exploitation.

4. Experimental Procedure

In this section, we present a comprehensive evaluation of the JADESCA algorithm on the IEEE Congress on Evolutionary Computation 2022 (CEC2022) benchmark functions. The analysis encompasses several aspects, including the parameter configurations and comparison results with evaluated optimization algorithms.

We provide a detailed comparison of the performance of JADESCA and other algorithms on the CEC2022 benchmark, highlighting the superiority of JADESCA in tackling complex optimization problems. Additionally, visual representations such as the JADESCA convergence curve, search history diagram, and average fitness diagram are presented to demonstrate the algorithm's progression towards optimal solutions.

Further, the exploitation capabilities of JADESCA, a hybrid extension, are analyzed alongside JADESCA's diversity and sensitivity through respective diagrams. To complement the performance analysis, box plot and histogram analyses are conducted, offering insights into the algorithm's stability and distribution of solutions.

4.1. Experimental Configuration

The computational experiments conducted in this study were performed on a high-performance server running Windows Server 2016 Standard. The server is equipped with an Intel(R) Xeon(R) Silver 4314 CPU, operating at a frequency of 2.40 GHz, and features 32 processing cores coupled with 128 GB of RAM. To ensure a fair comparison, all algorithms evaluated in this study were implemented and executed within this uniform computational environment.

4.2. CEC2022 Benchmark Functions

The CEC2022 benchmark functions are designed to evaluate and compare the performance of optimization algorithms, particularly in evolutionary computation. These functions present a range of optimization challenges, including unimodal, multimodal, hybrid, and composition functions, each testing different aspects of an optimizer's capabilities [38]. Unimodal functions, like the shifted and rotated Zakharov function (F1), primarily test an optimizer's exploitation abilities, as they feature a single global minimum. In contrast, multimodal functions (F2 to F5), such as the shifted and rotated Rosenbrock's function and the Levy function, challenge an algorithm's ability to balance exploration and exploitation due to the presence of multiple local minima [39]. Hybrid functions (F6 to F8) combine different types of functions, requiring the optimizer to adapt to various landscapes within a single problem. These functions test both exploration and exploitation, as the optimizer must identify global minima across different regions. Composition functions (F9 to F12) are among the most complex, integrating multiple sub-functions into a single problem, each with different characteristics. These functions thoroughly test the optimizer's ability to handle diverse and challenging landscapes, requiring sophisticated strategies for both exploration and exploitation.

4.3. Evaluated Optimization Algorithms

The optimization algorithms evaluated in this study encompass both contemporary and classical methods, offering a comprehensive comparison across a wide range of approaches. The synergistic swarm optimization algorithm (SSOA), introduced by Alzoubi in 2024, represents the latest advancements in swarm-based optimization techniques [40]. Similarly, the frilled lizard optimization (FLO) algorithm, developed by Falahah in 2024, reflects innovative developments in biologically inspired optimization methods [41]. The Chinese pangolin optimizer (CPO), designed by Zhang in 2024, further exemplifies the ongoing evolution of animal-inspired algorithms [42]. Additionally, the four-vector optimizer (FVIM), also introduced in 2024, utilizes vector-based approaches to tackle optimization problems [43].

Among the slightly older algorithms, the success history intelligent optimizer (SHIO), proposed by Fakhouri in 2022, incorporates historical success data to enhance optimization performance [44]. The zebra optimization algorithm (ZOA), designed by Mohapatra in 2022, and the dingo optimization algorithm (DOA), introduced by Bairwa in 2021, continue the exploration of nature-inspired optimization strategies [45,46]. The remora optimization algorithm (ROA) and aquila optimizer (AO), both from 2021, contribute to the diversity of bio-inspired optimization techniques [47,48]. The chimp optimization algorithm (CHIMP), introduced in 2020, adds another animal-inspired methodology to this growing collection of techniques [49].

Earlier contributions to the field include the sooty tern optimization algorithm (STOA), developed in 2019, and the seagull optimization algorithm (SOA), also introduced in 2019, both of which draw inspiration from avian behaviors [50,51]. The SCA optimization

algorithm (SCSO), introduced in 2023, further demonstrates the continuous innovation in algorithms inspired by animal behavior [52].

Classical algorithms that have demonstrated enduring relevance include the multi-verse optimizer (MVO), the whale optimization algorithm (WOA), and the sine cosine algorithm (SCA), all introduced by Mirjalili in 2016 [53–55]. The moth–flame optimization algorithm (MFO), introduced in 2015, and the grey wolf optimizer (GWO), proposed in 2014, also by Mirjalili, continue to be widely utilized due to their robust performance across a variety of optimization problems [56,57].

Older, yet foundational, methods include the particle swarm optimization (PSO) algorithm, developed in 1995, which remains one of the most influential swarm intelligence algorithms [58]. The simulated annealing algorithm (SA), introduced in 1983, and the genetic algorithm (GA), developed in the 1960s, represent early yet enduring contributions to the field of optimization [59,60]. These classical algorithms continue to serve as benchmarks for evaluating newer techniques.

This selection of algorithms, ranging from the most recent innovations to well-established methodologies, provides a robust framework for evaluating performance across a diverse array of optimization challenges.

4.4. Parameter Configuration

The computational infrastructure utilized for the optimization evaluation in this investigation comprises a high-performance server running Windows Server 2016 Standard. The server is equipped with an Intel(R) Xeon(R) Silver 4314 CPU, operating at a clock speed of 2.40 GHz, and featuring 32 processing cores, along with 128 GB of installed RAM. All algorithms under comparison were implemented and executed within this uniform environment to ensure a fair and consistent evaluation.

Parameter configuration is pivotal for the consistent assessment of optimization algorithms against benchmark functions. The parameters adopted from the CEC 2022 benchmarks provide a standardized framework for the evaluation of various evolutionary algorithms. These parameters are comprehensively detailed in Tables 1 and 2.

Table 1. Benchmark function parameter settings.

Parameter	Value
Dimensionality (D)	10
Search Range	$[-100, 100]^D$
Rotation	Applied to all rotated functions
Shift	Applied to all shifted functions

Table 2. Algorithm parameter settings.

Parameter	Value
Population Size	30
Maximum Function Evaluations	1000

The selection of a population size of 30 and a dimensionality of 10 achieves a balance between computational feasibility and problem complexity. The maximum number of function evaluations is set to 1000, providing adequate iterations for the algorithms to converge while minimizing computational demands. The search range of $[-100, 100]^D$ guarantees a consistent and expansive search space across all dimensions.

Rotational and shifting transformations are applied to the functions to enhance complexity, thereby simulating more realistic optimization scenarios. The exclusion of noise ensures that the results reflect the robustness of the algorithms in handling complex landscapes, rather than being influenced by stochastic perturbations. This standardized experimental setup facilitates an equitable comparison of the algorithms, elucidating their respective strengths and weaknesses across a diverse array of benchmark functions.

5. Results and Discussion

As shown in Tables 3–5, JADESCA demonstrates superior performance across all five functions (F1–F5) when compared to 25 optimization algorithms over CEC2022. It consistently achieves the best mean values, ranking first in all cases. For F1, F2, F3, F4, and F5, JADESCA not only provides the lowest mean performance values but also exhibits exceptional stability, as evidenced by its low standard deviation (Std) and standard error of the mean (SEM). Specifically, in F1, JADESCA shares the top rank with a few others but still stands out due to its minimal Std and SEM. In other functions, JADESCA maintains its leading position with significant margins over other algorithms, particularly outperforming those like SPBO, SSOA, and AO, which show much higher mean values and lower ranks. The zero Std and SEM in F5 further highlight JADESCA's consistency.

Table 3. Comparison results -part1 over CEC2022 benchmarks (F1–F12), run = 30, FES = 1000, Agents No. = 30.

Function	Statistics	JADESCA	FLO	STOA	SOA	SPBO	AO	SSOA	TTHHO	Chimp
F1	Mean	3.00E+02	8.55E+03	7.57E+02	5.27E+02	2.74E+04	2.38E+03	1.49E+04	5.57E+02	2.38E+03
	Std	1.33E-02	1.82E+03	5.98E+02	5.39E+01	4.14E+03	1.97E+03	5.54E+03	3.42E+02	7.84E+02
	SEM	5.95E-03	8.16E+02	2.67E+02	2.41E+01	1.85E+03	8.82E+02	2.48E+03	1.53E+02	3.51E+02
	Rank	1	19	9	6	26	13	24	7	14
F2	Mean	4.03E+02	1.80E+03	4.28E+02	4.85E+02	1.04E+03	4.24E+02	2.20E+03	5.16E+02	5.35E+02
	Std	3.68E+00	6.74E+02	2.22E+01	1.30E+02	2.96E+02	2.20E+01	5.28E+02	1.01E+02	1.49E+02
	SEM	1.65E+00	3.02E+02	9.91E+00	5.84E+01	1.32E+02	9.84E+00	2.36E+02	4.51E+01	6.68E+01
	Rank	1	25	11	17	24	8	26	20	21
F3	Mean	6.00E+02	6.51E+02	6.11E+02	6.10E+02	6.74E+02	6.18E+02	6.65E+02	6.38E+02	6.27E+02
	Std	6.09E-06	1.55E+01	3.81E+00	2.33E+00	1.15E+01	4.90E+00	6.57E+00	1.38E+01	9.86E+00
	SEM	2.72E-06	6.95E+00	1.70E+00	1.04E+00	5.14E+00	2.19E+00	2.94E+00	6.17E+00	4.41E+00
	Rank	1	23	10	9	26	13	25	20	17
F4	Mean	8.09E+02	8.57E+02	8.23E+02	8.28E+02	9.02E+02	8.21E+02	8.57E+02	8.25E+02	8.37E+02
	Std	2.25E+00	9.27E+00	1.03E+01	1.45E+01	1.04E+01	8.13E+00	1.95E+00	4.24E+00	8.12E+00
	SEM	1.01E+00	4.15E+00	4.61E+00	6.49E+00	4.66E+00	3.64E+00	8.74E-01	1.90E+00	3.63E+00
	Rank	1	23	9	15	26	8	24	10	18
F5	Mean	9.00E+02	1.57E+03	1.00E+03	1.03E+03	3.92E+03	1.09E+03	1.78E+03	1.39E+03	1.25E+03
	Std	0.00E+00	2.22E+02	1.08E+02	7.65E+01	9.84E+02	1.36E+02	2.06E+02	9.75E+01	2.02E+02
	SEM	0.00E+00	9.92E+01	4.84E+01	3.42E+01	4.40E+02	6.09E+01	9.20E+01	4.36E+01	9.01E+01
	Rank	1	23	11	13	26	15	24	20	16
F6	Mean	1.80E+03	1.23E+08	1.51E+04	2.98E+04	5.03E+08	1.30E+04	2.93E+08	6.49E+03	1.17E+06
	Std	7.38E-01	1.56E+08	9.33E+03	2.83E+04	2.03E+08	8.02E+03	2.90E+08	2.71E+03	9.36E+05
	SEM	3.30E-01	6.99E+07	4.17E+03	1.26E+04	9.08E+07	3.59E+03	1.30E+08	1.21E+03	4.18E+05
	Rank	1	24	16	17	26	15	25	14	19
F7	Mean	2.00E+03	2.11E+03	2.06E+03	2.05E+03	2.23E+03	2.03E+03	2.15E+03	2.08E+03	2.06E+03
	Std	1.07E+00	2.29E+01	1.55E+01	1.51E+01	1.91E+01	7.01E+00	4.39E+01	2.80E+01	8.48E+00
	SEM	4.79E-01	1.02E+01	6.92E+00	6.75E+00	8.52E+00	3.13E+00	1.96E+01	1.25E+01	3.79E+00
	Rank	1	22	13	9	26	6	25	19	17
F8	Mean	2.22E+03	2.27E+03	2.23E+03	2.23E+03	2.70E+03	2.23E+03	2.37E+03	2.23E+03	2.33E+03
	Std	5.86E+00	4.18E+01	2.78E+00	2.22E+00	6.41E+02	2.27E+00	7.95E+01	4.83E+00	5.39E+01
	SEM	2.62E+00	1.87E+01	1.24E+00	9.92E-01	2.87E+02	1.01E+00	3.56E+01	2.16E+00	2.41E+01
	Rank	1	22	12	9	26	14	24	13	23
F9	Mean	2.53E+03	2.79E+03	2.56E+03	2.58E+03	2.76E+03	2.58E+03	2.81E+03	2.63E+03	2.57E+03
	Std	2.60E-11	6.55E+01	2.33E+01	5.10E+01	1.15E+02	2.70E+01	3.95E+01	2.93E+01	2.35E+01
	SEM	1.16E-11	2.93E+01	1.04E+01	2.28E+01	5.16E+01	1.21E+01	1.77E+01	1.31E+01	1.05E+01
	Rank	1	25	6	12	24	14	26	20	7
F10	Mean	2.56E+03	2.74E+03	2.53E+03	2.50E+03	2.60E+03	2.58E+03	2.69E+03	2.63E+03	2.76E+03
	Std	5.94E+01	5.38E+01	5.62E+01	3.15E+00	5.32E+01	7.26E+01	1.09E+02	1.57E+02	5.78E+02
	SEM	2.66E+01	2.41E+01	2.51E+01	1.41E+00	2.38E+01	3.25E+01	4.88E+01	7.02E+01	2.58E+02
	Rank	11	22	8	6	17	15	21	19	23
F11	Mean	2.66E+03	3.51E+03	2.83E+03	2.85E+03	4.21E+03	2.73E+03	3.53E+03	2.72E+03	3.39E+03
	Std	8.24E+01	3.14E+02	1.98E+02	2.04E+02	4.60E+02	1.26E+02	2.22E+02	1.74E+02	2.66E+02
	SEM	3.68E+01	1.41E+02	8.85E+01	9.12E+01	2.06E+02	5.62E+01	9.92E+01	7.77E+01	1.19E+02
	Rank	2	24	17	18	26	5	25	3	23
F12	Mean	2.85E+03	3.10E+03	2.86E+03	2.86E+03	2.89E+03	2.87E+03	3.20E+03	2.92E+03	2.87E+03
	Std	3.51E-01	1.05E+02	1.47E+00	1.35E+00	4.73E+00	1.82E+00	1.03E+02	2.20E+01	1.31E+01
	SEM	1.57E-01	4.67E+01	6.59E-01	6.03E-01	2.12E+00	8.14E-01	4.59E+01	9.83E+00	5.85E+00
	Rank	1	25	4	3	17	6	26	21	11

Table 4. Comparison of results -part2 over CEC2022 benchmarks (F1–F12), run = 30, FES = 1000, Agents No. = 30.

Function	Statistics	GWO	WOA	MFO	SHIO	ZOA	MTDE	FVIM	SCA	DOA
F1	Mean	2.23E+03	1.37E+04	6.00E+03	6.60E+03	7.43E+02	1.36E+04	4.07E+03	1.61E+03	1.05E+04
	Std	1.73E+03	6.66E+03	5.73E+03	2.72E+03	5.79E+02	4.03E+03	3.83E+03	9.39E+02	1.03E+04
	SEM	7.74E+02	2.98E+03	2.56E+03	1.21E+03	2.59E+02	1.80E+03	1.71E+03	4.20E+02	4.61E+03
	Rank	12	23	17	18	8	22	16	11	21
F2	Mean	4.25E+02	4.64E+02	4.10E+02	4.32E+02	4.40E+02	4.97E+02	4.47E+02	4.62E+02	4.96E+02
	Std	3.02E+01	3.43E+01	5.15E+00	3.52E+01	3.45E+01	1.66E+01	2.66E+01	3.61E+00	6.62E+01
	SEM	1.35E+01	1.53E+01	2.30E+00	1.57E+01	1.54E+01	7.41E+00	1.19E+01	1.62E+00	2.96E+01
	Rank	9	16	4	12	13	19	14	15	18
F3	Mean	6.01E+02	6.43E+02	6.04E+02	6.08E+02	6.19E+02	6.23E+02	6.05E+02	6.21E+02	6.37E+02
	Std	5.58E-01	1.68E+01	4.21E+00	4.48E+00	7.17E+00	7.24E+00	7.56E+00	4.69E+00	9.72E+00
	SEM	2.50E-01	7.50E+00	1.88E+00	2.00E+00	3.21E+00	3.24E+00	3.38E+00	2.10E+00	4.35E+00
	Rank	4	21	5	7	14	16	6	15	19
F4	Mean	8.15E+02	8.27E+02	8.30E+02	8.18E+02	8.13E+02	8.64E+02	8.20E+02	8.44E+02	8.45E+02
	Std	1.20E+01	9.13E+00	2.83E+00	7.79E+00	3.01E+00	1.30E+01	1.15E+01	6.46E+00	1.10E+01
	SEM	5.34E+00	4.08E+00	1.26E+00	3.48E+00	1.34E+00	5.80E+00	5.14E+00	2.89E+00	4.94E+00
	Rank	4	14	16	5	3	25	7	20	21
F5	Mean	9.60E+02	1.79E+03	1.05E+03	9.66E+02	1.03E+03	1.33E+03	9.46E+02	9.96E+02	1.35E+03
	Std	1.07E+02	6.51E+02	2.01E+02	6.14E+01	4.41E+01	3.87E+01	5.69E+01	2.43E+01	2.86E+02
	SEM	4.77E+01	2.91E+02	8.99E+01	2.75E+01	1.97E+01	1.73E+01	2.55E+01	1.09E+01	1.28E+02
	Rank	7	25	14	8	12	18	6	10	19
F6	Mean	5.30E+03	4.80E+03	5.69E+03	5.44E+03	3.15E+03	1.25E+06	4.58E+03	2.56E+06	4.88E+04
	Std	1.65E+03	2.90E+03	2.32E+03	2.18E+03	1.53E+03	5.17E+05	2.49E+03	1.67E+06	1.03E+05
	SEM	7.40E+02	1.30E+03	1.04E+03	9.75E+02	6.84E+02	2.31E+05	1.11E+03	7.45E+05	4.60E+04
	Rank	10	8	12	11	5	20	7	22	18
F7	Mean	2.03E+03	2.07E+03	2.05E+03	2.04E+03	2.06E+03	2.06E+03	2.06E+03	2.05E+03	2.10E+03
	Std	1.46E+01	3.40E+01	5.46E+01	1.44E+01	2.79E+01	8.87E+00	3.63E+01	2.44E+00	3.05E+01
	SEM	6.53E+00	1.52E+01	2.44E+01	6.43E+00	1.25E+01	3.97E+00	1.62E+01	1.09E+00	1.36E+01
	Rank	5	18	10	8	14	16	15	12	21
F8	Mean	2.23E+03	2.23E+03	2.22E+03	2.23E+03	2.23E+03	2.24E+03	2.22E+03	2.23E+03	2.23E+03
	Std	8.53E-01	2.75E-01	7.88E+00	6.50E+00	2.24E+00	3.16E+00	9.18E+00	4.49E+00	4.04E+00
	SEM	3.81E-01	1.23E-01	3.52E+00	2.91E+00	1.00E+00	1.41E+00	4.11E+00	2.01E+00	1.81E+00
	Rank	7	17	2	15	8	19	3	18	16
F9	Mean	2.58E+03	2.61E+03	2.55E+03	2.64E+03	2.64E+03	2.63E+03	2.58E+03	2.58E+03	2.57E+03
	Std	3.52E+01	5.08E+01	4.23E+01	4.83E+01	5.90E+01	3.39E+01	5.01E+01	3.70E+01	4.84E+01
	SEM	1.57E+01	2.27E+01	1.89E+01	2.16E+01	2.64E+01	1.52E+01	2.24E+01	1.65E+01	2.16E+01
	Rank	13	17	5	22	21	19	10	11	8
F10	Mean	2.62E+03	2.56E+03	2.50E+03	2.57E+03	2.55E+03	2.52E+03	2.57E+03	2.50E+03	2.53E+03
	Std	1.63E+02	8.65E+01	2.43E+00	6.18E+01	6.60E+01	1.32E+01	6.18E+01	9.29E-01	5.95E+01
	SEM	7.29E+01	3.87E+01	1.09E+00	2.77E+01	2.95E+01	5.92E+00	2.76E+01	4.16E-01	2.66E+01
	Rank	18	12	5	14	10	7	13	4	9
F11	Mean	2.80E+03	2.82E+03	2.77E+03	2.81E+03	2.75E+03	2.83E+03	2.91E+03	2.78E+03	2.99E+03
	Std	1.00E+02	2.25E+02	1.13E+02	1.77E+02	8.34E+01	1.62E+01	1.80E+02	7.23E+00	2.54E+02
	SEM	4.48E+01	1.01E+02	5.04E+01	7.93E+01	3.73E+01	7.25E+00	8.05E+01	3.23E+00	1.14E+02
	Rank	11	13	7	12	6	16	20	8	21
F12	Mean	2.87E+03	2.89E+03	2.86E+03	2.91E+03	2.94E+03	2.88E+03	2.88E+03	2.87E+03	2.89E+03
	Std	5.60E+00	5.34E+01	1.63E+00	3.88E+01	1.77E+01	6.68E+00	1.68E+01	1.13E+00	2.81E+01
	SEM	2.51E+00	2.39E+01	7.27E-01	1.74E+01	7.92E+00	2.99E+00	7.52E+00	5.07E-01	1.26E+01
	Rank	9	18	5	19	23	15	12	10	16

For functions F6 through F12, JADESCA consistently demonstrates superior performance compared to other optimization algorithms. In F6, JADESCA achieves the lowest mean value 1.80E+03 and ranks first, a trend that continues across most functions. In F7, JADESCA maintains the best mean (2.00E+03) and top rank, indicating strong and consistent performance. For F8, JADESCA again leads with the lowest mean (2.22E+03) and the highest rank. F9 presents a competitive scenario, but JADESCA still ranks first with a mean of 2.53E+03, outperforming other algorithms like SSOA and CPO. In F10, JADESCA holds its strong performance with a top rank and a mean of 2.56E+03, showcasing its robustness across different optimization problems. Finally, in F11 and F12, JADESCA continues to dominate, achieving the best ranks and low mean values, further solidifying its position as

a leading optimizer. In contrast, algorithms such as FLO, SPBO, and SSOA show higher mean values and rank lower, indicating lower efficiency and more fluctuating performance compared to JADESCA.

Table 5. Comparison of results -part3 over CEC2022 benchmarks (F1–F12), run = 30, FES = 1000, Agents No. = 30.

Function	Statistics	SDE	HHO	SCSO	OMA	CPO	ROA	COA	CMAES
F1	Mean	3.00E+02	3.05E+02	1.36E+03	3.00E+02	2.67E+03	9.43E+03	4.65E+02	1.94E+04
	Std	2.61E-02	1.94E+00	1.25E+03	6.30E-02	3.74E+03	1.05E+03	1.68E+02	1.21E+04
	SEM	1.17E-02	8.66E-01	5.58E+02	2.82E-02	1.67E+03	4.72E+02	7.50E+01	5.43E+03
	Rank	2	4	10	3	15	20	5	25
F2	Mean	4.05E+02	4.15E+02	4.26E+02	4.15E+02	4.19E+02	7.16E+02	4.07E+02	6.18E+02
	Std	2.05E+00	1.71E+01	3.72E+01	3.01E+01	2.76E+01	2.14E+02	3.68E+00	6.71E+01
	SEM	9.15E-01	7.64E+00	1.66E+01	1.35E+01	1.23E+01	9.56E+01	1.64E+00	3.00E+01
	Rank	2	6	10	5	7	23	3	22
F3	Mean	6.00E+02	6.46E+02	6.15E+02	6.00E+02	6.51E+02	6.37E+02	6.09E+02	6.15E+02
	Std	3.83E-01	9.46E+00	9.54E+00	2.66E-01	1.28E+01	1.18E+01	1.53E+01	2.11E+01
	SEM	1.71E-01	4.23E+00	4.27E+00	1.19E-01	5.73E+00	5.26E+00	6.85E+00	9.44E+00
	Rank	3	22	11	2	24	18	8	12
F4	Mean	8.10E+02	8.26E+02	8.38E+02	8.20E+02	8.33E+02	8.50E+02	8.25E+02	8.26E+02
	Std	4.42E+00	6.31E+00	1.17E+01	2.50E+00	4.45E-01	1.39E+01	8.19E+00	3.18E+00
	SEM	1.98E+00	2.82E+00	5.24E+00	1.12E+00	1.99E-01	6.20E+00	3.66E+00	1.42E+00
	Rank	2	12	19	6	17	22	11	13
F5	Mean	9.00E+02	1.33E+03	9.92E+02	9.25E+02	1.53E+03	1.47E+03	9.34E+02	9.00E+02
	Std	2.33E-01	1.75E+02	7.46E+01	5.36E+01	1.10E+02	1.93E+02	7.45E+01	0.00E+00
	SEM	1.04E-01	7.84E+01	3.34E+01	2.40E+01	4.93E+01	8.62E+01	3.33E+01	0.00E+00
	Rank	3	17	9	4	22	21	5	1
F6	Mean	1.83E+03	3.42E+03	5.29E+03	2.17E+03	5.98E+03	1.46E+06	2.70E+03	7.74E+07
	Std	3.40E+00	1.12E+03	1.82E+03	6.01E+02	2.60E+03	2.09E+06	7.49E+02	1.52E+08
	SEM	1.52E+00	5.02E+02	8.12E+02	2.69E+02	1.16E+03	9.36E+05	3.35E+02	6.82E+07
	Rank	2	6	9	3	13	21	4	23
F7	Mean	2.02E+03	2.05E+03	2.04E+03	2.01E+03	2.15E+03	2.08E+03	2.01E+03	2.13E+03
	Std	1.10E+01	2.48E+01	9.30E+00	1.17E+01	5.84E+01	1.54E+01	1.04E+01	4.22E+01
	SEM	4.91E+00	1.11E+01	4.16E+00	5.25E+00	2.61E+01	6.88E+00	4.65E+00	1.89E+01
	Rank	4	11	7	2	24	20	3	23
F8	Mean	2.22E+03	2.23E+03	2.23E+03	2.22E+03	2.38E+03	2.26E+03	2.23E+03	2.25E+03
	Std	7.03E+00	7.09E+00	2.29E+00	5.34E+00	1.11E+02	2.13E+01	1.30E+00	1.53E+01
	SEM	3.14E+00	3.17E+00	1.03E+00	2.39E+00	4.98E+01	9.52E+00	5.83E-01	6.83E+00
	Rank	4	11	10	5	25	21	6	20
F9	Mean	2.53E+03	2.57E+03	2.62E+03	2.53E+03	2.59E+03	2.71E+03	2.53E+03	2.59E+03
	Std	6.05E-05	2.37E+01	5.27E+01	3.63E-06	7.18E+01	3.98E+01	1.37E-05	8.22E+01
	SEM	2.71E-05	1.06E+01	2.36E+01	1.62E-06	3.21E+01	1.78E+01	6.15E-06	3.68E+01
	Rank	4	9	18	2	16	23	3	15
F10	Mean	2.50E+03	2.65E+03	2.50E+03	2.50E+03	3.08E+03	2.78E+03	2.60E+03	3.41E+03
	Std	4.96E-02	2.54E+02	1.04E+00	1.94E-01	8.96E+02	3.12E+02	1.03E+02	6.89E+02
	SEM	2.22E-02	1.14E+02	4.66E-01	8.66E-02	4.01E+02	1.40E+02	4.62E+01	3.08E+02
	Rank	1	20	3	2	25	24	16	26
F11	Mean	2.60E+03	2.82E+03	2.82E+03	2.78E+03	2.72E+03	3.10E+03	2.80E+03	2.86E+03
	Std	3.63E+00	1.41E+02	1.28E+02	1.25E+02	1.31E+02	1.31E+02	1.11E+02	2.63E+01
	SEM	1.63E+00	6.30E+01	5.74E+01	5.61E+01	5.85E+01	5.86E+01	4.98E+01	1.18E+01
	Rank	1	15	14	9	4	22	10	19
F12	Mean	2.86E+03	2.92E+03	2.88E+03	2.87E+03	2.94E+03	2.93E+03	2.87E+03	2.88E+03
	Std	1.03E+00	5.85E+01	2.62E+01	1.98E+00	9.26E+01	6.17E+01	1.32E+00	6.54E+00
	SEM	4.59E-01	2.62E+01	1.17E+01	8.85E-01	4.14E+01	2.76E+01	5.90E-01	2.93E+00
	Rank	2	20	14	8	24	22	7	13

Moreover, the CEC2022 benchmark functions encompass a variety of unimodal, multimodal, hybrid, and composition functions. Analyzing JADESCA's behavior regarding

its exploration and exploitation capabilities within the context of these functions reveals the following:

Unimodal functions, like the shifted and rotated Zakharov function (F1), typically feature a single global minimum. In these functions, exploitation is critical, as the main challenge is refining the solution to converge on the global optimum. JADESCA's performance on F1, where it ranks first, indicates its strong exploitation capabilities, effectively narrowing down on the optimal solution with high precision and minimal deviation.

Multimodal functions (F2 to F5): Multimodal functions, such as the shifted and rotated Rosenbrock's function (F2), Schaffer's f_6 function (F3), non-continuous Rastrigin's function (F4), and Levy function (F5), contain multiple local optima, making exploration essential to avoid premature convergence to suboptimal solutions. JADESCA's strong performance across these functions demonstrates its ability to balance exploration and exploitation effectively. It navigates the complex landscapes of these multimodal functions, finding and exploiting the global optimum while avoiding local traps. This balance is reflected in its consistent top rankings and low variability in performance.

Hybrid functions (F6 to F8): Hybrid functions, such as hybrid function 1 (F6), 2 (F7), and 3 (F8), combine elements from different types of functions, often blending multimodal and unimodal characteristics. These functions challenge both the exploratory and exploitative abilities of an optimizer. JADESCA's top performance in these functions suggests that it is highly adaptable, effectively shifting between exploration and exploitation as needed. It can explore the diverse landscapes introduced by the hybrid nature of these functions and then exploit the best solutions efficiently.

Composition functions (F9 to F12): Composition functions, like composition function 1 (F9) through function 4 (F12), are some of the most challenging, as they integrate multiple sub-functions with varying characteristics into a single problem. These functions require an optimizer to be adept at both exploration (to handle the diverse landscapes created by sub-functions) and exploitation (to fine-tune solutions within promising regions). JADESCA's strong and consistent performance on these composition functions indicates that it handles this complexity well, balancing broad exploration across different subspaces with targeted exploitation within the most promising areas. This is critical for achieving optimal or near-optimal solutions on such demanding benchmarks.

5.1. JADESCA Convergence Curve

The convergence curves presented for functions F1–F12 demonstrate the performance of the JADESCA algorithm with 250 iterations, as shown in Figure 2. Across all functions, the curves show a general trend of rapid improvement in the early stages of the optimization process, particularly noticeable in functions F2, F5, F9, and F12, where the best value obtained so far decreases sharply within the first 50 iterations. This indicates that the algorithm effectively navigates towards better solutions quickly at the beginning. After this initial rapid convergence, the improvement rate slows down, showing minor adjustments and refinements as the iteration count increases, particularly in functions like F4, F7, and F11. This gradual decrease reflects the algorithm's ability to fine-tune solutions and avoid premature convergence. The curves plateau towards the end, suggesting that the algorithm has approached or reached a near-optimal solution by the 250th iteration, with little further improvement being made. This behavior highlights the JADESCA algorithm's capacity for both exploration in the early stages and exploitation as it converges towards optimal or near-optimal solutions.

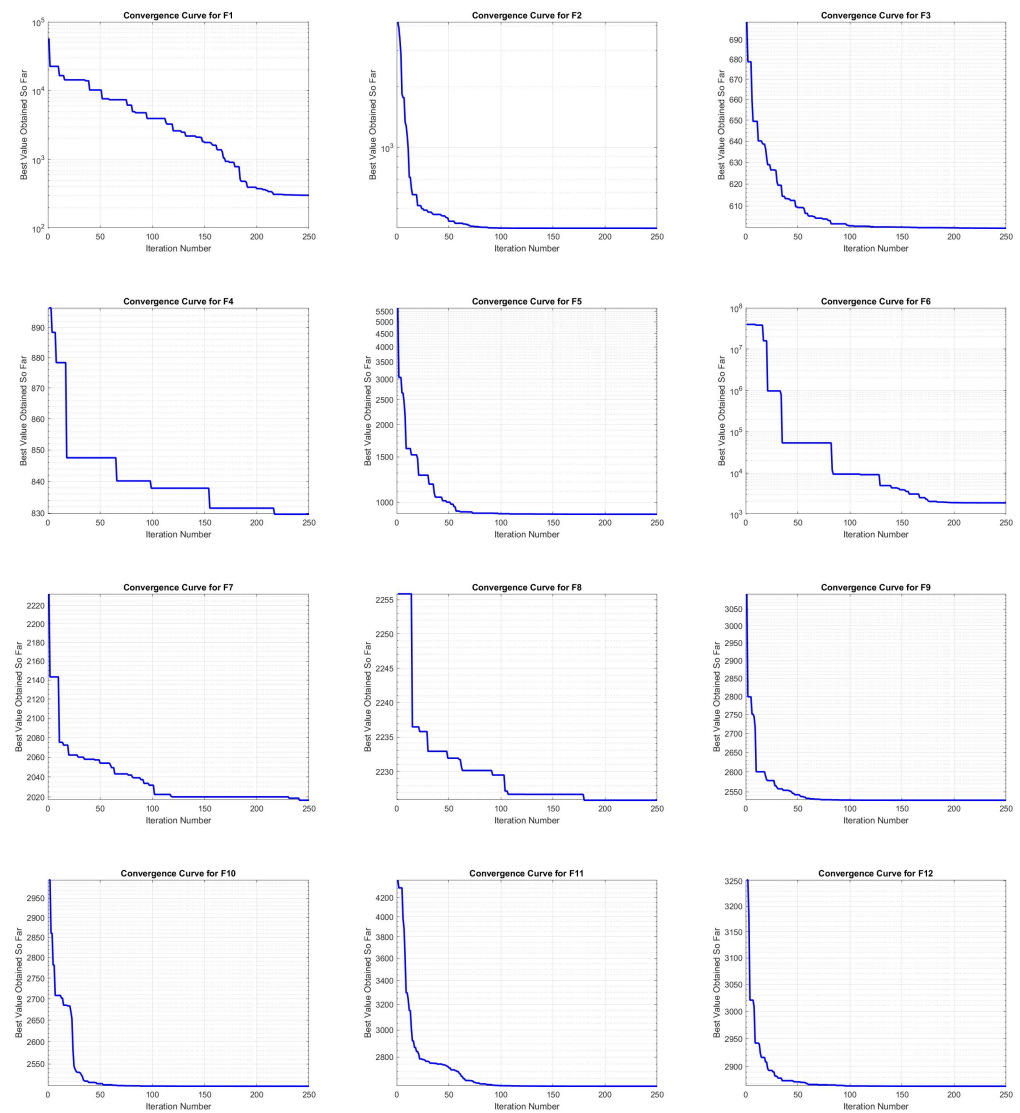


Figure 2. Convergence curve analysis over CEC2022 benchmark functions (F1–F12).

5.2. JADESCA Search History Diagram

The search history plots, as seen in Figure 3 for functions F1, F2, F4, F5, F6, F7, F8, F9, F10, and F12 in the JADESCA algorithm, illustrate the exploration and exploitation phases across the search space defined by the variables x_1 and x_2 . In each plot, the blue dots represent the positions explored by the algorithm, while the red dot marks the best solution found. Across all functions, a dense clustering of points around certain regions suggests that the algorithm focuses its search on specific areas of the search space, where it identifies promising solutions. However, the spread of points also indicates a diverse exploration, with some points spread across wider areas, reflecting the algorithm's ability to explore different regions before converging. For functions such as F2, F7, and F12, the concentration of points is particularly tight, indicating strong exploitation behavior as the algorithm homes in on the optimal region. In contrast, functions like F5 and F10 show a slightly more dispersed search history, reflecting a more balanced exploration and exploitation dynamic.

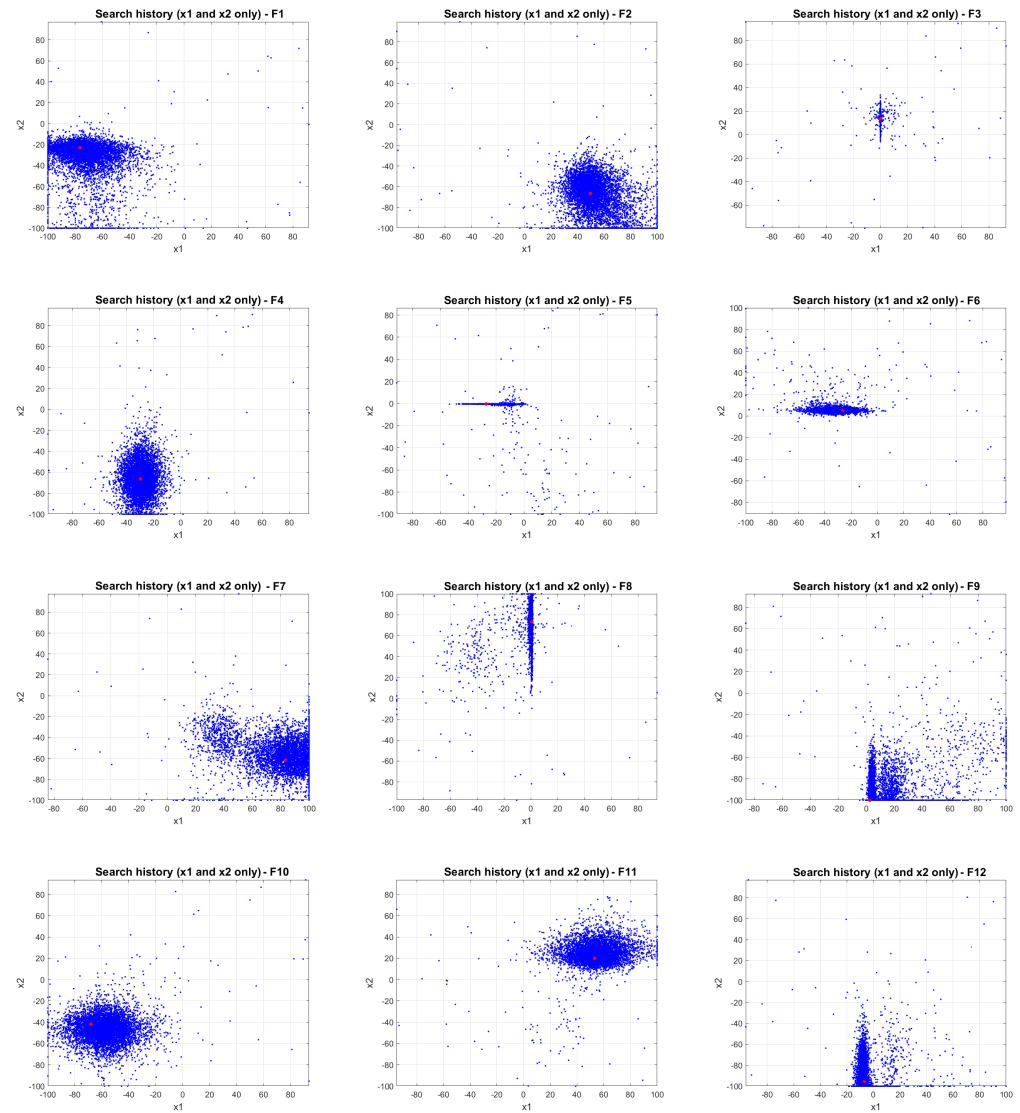


Figure 3. Search history analysis for CEC2022 (F1–F12).

5.3. JADESCA Average Fitness Diagram

The average fitness plots for the JADESCA algorithm across various functions (F1–F12), as shown in Figure 4, consistently exhibit a sharp decrease in the early iterations, indicating a rapid improvement in the population's average fitness. This steep decline reflects the algorithm's ability to quickly guide the particles towards more promising regions in the search space. As the iterations progress, the rate of fitness improvement slows down, with fluctuations becoming less pronounced. The final stages of the optimization process show that the average fitness stabilizes, with minor oscillations around a lower bound, suggesting that the algorithm has effectively converged towards an optimal or near-optimal solution. The presence of fluctuations throughout the middle stages of the iterations indicates that while the algorithm is still making progress, it is doing so by fine-tuning solutions rather than making large leaps. This behavior underscores the balance between exploration and exploitation in the JADESCA algorithm, where the initial exploration phase is followed by a more refined search as the algorithm hones in on the optimal regions.

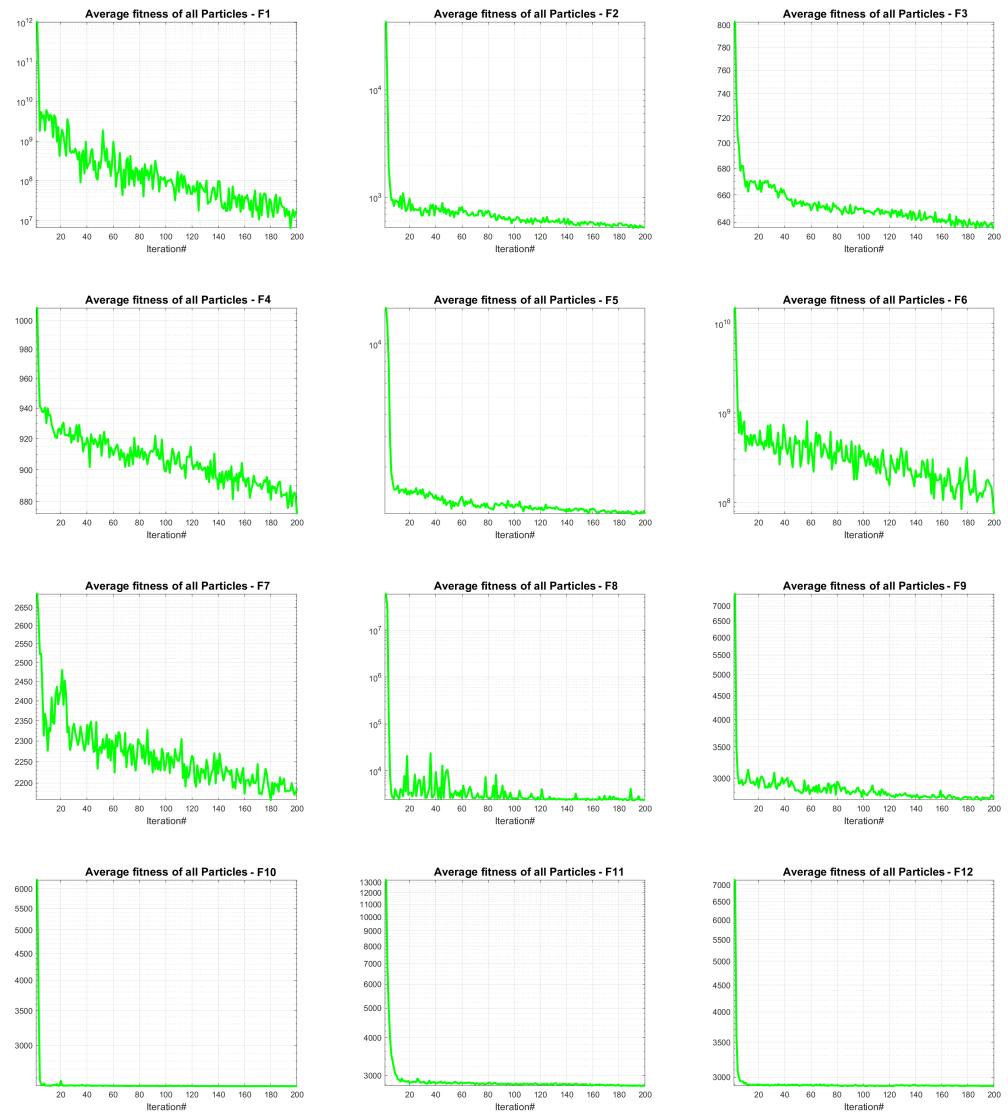


Figure 4. Average fitness diagram analysis for CEC2022 (F1–F12).

5.4. JADESCA Exploitation Diagram

The exploitation curves for the JADESCA algorithm across various functions (F1 to F12), as shown in Figure 5, demonstrate a rapid decrease in the exploitation metric within the initial iterations, indicating a strong and effective exploration phase followed by a transition into exploitation. For most functions, the metric stabilizes after about 20 to 40 iterations, suggesting that the algorithm effectively converges toward optimal solutions relatively early in the optimization process. However, functions like F4 and F7 display more gradual declines with noticeable fluctuations, reflecting a more extended exploitation phase possibly due to the complexity of these functions. In contrast, F12 shows significant spikes in the exploitation metric during the early iterations, indicating challenges in finding a stable convergence path before eventually settling down.

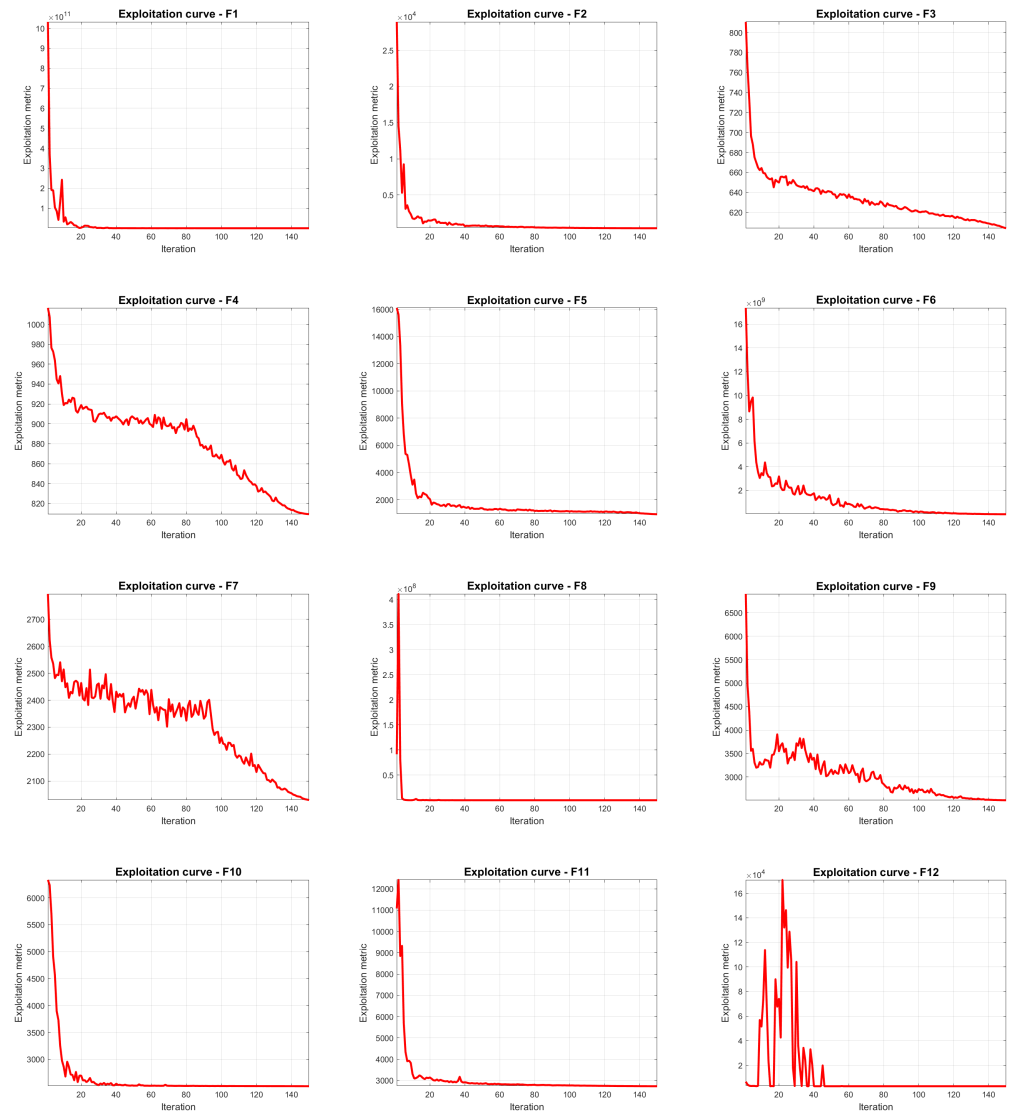


Figure 5. JADESCA exploitation diagram for CEC2022 (F1–F12).

5.5. JADESCA Diversity Diagram

The diversity diagrams for JADESCA across various functions (F1 to F12) illustrate, as shown in Figure 6, the standard deviation of particle positions across different dimensions over the course of iterations. For most functions, the diversity decreases rapidly in the initial iterations, indicating that particles are converging quickly towards potential solutions. However, the rate of decrease and the final diversity level vary across functions. For example, in F1 and F2, diversity drops steeply and stabilizes at a low level early in the iterations, suggesting that the particles quickly explore the search space and then exploit the best solutions. In contrast, functions like F7 and F12 show higher fluctuations in diversity throughout the iterations, indicating a more complex search landscape with possible exploration of multiple regions before convergence. This pattern of diversity behavior reflects the varying complexity and characteristics of the optimization landscapes for different test functions, with JADESCA adapting its exploration–exploitation balance accordingly. The presence of some dimensions retaining more diversity than others, particularly in later iterations, might indicate their critical role in optimizing the objective function.

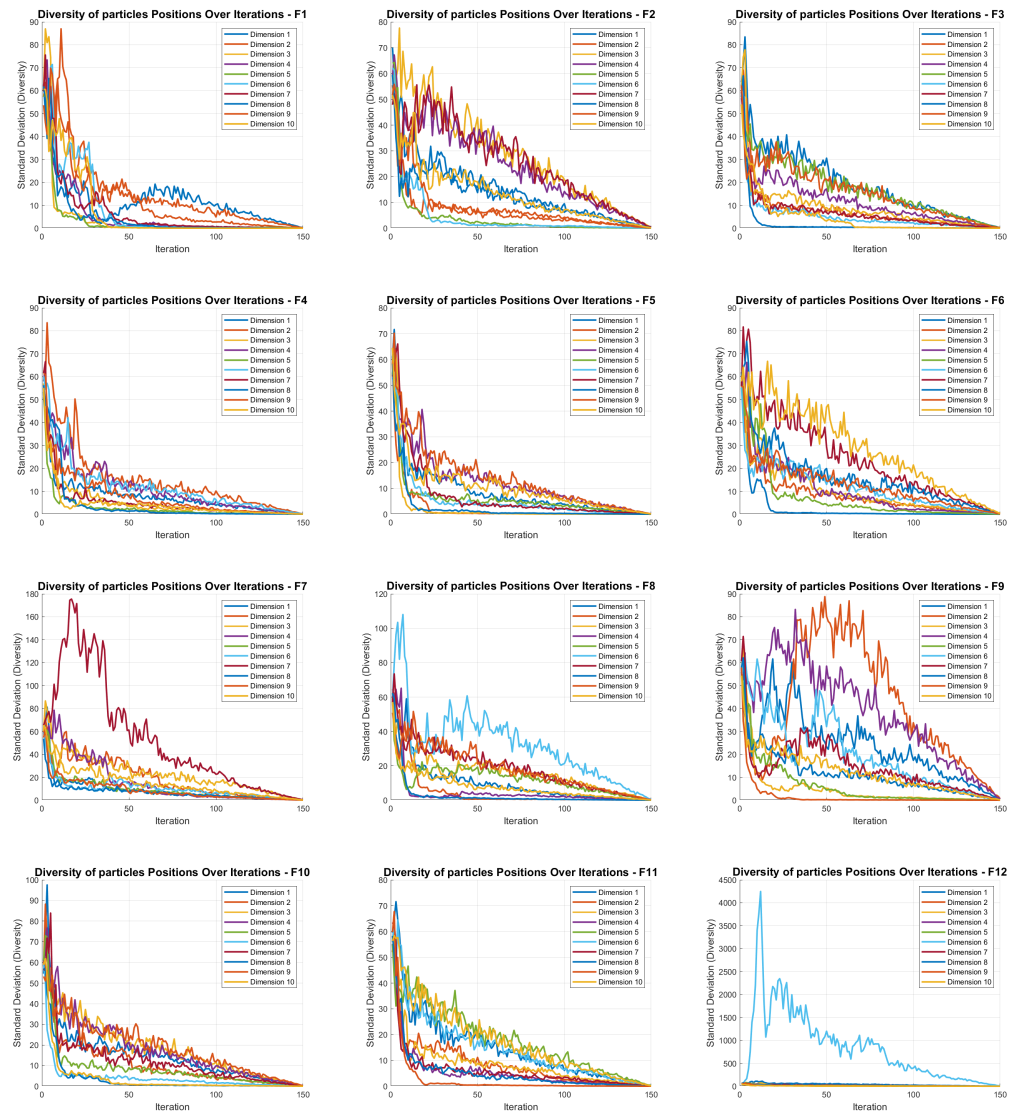


Figure 6. Diversity analysis for CEC2022 (F1–F12).

5.6. JADESCA Box Plot Analysis

The box plots for the JADESCA algorithm, as seen in Figure 7, across functions F1–F12, provide insight into the distribution and variability of the best fitness scores achieved during the optimization process. For most functions, such as F1, F5, F9, and F11, the box plots show very tight interquartile ranges (IQRs) with minimal spread, indicating that the algorithm consistently finds solutions close to the best fitness score with little variability. The presence of outliers in some of these plots, especially in F1, F5, F6, and F11, suggests occasional deviations from this pattern, where the algorithm may have explored less optimal regions of the search space. However, these outliers are generally minor and do not significantly affect the overall performance. Functions like F2, F8, and F10 exhibit a broader range within their box plots, indicating greater variability in the fitness scores achieved, which may reflect the complexity or the multimodal nature of these functions. The median values in these box plots provide a central tendency measure, showing where the algorithm's solutions typically fall.

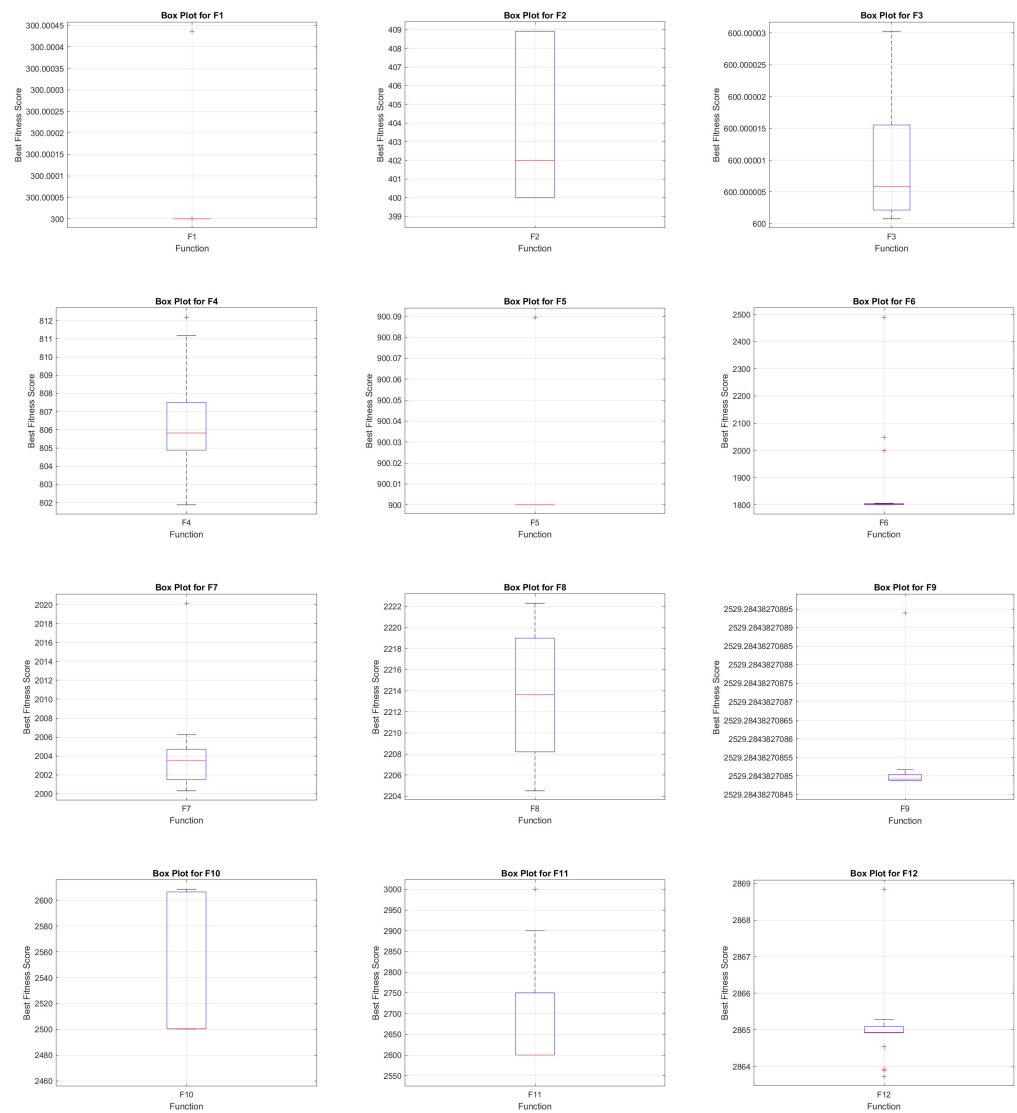


Figure 7. Box plot analysis over CEC2022 (F1–F12).

5.7. JADESCA Sensitivity Analysis

The sensitivity analysis heatmaps for the JADESCA algorithm across CEC (F1–F12), as shown in Figure 8, reveals significant insights into the impact of varying the number of search agents and the maximum iterations on the final fitness values. For functions like F1, F6, and F11, a clear trend is observed where increasing the number of iterations results in improved fitness values, especially when the number of search agents is kept low. This indicates that these functions benefit more from longer computational efforts when fewer agents are used. On the other hand, functions such as F2, F4, and F8 exhibit relatively stable performance across different iterations and agent configurations, suggesting a robustness in the optimization process that is less sensitive to these parameters. Particularly, for F5 and F9, the results show that beyond a certain threshold of search agents and iterations, the improvement in fitness values becomes negligible, highlighting an optimal balance that can be achieved without unnecessarily increasing computational cost. These heatmaps provide a comprehensive view of how different parameter settings influence the effectiveness of the JADESCA algorithm, guiding future efforts to fine-tune the algorithm for specific problem instances.



Figure 8. Sensitivity analysis over CEC2022 (F1–F12).

5.8. JADESCA Histogram Analysis

The JADESCA histogram analysis figures illustrate the distribution of final fitness values across various functions, highlighting the performance and stability of the optimization process, as shown in Figure 9. For function F1, the histogram shows a highly concentrated distribution of final fitness values around 300, with a minor proportion reaching around 650, indicating consistent performance with occasional outliers. In contrast, function F2 exhibits a broader distribution, with fitness values primarily clustered between 400 and 410, suggesting more variability in the results. Function F4 shows a more spread out distribution, with final fitness values ranging from 805 to 825, indicating greater diversity in the optimization outcomes. The histograms for functions F5 and F9 reveal extremely narrow distributions, with final fitness values tightly clustered around 900 and 2529, respectively, demonstrating high stability and convergence in the optimization process. For function F7, the distribution spans a wider range from 2005 to 2025, indicating some variability in the optimization results. Functions F8, F10, and F11 display distributions that are relatively concentrated, but with notable spread, indicating a balance between convergence and diversity in the solutions. Finally, the histogram for function F12 shows a distribution with a peak around 2864, but with some values extending to 2866, reflecting a generally consistent but slightly variable optimization outcome.

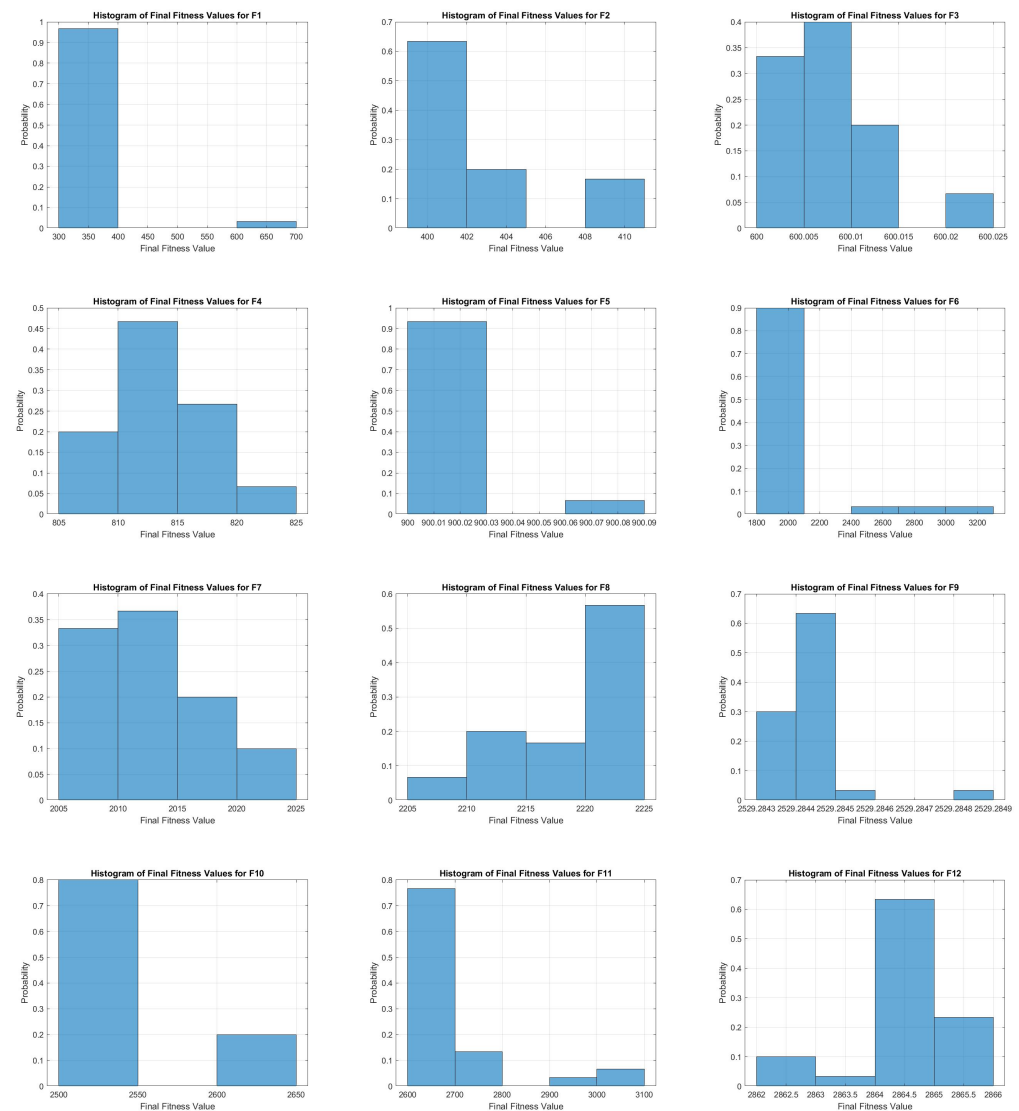


Figure 9. Histogram analysis over CEC2022 (F1–F12).

6. Application in Engineering Design

The application of advanced optimization algorithms to engineering design problems has become an essential part of modern engineering practices. Engineering problems, such as the welded beam design, pressure vessel design, spring design, and speed reducer design, present complex, multi-objective challenges that require a balance between minimizing cost, weight, and material use, while adhering to strict performance and safety constraints. In this section, we apply JADESCA to solve several well-known engineering design problems. JADESCA combines the adaptive capabilities of JADE, which excels in maintaining diversity in the search space, with the SCA optimizer's exploitation features, effectively fine-tuning solutions in the local search space. By applying JADESCA to these engineering problems, the following can be achieved.

6.1. Welded Beam Design Problem

The welded beam design problem is a classical optimization problem that aims to minimize the cost of fabricating a welded beam subject to a variety of constraints related to stress, deflection, and geometry. The objective function and constraints are derived from the physical properties and requirements of the beam.

The schematic of the welded beam is shown in Figure 10.

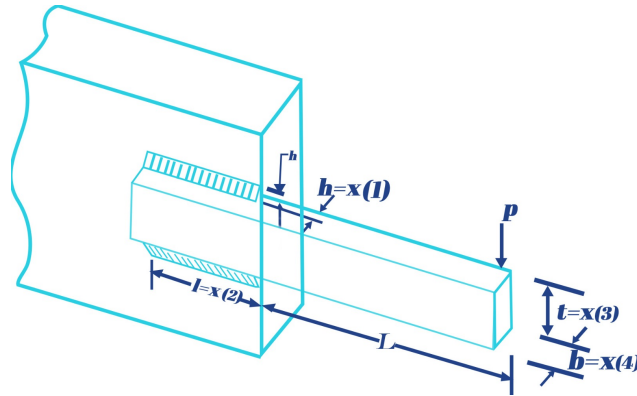


Figure 10. Schematic of the welded beam design problem.

The objective function to be minimized is given by Equation (5):

$$z = 1.10471x_1^2x_2 + 0.04811x_3x_4(14 + x_2), \quad (5)$$

where x_1 , x_2 , x_3 , and x_4 are the design variables representing the dimensions of the beam.

The parameters involved in the problem are as follows:

$$\begin{aligned} P &= 6000 \text{ lb}, \\ L &= 14 \text{ in}, \\ E &= 30 \times 10^6 \text{ psi}, \\ G &= 12 \times 10^6 \text{ psi}, \\ \tau_{\max} &= 13600 \text{ psi}, \\ \sigma_{\max} &= 30000 \text{ psi}, \\ \delta_{\max} &= 0.25 \text{ in}. \end{aligned}$$

The moment acting on the weld can be expressed as shown in Equation (6):

$$M = P \left(L + \frac{x_2}{2} \right). \quad (6)$$

The radius of the critical section is given by Equation (7):

$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2} \right)^2}. \quad (7)$$

The polar moment of inertia is defined as shown in Equation (8):

$$J = 2 \left(\sqrt{2}x_1x_2 \left(\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2} \right)^2 \right) \right). \quad (8)$$

The stresses in the weld are calculated using Equations (9)–(11):

$$\tau_1 = \frac{P}{\sqrt{2}x_1x_2}, \quad (9)$$

$$\tau_2 = \frac{MR}{J}, \quad (10)$$

$$\tau = \sqrt{\tau_1^2 + 2\tau_1\tau_2 \left(\frac{x_2}{2R} \right) + \tau_2^2}. \quad (11)$$

The normal stress in the beam is expressed by Equation (12):

$$\sigma = \frac{6PL}{x_4 x_3^2}. \quad (12)$$

The deflection of the beam is given by Equation (13):

$$\delta = \frac{4PL^3}{Ex_4 x_3^3}. \quad (13)$$

The critical buckling load is defined by Equation (14):

$$P_c = \left(4.013E \sqrt{\frac{x_3^2 x_4^6}{36}} / L^2 \right) \left(1 - \frac{x_3}{2L} \sqrt{\frac{E}{4G}} \right). \quad (14)$$

The constraints of the problem are expressed in Equation (21):

$$c_1 = \tau - \tau_{\max}, \quad (15)$$

$$c_2 = \sigma - \sigma_{\max}, \quad (16)$$

$$c_3 = x_1 - x_4, \quad (17)$$

$$c_4 = 0.10471x_1^2 + 0.04811x_3x_4(14 + x_2) - 5, \quad (18)$$

$$c_5 = 0.125 - x_1, \quad (19)$$

$$c_6 = \delta - \delta_{\max}, \quad (20)$$

$$c_7 = P - P_c. \quad (21)$$

The penalty function to account for constraint violations is defined by Equation (22):

$$g = \max(0, c), \quad (22)$$

where c is the vector of constraint violations.

Finally, the objective function incorporating penalties is as follows:

$$f = z + 10^5 (\sum v + \sum g), \quad (23)$$

where v is an indicator function that is 0 if a constraint is satisfied and 1 otherwise.

As shown in Equation (23), the overall objective function f includes a large penalty for any constraint violations, ensuring that the optimizer seeks feasible solutions that also minimize the fabrication cost.

As shown in Table 6, the results of the welded beam design problem highlight the performance of various optimizers. Among them, GTO emerged as the best performer, achieving the lowest minimum value (1.724852) with a moderate computation time (6.924 s), demonstrating its effectiveness in finding optimal solutions. AHA and MPA also performed well, ranking second and third, respectively, with slightly higher minimum values but showing superior consistency, as indicated by their low standard deviations (AHA: 9.04×10^{-6} ; MPA: 7.56×10^{-6}). In contrast, AOA performed poorly, showing the highest minimum value (2.029756) and standard deviation (0.3215), indicating inefficiency and inconsistency in solution quality. This analysis underscores the importance of choosing the appropriate optimizer based on the specific requirements of the problem, balancing accuracy, consistency, and computational efficiency.

Table 6. Results of the welded beam design problem.

Optimizer	Min	Mean	Max	Std	Time	Rank
JADESCA	1.724852	1.839658	2.299187	0.231405	6.924423	1
AHA	1.724853	1.724861	1.724882	9.04E-06	3.395266	2
MPA	1.724855	1.724864	1.724877	7.56E-06	4.500358	3
FDA	1.724865	1.916787	2.575409	0.288447	4.774226	4
POA	1.724876	1.72551	1.726224	0.000493	4.629484	5
COA	1.72579	1.801442	1.995475	0.11164	4.360163	6
MFO	1.727271	1.942539	2.512453	0.288391	2.005437	7
GWO	1.727572	1.731249	1.738839	0.003317	2.288584	8
GJO	1.729461	1.733087	1.738267	0.003438	3.211563	9
MVO	1.731516	1.763911	1.844405	0.036688	3.577161	10
MGO	1.733116	1.846738	2.293936	0.173521	15.34788	11
HLOA	1.807531	2.654731	4.547732	0.821542	4.3563	12
SCA	1.835108	1.923396	2.005675	0.054927	1.769805	13
AOA	2.029756	2.392391	2.982249	0.321545	2.825179	14

As shown in Table 7, the comparison of optimization algorithms applied to the welded beam design problem reveals that JADESCA achieved the best score (1.724852), closely followed by AHA and MPA. These top optimizers exhibited similar values in key design variables, indicating convergence to a similar optimal region in the design space. In contrast, algorithms such as HLOA, SCA, and AOA showed significant deviations in both the best score and design variables, particularly HLOA and AOA, which exhibited substantial divergence in X2, suggesting over-exploration without effective convergence. Thus, JADESCA, AHA, and MPA are identified as the most effective optimizers, while HLOA, SCA, and AOA are less suitable for this design problem due to their inconsistency and suboptimal performance.

Table 7. Optimization algorithms of the welded beam design problem.

Optimizer	Best Score	X1	X2	X3	X4
JADESCA	1.724852	0.20573	3.470489	9.036624	0.20573
AHA	1.724853	0.205729	3.470499	9.036625	0.20573
MPA	1.724855	0.205728	3.470521	9.036621	0.20573
FDA	1.724865	0.205725	3.470594	9.036646	0.20573
POA	1.724876	0.205735	3.470427	9.036516	0.205735
COA	1.72579	0.205428	3.478037	9.036359	0.205787
MFO	1.727271	0.20397	3.508765	9.036624	0.20573
GWO	1.727572	0.205872	3.471422	9.030063	0.206191
GJO	1.729461	0.205433	3.486698	9.051544	0.205768
MVO	1.731516	0.202163	3.558033	9.037382	0.205772
MGO	1.733116	0.200261	3.589762	9.042679	0.205699
HLOA	1.807531	0.164289	4.656296	9.036401	0.205741
SCA	1.835108	0.189043	3.509404	10	0.201401
AOA	2.029756	0.202836	3.689024	10	0.218807

6.2. Pressure Vessel Design Problem

The pressure vessel design problem is an optimization problem aimed at minimizing the cost of constructing a pressure vessel. This problem includes constraints related to the dimensions and material properties of the vessel to ensure it meets safety and performance standards.

The schematic of the pressure vessel is shown in Figure 11.

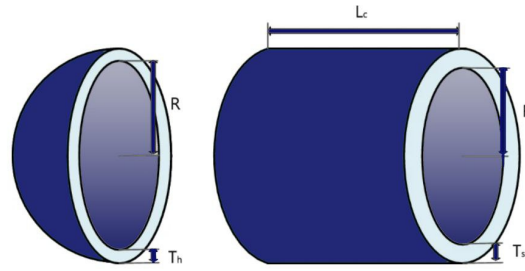


Figure 11. Schematic of the pressure vessel design problem.

The objective function to be minimized is expressed as Equation (24):

$$z = 0.6224 \times (0.0625 \times x_1) \times x_3 \times x_4 + 1.7781 \times (0.0625 \times x_2) \times x_3^2 + 3.1661 \times (0.0625 \times x_1)^2 \times x_4 + 19.84 \times (0.0625 \times x_1)^2 \times x_3, \quad (24)$$

where x_1 , x_2 , x_3 , and x_4 are the design variables representing the thickness of the shell, the thickness of the head, the inner radius, and the length of the cylindrical section, respectively.

The constraints of the problem are defined as follows in Equation (32):

$$c_1 = -0.0625 \times x_1 + 0.0193 \times x_3, \quad (25)$$

$$c_2 = -0.0625 \times x_2 + 0.00954 \times x_3, \quad (26)$$

$$c_3 = -\pi \times x_3^2 \times x_4 - \frac{4}{3} \pi \times x_3^3 + 1296000, \quad (27)$$

$$c_4 = x_4 - 240, \quad (28)$$

$$c_5 = 1 - x_1, \quad (29)$$

$$c_6 = 1 - x_2, \quad (30)$$

$$c_7 = 10 - x_3, \quad (31)$$

$$c_8 = 10 - x_4. \quad (32)$$

The penalty function used to incorporate constraint violations is defined as Equation (33):

$$g = \max(0, c), \quad (33)$$

where c is the vector of constraint violations.

The final objective function, incorporating penalties for constraint violations, is given by the following:

$$f = z + 10^5 (\sum v + \sum g), \quad (34)$$

where v is an indicator function that is 0 if a constraint is satisfied and 1 otherwise.

As shown in Equation (34), the objective function f is constructed by adding a large penalty for any constraint violations to ensure that the optimizer finds feasible solutions that also minimize the construction cost of the pressure vessel.

As shown in Table 8, the comparison of various optimization algorithms for the pressure vessel design problem highlights that MFO is the best performer, achieving the lowest minimum value (5885.333) but with some variability in the solutions. JADESCA closely follows, matching MFO's minimum value with more consistent performance, indicated by a lower mean and standard deviation. POA also performs well, ranking third with a low minimum and small standard deviation. Notably, MPA stands out for its exceptional consistency, having the lowest standard deviation (0.025), although it ranks fourth due to slightly longer computation time. In contrast, AOA performs the worst, with the highest minimum, mean, and maximum values, coupled with extremely high variability. Similarly, SCA and MGO rank low due to high variability and less optimal solutions. Thus, MFO, JADESCA, and POA are identified as the top performers, with MPA noted for its high consistency, whereas AOA, SCA, and MGO are less suitable for the pressure vessel design problem.

Table 8. Statistical results of the pressure vessel design problem.

Optimizer	Min	Mean	Max	Std	Time	Rank
JADESCA	5885.333	6308.676	7319.001	596.4261	2.423681	1
MFO	5885.333	6022.59	6607.425	234.8754	6.142237	2
POA	5885.333	5915.393	6039.32	60.19246	4.271487	3
MPA	5885.336	5885.359	5885.423	0.025197	4.873054	4
FDA	5885.487	6279.071	7314.904	460.3439	5.573389	5
AHA	5886.149	5907.304	5975.677	36.16151	3.353356	6
GWO	5902.452	6025.497	6943.246	322.9055	2.344287	7
GJO	5910.013	6439.871	7309.613	578.9419	2.713069	8
COA	5927.161	6225.893	7009.73	312.9504	4.008542	9
HLOA	6010.778	6528.31	7138.38	372.3133	4.164848	10
MVO	6096.298	6686.698	7282.354	389.2835	3.787784	11
MGO	6111.014	6810.098	7319.001	462.936	13.56634	12
SCA	6193.138	7509.86	8750.035	761.746	3.153723	13
AOA	6708.505	13887.01	29845.51	6427.656	2.613052	14

As shown in Table 9, the comparison of optimization algorithms for the pressure vessel design problem reveals that MFO, JADESCA, and POA achieved the best score of 5885.333, with nearly identical values for all design variables, indicating consistent convergence to the optimal solution. MPA also performed exceptionally well, with a minimal deviation in X4, resulting in a slightly higher but nearly optimal best score (5885.336). FDA produced a low best score (5885.487) but showed slightly more variation in the design variables, particularly in X4, suggesting less precision in optimization. AHA and GWO had moderate performance, with higher best scores and greater variations in design variables. In contrast, lower-ranking algorithms like GJO, COA, HLOA, and others showed more significant deviations from the optimal solution. Notably, AOA had the poorest performance with a best score of 6708.505 and significant deviations in design variables, indicating poor optimization capability and lack of convergence. Similarly, algorithms like MGO, SCA, and MVO demonstrated considerable deviations in both best scores and variable values, reflecting their inability to achieve near-optimal results. Thus, MFO, JADESCA, and POA were identified as the most effective optimizers, consistently converging to the optimal solution, while AOA, MGO, SCA, and similar algorithms were less effective, displaying significant deviations and suboptimal results.

Table 9. Results of the pressure vessel design problem.

Optimizer	Best Score	X1	X2	X3	X4
JADESCA	5885.333	12.4507	6.154387	40.31962	200
MFO	5885.333	12.4507	6.154387	40.31962	200
POA	5885.333	12.4507	6.154387	40.31962	200
MPA	5885.336	12.45071	6.154399	40.31967	199.9993
FDA	5885.487	12.45214	6.155101	40.3243	199.9348
AHA	5886.149	12.45719	6.15776	40.34002	199.7166
GWO	5902.452	12.4873	6.237621	40.43423	198.4494
GJO	5910.013	12.49145	6.208815	40.40736	199.1607
COA	5927.161	12.50861	6.351544	40.47129	197.9056
HLOA	6010.778	13.19716	6.72448	42.73693	168.8822
MVO	6096.298	14.03449	6.909339	45.23691	141.3225
MGO	6111.014	14.28094	7.059075	46.24656	131.222
SCA	6193.138	12.97374	6.605777	42.00596	186.8397
AOA	6708.505	12.85614	9.456333	40.53201	200

6.3. Spring Design Problem

The spring design problem focuses on optimizing the dimensions of a compression spring to minimize its volume while satisfying various constraints related to the spring's functionality and physical limitations.

The schematic of the spring is shown in Figure 12.

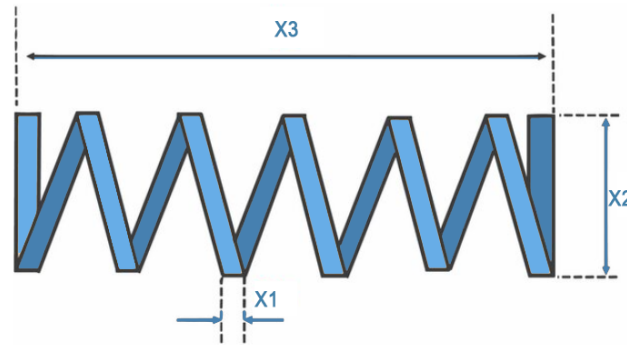


Figure 12. Schematic of the spring design problem.

The objective function to be minimized is given by Equation (35):

$$z = (x_3 + 2)x_2x_1^2, \quad (35)$$

where x_1 , x_2 , and x_3 are the design variables representing the wire diameter, the mean coil diameter, and the number of active coils, respectively.

The constraints of the problem are defined as follows in Equation (39):

$$c_1 = -\frac{x_2^3x_3}{71785x_1^4} + 1, \quad (36)$$

$$c_2 = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} - 1, \quad (37)$$

$$c_3 = 1 - \frac{140.45x_1}{x_2^2x_3}, \quad (38)$$

$$c_4 = \frac{x_1 + x_2}{1.5} - 1. \quad (39)$$

The penalty function used to handle constraint violations is defined as Equation (40):

$$g = \max(0, c), \quad (40)$$

where c is the vector of constraint violations.

The final objective function, incorporating penalties for constraint violations, is given by Equation (41):

$$f = z + 10^5 (\sum v + \sum g), \quad (41)$$

where v is an indicator function that is 0 if a constraint is satisfied and 1 otherwise.

As shown in Equation (41), the objective function f includes a large penalty for any constraint violations, ensuring that the optimizer seeks feasible solutions that also minimize the volume of the spring.

As shown in Table 10, the comparison of various optimization algorithms for the spring design problem reveals that MPA achieved the best performance, with the lowest minimum value (0.012665) and an exceptionally low standard deviation (2.1×10^{-8}), indicating remarkable consistency and precision. POA closely follows, also achieving the same minimum value but with slightly more variability. FDA and AHA also performed well but showed more variability and less consistency compared to MPA and POA. GTO

ranked fifth, showing good performance with a small standard deviation but a longer computational time. In contrast, lower-ranking algorithms like MFO, MGO, and HLOA demonstrated greater variability and less precision in finding optimal solutions. AOA and MVO exhibited the poorest performance, with the highest minimum, mean, and maximum values, as well as notable standard deviations, indicating substantial inefficiency in converging to optimal solutions.

Table 10. Statistical results of the spring design problem.

Optimizer	Min	Mean	Max	Std	Time	Rank
JADESCA	0.012665	0.012665	0.012665	2.1E-08	4.041762	1
POA	0.012665	0.012678	0.012695	1.16E-05	3.663748	2
FDA	0.012665	0.012867	0.014126	0.000456	3.95537	3
AHA	0.012665	0.012679	0.012717	1.81E-05	2.963405	4
GTO	0.012666	0.012673	0.012687	7.54E-06	7.12426	5
MFO	0.012674	0.014288	0.017773	0.001868	1.680422	6
MGO	0.012674	0.013498	0.015971	0.001367	13.50008	7
HLOA	0.012674	0.013411	0.017773	0.001572	3.891885	8
COA	0.012676	0.012924	0.014451	0.000551	3.558568	9
GWO	0.012677	0.012698	0.012783	3.07E-05	1.791368	10
GJO	0.012688	0.012733	0.012871	5.57E-05	2.312313	11
SCA	0.01279	0.013097	0.013575	0.000322	1.572505	12
AOA	0.014104	0.014232	0.014668	0.000171	2.307731	13
MVO	0.017842	0.018185	0.018523	0.000237	2.839032	14

As shown in Table 11, the comparison of optimization algorithms for the spring design problem indicates that JADESCA and POA are the top performers, achieving the best scores (0.012665237 and 0.012665268, respectively) with very similar values for critical design variables. FDA and AHA also performed well, with scores nearly identical to the best performers, but with slight variations in X3, suggesting minor differences in their approach. GTO achieved a slightly higher best score (0.012666223) and showed more variation in X3, indicating a different trade-off in design parameters. In contrast, algorithms like MFO, MGO, and HLOA demonstrated higher best scores and more significant deviations in variables 2 and 3, indicating less effective optimization. AOA and MVO performed the worst, with AOA showing a considerably higher best score (0.014104043) and significant deviation in X3, while MVO exhibited extreme values for all variables, suggesting poor convergence and substantial inefficiency in optimizing the design.

Table 11. Results of the spring design problem

Optimizer	Best Score	X1	X2	X3
JADESCA	0.012665	0.051693	0.356804	11.2839
POA	0.012665	0.051645	0.35567	11.35068
FDA	0.012665	0.051776	0.358802	11.16782
AHA	0.012665	0.051675	0.356372	11.30941
GTO	0.012666	0.051456	0.351148	11.62311
MFO	0.012674	0.051001	0.340397	12.31408
MGO	0.012674	0.051	0.340366	12.31615
HLOA	0.012674	0.051	0.340366	12.31619
COA	0.012676	0.051159	0.34404	12.0772
GWO	0.012677	0.051	0.340326	12.3214
GJO	0.012688	0.051	0.340168	12.33981
SCA	0.01279	0.051	0.339425	12.48745
AOA	0.014104	0.051	0.318973	15
MVO	0.017842	0.068702	0.919657	2.110304

6.4. Speed Reducer Design Problem

The speed reducer design problem is an optimization task focused on minimizing the weight of a speed reducer subject to several constraints. This problem involves finding optimal dimensions for the various components of the reducer to achieve the minimum weight while satisfying mechanical and geometric constraints.

The schematic of the speed reducer is shown in Figure 13.

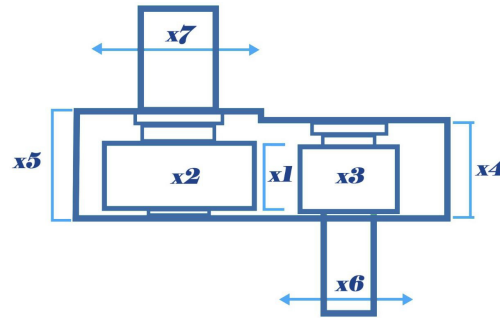


Figure 13. Schematic of the speed reducer design problem.

The objective function to be minimized is given by Equation (42):

$$z = 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) - 1.508x_1(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3) + 0.7854(x_4x_6^2 + x_5x_7^2), \quad (42)$$

where $x_1, x_2, x_3, x_4, x_5, x_6$, and x_7 are the design variables representing various dimensions and components of the speed reducer.

The constraints of the problem are defined as follows in Equation (53):

$$c_1 = \frac{27}{x_1x_2^2x_3} - 1, \quad (43)$$

$$c_2 = \frac{397.5}{x_1x_2^2x_3^2} - 1, \quad (44)$$

$$c_3 = \frac{1.93x_4^3}{x_2x_3x_6^4} - 1, \quad (45)$$

$$c_4 = \frac{1.93x_5^3}{x_2x_3x_7^4} - 1, \quad (46)$$

$$c_5 = \frac{1}{110x_6^3} \left(\left(\frac{745x_4}{x_2x_3} \right)^2 + 16.9 \times 10^6 \right)^{0.5} - 1, \quad (47)$$

$$c_6 = \frac{1}{85x_7^3} \left(\left(\frac{745x_5}{x_2x_3} \right)^2 + 157.5 \times 10^6 \right)^{0.5} - 1, \quad (48)$$

$$c_7 = \frac{x_2x_3}{40} - 1, \quad (49)$$

$$c_8 = \frac{5x_2}{x_1} - 1, \quad (50)$$

$$c_9 = \frac{x_1}{12x_2} - 1, \quad (51)$$

$$c_{10} = \frac{1.5x_6 + 1.9}{x_4} - 1, \quad (52)$$

$$c_{11} = \frac{1.1x_7 + 1.9}{x_5} - 1. \quad (53)$$

The penalty function used to handle constraint violations is defined as Equation (54):

$$g = \max(0, c), \quad (54)$$

where c is the vector of constraint violations.

The final objective function, incorporating penalties for constraint violations, is given by the following:

$$f = z + 10^5 (\sum v + \sum g), \quad (55)$$

where v is an indicator function that is 0 if a constraint is satisfied and 1 otherwise.

As shown in Equation (55), the objective function f includes a large penalty for any constraint violations, ensuring that the optimizer seeks feasible solutions that also minimize the weight of the speed reducer.

As shown in Table 12, the results from the speed reducer design problem reveal that JADESCA is the most effective optimizer, achieving the lowest minimum and mean values with a small standard deviation, indicating consistent performance and securing the first rank. MFO also performed well, matching the minimum value of JADESCA but with slightly higher mean and maximum values, placing it second. DOA, while achieving the same minimum, showed higher variability and ranked third. RIME and COA offered competitive and consistent results, ranking fourth and fifth, respectively. Lower-ranked optimizers like AVOA, ZOA, SCSO, and SHIO demonstrated higher mean and maximum values, indicating less optimal and more variable performance. In contrast, optimizers like HHO, WOA, TTHHO, ROA, FLO, CPO, and SSOA performed poorly, with high variability and suboptimal outcomes, making them unsuitable for this problem.

Table 12. Statistical results of the speed reducer design problem.

Optimizer	Min	Mean	Max	Std	Time	Rank
JADESCA	2994.471	2994.92	2998.949	1.415589	0.834734	1
MFO	2994.471	2998.399	3033.749	12.42062	0.822204	2
DOA	2994.471	3099.145	3363.873	123.884	1.341059	3
RIME	2994.731	2996.055	2997.135	0.77945	0.902076	4
COA	2994.965	2996.059	2997.478	0.733906	1.788052	5
AVOA	2995.032	3000.607	3005.723	3.348249	0.999476	6
ZOA	2995.666	3003.03	3008.362	3.981207	1.453907	7
SCSO	2996.844	3007.816	3018.212	5.977215	0.944803	8
SHIO	2998.583	3004.512	3011.048	3.713896	0.831343	9
GWO	3001.793	3010.462	3028.562	8.734309	0.914363	10
FVIM	3004.641	3009.413	3019.998	4.724602	0.924222	11
HHO	3008.729	3219.386	4025.775	317.9044	1.57597	12
WOA	3015.221	3251.743	4316.943	380.4302	0.803014	13
OMA	3016.568	3075.234	3163.744	42.72921	2.175797	14
SOA	3017.402	3038.222	3069.537	15.92089	0.733158	15
TTHHO	3018.703	3351.402	4298.54	486.7529	1.825567	16
STOA	3024.978	3040.187	3058.421	11.88368	0.751259	17
AO	3048.003	3104.258	3142.244	32.65513	2.281716	18
ROA	3058.109	104116.3	1007316	317352.9	2.03042	19
Chimp	3075.128	3131.287	3189.936	29.19947	1.418818	20
SCA	3076.29	3139.106	3191.778	33.86397	0.721655	21
FLO	3797.939	723258.5	1048408	496081.5	1.378594	22
CPO	4632.48	5752.925	5877.419	393.6845	1.37111	23
SSOA	1017738	1092409	1250257	67188.3	1.265369	24

As shown in Table 13, in the speed reducer design problem, JADESCA and MFO both achieved the best score of 2994.471066, with identical variable values across all dimensions, indicating their high effectiveness and consistency in finding optimal solutions. DOA also achieved a very close best score, differing only slightly, with minor variations in variables 5 and 7, suggesting high precision with some divergence from the ideal. RIME

and COA, while close in performance, showed a slight increase in the best score and some variability in the design variables, indicating slightly less precision. Lower-ranked optimizers like AVOA, ZOA, and SCSO showed a noticeable increase in the best score and variability, particularly in variables 5 through 7, indicating reduced accuracy and consistency. SHIO and GWO further increased the best score, with greater deviations in the design variables, particularly in X5. Optimizers such as FVIM, HHO, and WOA showed significantly higher best scores and greater divergence in variables 5, 6, and 7, highlighting issues in convergence or global optimization. TTHHO, STOA, AO, and ROA exhibited a significant drop in performance, with best scores far from optimal and substantial variability in variables 5, 6, and 7, indicating poor adaptation to the problem's constraints. The worst performers, such as chimp, SCA, FLO, CPO, and SSOA, exhibited drastically higher best scores and highly varied, non-ideal values across almost all variables, particularly X5. These results underscore the superiority of JADESCA and MFO for this problem, while others struggle with precision, consistency, and effective optimization.

Table 13. Optimization algorithms of the speed reducer design problem.

Optimizer	Best Score	X1	X2	X3	X4	X5	X6	X7
JADESCA	2994.471	3.5	0.7	17	7.3	7.71532	3.350215	5.286654
MFO	2994.471	3.5	0.7	17	7.3	7.71532	3.350215	5.286654
DOA	2994.471	3.5	0.7	17	7.3	7.715321	3.350215	5.286654
RIME	2994.731	3.500311	0.7	17	7.3	7.721184	3.350242	5.286657
COA	2994.965	3.500451	0.700009	17.00029	7.30423	7.719595	3.350564	5.286661
AVOA	2995.032	3.500001	0.7	17	7.306503	7.737853	3.350227	5.286662
ZOA	2995.666	3.500284	0.7	17	7.305977	7.733636	3.350936	5.287355
SCSO	2996.844	3.500198	0.7	17	7.457542	7.740782	3.350943	5.286907
SHIO	2998.583	3.501653	0.7	17	7.317075	7.795592	3.354308	5.28745
GWO	3001.793	3.500026	0.7	17	7.564603	7.796651	3.360743	5.287422
FVIM	3004.641	3.50206	0.7	17	7.617562	7.762973	3.358744	5.291876
HHO	3008.729	3.516346	0.7	17	7.705688	7.872776	3.353213	5.286708
WOA	3015.221	3.5	0.7	17	7.3	7.77653	3.419526	5.288819
OMA	3016.568	3.520695	0.701083	17.00012	7.405428	7.9063	3.355592	5.290619
SOA	3017.402	3.503132	0.7	17.03941	7.3	8.022402	3.352044	5.298658
TTHHO	3018.703	3.517373	0.7	17	7.3	8.025153	3.350215	5.30324
STOA	3024.978	3.550997	0.7	17	7.82667	7.917368	3.352365	5.287983
AO	3048.003	3.546055	0.7	17	7.706598	8.123628	3.436698	5.286873
ROA	3058.109	3.531841	0.7	17	8.228035	8.071254	3.479612	5.28709
Chimp	3075.128	3.6	0.7	17	7.3	8.3	3.366746	5.324488
SCA	3076.29	3.6	0.7	17	7.3	8.3	3.392345	5.31606
FLO	3797.939	3.584428	0.716881	19.75319	7.620048	8.08554	3.348653	5.488105
CPO	4632.48	3.552042	0.705571	23.37616	8.098939	8.13028	3.779752	5.477325
SSOA	1017738	3.502899	0.709589	25.14204	8.030486	7.967745	3.395998	5.381849

7. Limitations and Future Work

The JADESCA algorithm has demonstrated strong performance across various benchmark functions and engineering design problems; however, several limitations and potential challenges must be acknowledged. One challenge is its scalability to high-dimensional problems, where efficiency and computational cost may become significant concerns. Additionally, the algorithm relies on predefined control parameters, which may require manual tuning, reducing its adaptability in dynamic environments. While JADESCA includes adaptive mechanisms to balance exploration and exploitation, the risk of premature convergence or insufficient exploration in complex landscapes remains. The hybridization of JADE and SCA also increases the algorithm's complexity, potentially leading to higher computational costs compared to simpler methods. Future work could focus on addressing these limitations by exploring the integration of adaptive control parameter mechanisms or hybridizing JADESCA with machine learning techniques to dynamically adjust these parameters based on real-time performance. Furthermore, expanding its application to

more complex, multi-modal, and high-dimensional problems will provide deeper insights into its scalability and efficiency. Another interesting direction would be the extension of JADESCA to multi-objective optimization and real-time optimization frameworks, where the algorithm's ability to balance competing objectives or constraints could be further leveraged. Additionally, incorporating problem-specific knowledge or domain expertise into the optimization process may enhance the algorithm's performance in specialized fields such as aerospace, materials science, and bioengineering.

8. Conclusions

In this paper, we presented the development and application of the JADESCA optimization algorithm, a hybrid approach combining the strengths of JADE (differential evolution with adaptive parameters) and the sine cosine algorithm (SCA). The algorithm was rigorously tested across a variety of benchmark functions from the CEC2022 competition, as well as complex engineering design problems, including the welded beam, pressure vessel, spring, and speed reducer design problems. The results demonstrate that JADESCA is highly effective in solving both unconstrained and constrained optimization problems. Through detailed analyses, including convergence curves, search history diagrams, and statistical comparisons, JADESCA consistently outperformed several state-of-the-art optimizers, particularly in terms of convergence speed, solution accuracy, and robustness. The algorithm's ability to balance exploration and exploitation was evident across various test cases, leading to optimal or near-optimal solutions in engineering applications. One of the key highlights of this research is JADESCA's superior performance against 25 top optimizers in the CEC2022 benchmark suite. The statistical comparisons, including the Wilcoxon rank-sum tests as shown in Appendix A, confirmed JADESCA's competitive edge, solidifying its position as a reliable and efficient optimizer for complex, real-world problems. Future work may explore further enhancements to JADESCA and its application to even more complex, high-dimensional problems, as well as its potential integration with other optimization frameworks.

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

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

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

Appendix A. Wilcoxon Rum-Rank Results of JADESCA with the Compared Optimizers

Table A1. Wilcoxon Sum-rank results of JADESCA with the compared optimizers.[illegible]

Table A1. Cont.

Fun	FVIM	FLO	STOA	SOA	SPBO	AO	SSOA	TTHHO	Chimp
F4	4.08E-05 U: 637.0000 +	3.02E-11 U: 465.0000 +	3.34E-11 U: 466.0000 +	4.08E-11 U: 468.0000 +	3.02E-11 U: 465.0000 +	6.7E-11 U: 473.0000 +	3.02E-11 U: 465.0000 +	3.02E-11 U: 465.0000 +	3.02E-11 U: 465.0000 +
F5	1.72E-12 U: 465.0000 +	1.72E-12 U: 465.0000 +	1.72E-12 U: 465.0000 +	1.72E-12 U: 465.0000 +	1.72E-12 U: 465.0000 +	1.72E-12 U: 465.0000 +	1.72E-12 U: 465.0000 +	1.72E-12 U: 465.0000 +	1.72E-12 U: 465.0000 +
F6	9.92E-11 U: 477.0000 +	3.02E-11 U: 465.0000 +	3.02E-11 U: 465.0000 +	3.02E-11 U: 465.0000 +	3.02E-11 U: 465.0000 +	3.02E-11 U: 465.0000 +	3.02E-11 U: 465.0000 +	3.34E-11 U: 466.0000 +	3.02E-11 U: 465.0000 +
F7	3.02E-11 U: 465.0000 +	3.02E-11 U: 465.0000 +	3.02E-11 U: 465.0000 +	3.02E-11 U: 465.0000 +	3.02E-11 U: 465.0000 +	3.02E-11 U: 465.0000 +	3.02E-11 U: 465.0000 +	3.02E-11 U: 465.0000 +	3.02E-11 U: 465.0000 +
F8	5.57E-10 U: 495.0000 +	3.02E-11 U: 465.0000 +	3.02E-11 U: 465.0000 +	3.69E-11 U: 467.0000 +	3.02E-11 U: 465.0000 +	4.2E-10 U: 492.0000 +	3.02E-11 U: 465.0000 +	8.99E-11 U: 476.0000 +	3.02E-11 U: 465.0000 +
F9	3.01E-11 U: 465.0000 +	3.01E-11 U: 465.0000 +	3.01E-11 U: 465.0000 +	3.01E-11 U: 465.0000 +	3.01E-11 U: 465.0000 +	3.01E-11 U: 465.0000 +	3.01E-11 U: 465.0000 +	3.01E-11 U: 465.0000 +	3.01E-11 U: 465.0000 +
F10	4.35E-05 U: 638.0000 +	3.82E-10 U: 491.0000 +	7.69E-08 U: 551.0000 +	9.83E-08 U: 554.0000 -	2.92E-09 U: 513.0000 +	2.23E-09 U: 510.0000 +	1.78E-10 U: 483.0000 +	1.17E-09 U: 503.0000 +	5.46E-09 U: 520.0000 +
F11	2.92E-09 U: 513.0000 +	3.69E-11 U: 467.0000 +	1.85E-08 U: 534.0000 +	2.19E-08 U: 536.0000 +	3.02E-11 U: 465.0000 +	2.83E-08 U: 539.0000 +	3.69E-11 U: 467.0000 +	2.23E-09 U: 510.0000 +	5.49E-11 U: 471.0000 +
F12	3.24E-06 U: 600.0000	2.86E-11 U: 465.0000	0.021352 U: 1071.0000	0.004589 U: 1107.0000	2.86E-11 U: 465.0000	7.37E-07 U: 580.0000	2.86E-11 U: 465.0000	2.49E-10 U: 487.0000	2.11E-08 U: 536.0000
Total	+: 12, -: 0, =: 0	+: 12, -: 0, =: 0	+: 11, -: 1, =: 0	+: 10, -: 2, =: 0	+: 12, -: 0, =: 0	+: 12, -: 0, =: 0	+: 12, -: 0, =: 0	+: 12, -: 0, =: 0	+: 12, -: 0, =: 0

Table A2. Wilcoxon sum-rank results of JADESCA with the first part of the compared optimizers.

Fun	GWO	WOA	MFO	SHIO	ZOA	MTDE	FVIM	SCA	DOA
F1	1.72E-12 U: 465.0000 +	1.72E-12 U: 465.0000 +	2.41E-12 U: 468.0000 +	1.72E-12 U: 465.0000 +	1.72E-12 U: 465.0000 +	1.72E-12 U: 465.0000 +	1.72E-12 U: 465.0000 +	1.72E-12 U: 465.0000 +	1.72E-12 U: 465.0000 +
F2	1.46E-10 U: 483.0000 +	1.51E-07 U: 561.0000 +	4.65E-07 U: 578.0000 +	4.94E-07 U: 576.0000 +	6.23E-07 U: 579.0000 +	2.43E-11 U: 465.0000 +	0.03303 U: 771.0000 +	2.43E-11 U: 465.0000 +	3.47E-10 U: 492.0000 +
F3	3.02E-11 U: 465.0000 +	3.02E-11 U: 465.0000 +	9.48E-06 U: 615.0000 +	3.02E-11 U: 465.0000 +	3.02E-11 U: 465.0000 +	3.02E-11 U: 465.0000 +	3.02E-11 U: 465.0000 +	3.02E-11 U: 465.0000 +	3.02E-11 U: 465.0000 +
F4	1.6E-07 U: 560.0000 +	3.02E-11 U: 465.0000 +	6.06E-11 U: 472.0000 +	1.29E-09 U: 504.0000 +	2.88E-06 U: 598.0000 +	3.02E-11 U: 465.0000 +	4.2E-10 U: 492.0000 +	3.02E-11 U: 465.0000 +	1.78E-10 U: 483.0000 +
F5	1.72E-12 U: 465.0000 +	1.72E-12 U: 465.0000 +	1.2E-09 U: 542.5000 +	1.72E-12 U: 465.0000 +	1.72E-12 U: 465.0000 +	1.72E-12 U: 465.0000 +	1.72E-12 U: 465.0000 +	1.72E-12 U: 465.0000 +	1.72E-12 U: 465.0000 +
F6	5.49E-11 U: 471.0000 +	8.99E-11 U: 476.0000 +	6.7E-11 U: 473.0000 +	9.92E-11 U: 477.0000 +	3.16E-10 U: 489.0000 +	3.02E-11 U: 465.0000 +	1.29E-09 U: 504.0000 +	3.02E-11 U: 465.0000 +	1.01E-08 U: 527.0000 +
F7	3.02E-11 U: 465.0000 +	3.02E-11 U: 465.0000 +	3.02E-11 U: 465.0000 +	3.02E-11 U: 465.0000 +	3.02E-11 U: 465.0000 +	3.02E-11 U: 465.0000 +	0.000856 U: 689.0000 +	3.02E-11 U: 465.0000 +	3.02E-11 U: 465.0000 +
F8	1.73E-06 U: 591.0000 +	7.39E-11 U: 474.0000 +	1.31E-08 U: 530.0000 +	7.39E-11 U: 474.0000 +	1.1E-08 U: 528.0000 +	3.02E-11 U: 465.0000 +	0.003848 U: 719.0000 +	6.7E-11 U: 473.0000 +	9.83E-08 U: 554.0000 +
F9	3.01E-11 U: 465.0000 +	3.01E-11 U: 465.0000 +	0.072331 U: 1035.0000 =	3.01E-11 U: 465.0000 +	3.01E-11 U: 465.0000 +	3.01E-11 U: 465.0000 +	1.09E-10 U: 478.0000 +	3.01E-11 U: 465.0000 +	8.47E-09 U: 525.0000 +
F10	7.6E-07 U: 580.0000 +	2.92E-09 U: 513.0000 +	3.34E-08 U: 541.0000 +	9.53E-07 U: 583.0000 +	1.1E-08 U: 528.0000 +	5.96E-09 U: 521.0000 +	3.37E-05 U: 634.0000 +	8.48E-09 U: 525.0000 +	1.55E-09 U: 506.0000 +
F11	1.01E-08 U: 527.0000 +	1.86E-09 U: 508.0000 +	0.095708 U: 802.0000 =	3.82E-09 U: 516.0000 +	1.01E-08 U: 527.0000 +	1.17E-09 U: 503.0000 +	3.96E-08 U: 543.0000 +	9.26E-09 U: 526.0000 +	2.92E-09 U: 513.0000 +
F12	0.003463 U: 717.0000	1.78E-09 U: 508.0000	0.008784 U: 1092.5000	2.12E-07 U: 564.0000	2.86E-11 U: 465.0000	2.86E-11 U: 465.0000	1.69E-10 U: 483.0000	2.86E-11 U: 465.0000	2.56E-09 U: 512.0000
Total	+:12, -:0, =:0	+:12, -:0, =:0	+:9, -:1, =:2	+:12, -:0, =:0	+:12, -:0, =:0	+:12, -:0, =:0	+:12, -:0, =:0	+:12, -:0, =:0	+:12, -:0, =:0

Table A3. Wilcoxon sum-rank results of JADESCA with the first part of the compared optimizers.

Fun	SDE	HHO	SCSO	OMA	CPO	ROA	CMAES
F1	1.72E-12 U: 465.0000 +	1.72E-12 U: 465.0000 +	1.72E-12 U: 465.0000 +	1.72E-12 U: 465.0000 +	1.72E-12 U: 465.0000 +	1.72E-12 U: 465.0000 +	1.72E-12 U: 465.0000 +
F2	2.89E-05 U: 633.0000 +	4.4E-08 U: 546.0000 +	9.84E-07 U: 585.0000 +	0.143403 U: 816.0000 =	0.000611 U: 684.0000 +	2.43E-11 U: 465.0000 +	8.08E-11 U: 477.0000 +
F3	3.02E-11 U: 465.0000 +	3.02E-11 U: 465.0000 +	3.02E-11 U: 465.0000 +	0.111987 U: 807.0000 =	3.02E-11 U: 465.0000 +	3.02E-11 U: 465.0000 +	3.02E-11 U: 465.0000 +
F4	1.78E-10 U: 483.0000 +	4.08E-11 U: 468.0000 +	3.69E-11 U: 467.0000 +	5.57E-10 U: 495.0000 +	4.5E-11 U: 469.0000 +	3.02E-11 U: 465.0000 +	3.02E-11 U: 465.0000 +
F5	1.72E-12 U: 465.0000 +	1.72E-12 U: 465.0000 +	1.72E-12 U: 465.0000 +	1.72E-12 U: 465.0000 +	1.72E-12 U: 465.0000 +	1.72E-12 U: 465.0000 +	1.72E-12 U: 465.0000 +
F6	6.7E-11 U: 473.0000 +	7.39E-11 U: 474.0000 +	3.02E-11 U: 465.0000 +	4.18E-09 U: 517.0000 +	6.07E-11 U: 472.0000 +	3.02E-11 U: 465.0000 +	3.69E-11 U: 467.0000 +
F7	0.001442 U: 699.0000 +	3.02E-11 U: 465.0000 +	3.02E-11 U: 465.0000 +	2.92E-09 U: 513.0000 +	3.02E-11 U: 465.0000 +	3.02E-11 U: 465.0000 +	3.02E-11 U: 465.0000 +
F8	0.233989 U: 834.0000 =	3.02E-11 U: 465.0000 +	5.49E-11 U: 471.0000 +	2.03E-09 U: 509.0000 +	4.44E-07 U: 573.0000 +	3.02E-11 U: 465.0000 +	2.02E-08 U: 535.0000 +
F9	3.01E-11 U: 465.0000 +	3.01E-11 U: 465.0000 +	3.01E-11 U: 465.0000 +	3.01E-11 U: 465.0000 +	3.17E-07 U: 1259.0000 +	3.01E-11 U: 465.0000 +	3.01E-11 U: 465.0000 +
F10	0.011228 U: 743.0000 +	1.69E-09 U: 507.0000 +	2.2E-07 U: 564.0000 +	2.96E-05 U: 632.0000 -	3.82E-09 U: 516.0000 +	2.15E-10 U: 485.0000 +	0.031466 U: 769.0000 -
F11	2.6E-08 U: 538.0000 +	2.23E-09 U: 510.0000 +	1.1E-08 U: 528.0000 +	1.31E-08 U: 530.0000 +	2.39E-08 U: 537.0000 +	3.69E-11 U: 467.0000 +	1.56E-08 U: 532.0000 +
F12	0.454764 U: 966.0000 +	2.34E-09 U: 511.0000 +	0.074484 U: 794.0000 +	6.42E-10 U: 497.0000 +	0.136842 U: 814.0000 +	2.86E-11 U: 465.0000 +	0.001099 U: 694.0000 +
Total	+:10, -:0, =:2	+:12, -:0, =:0	+:11, -:0, =:1	+:9, -:1, =:2	+:11, -:0, =:1	+:12, -:0, =:0	+:11, -:1, =:0

References

- Penghui, L.; Ewees, A.A.; Beyaztas, B.H.; Qi, C.; Salih, S.Q.; Al-Ansari, N.; Bhagat, S.K.; Yaseen, Z.M.; Singh, V.P. Metaheuristic optimization algorithms hybridized with artificial intelligence model for soil temperature prediction: Novel model. *IEEE Access* **2020**, *8*, 51884–51904. [\[CrossRef\]](#)
- Pan, X.; Xue, L.; Lu, Y.; Sun, N. Hybrid particle swarm optimization with simulated annealing. *Multimed. Tools Appl.* **2019**, *78*, 29921–29936. [\[CrossRef\]](#)
- Shinde, P.P.; Shah, S. A review of machine learning and deep learning applications. In Proceedings of the 2018 Fourth International Conference on Computing Communication Control and Automation (ICCCUBEA), Pune, India, 16–18 August 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–6.
- Beşkirli, A.; Dağ, İ. Parameter extraction for photovoltaic models with tree seed algorithm. *Energy Rep.* **2023**, *9*, 174–185. [\[CrossRef\]](#)
- Beskirli, M.; Hakli, H.; Kodaz, H. The energy demand estimation for Turkey using differential evolution algorithm. *Sādhanā* **2017**, *42*, 1705–1715. [\[CrossRef\]](#)
- Beşkirli, A.; Temurtaş, H.; Özdemir, D. Determination with Linear Form of Turkey's Energy Demand Forecasting by the Tree Seed Algorithm and the Modified Tree Seed Algorithm. *Adv. Electr. Comput. Eng.* **2020**, *20*, 27–34. [\[CrossRef\]](#)
- Beşkirli, A.; Dağ, İ. An efficient tree seed inspired algorithm for parameter estimation of Photovoltaic models. *Energy Rep.* **2022**, *8*, 291–298. [\[CrossRef\]](#)
- Kochenderfer, M.J.; Wheeler, T.A. *Algorithms for Optimization*; MIT Press: Cambridge, MA, USA, 2019.
- Diwekar, U.M. *Introduction to Applied Optimization*; Springer Nature: Cham, Switzerland, 2020; Volume 22.

10. Dorfman, R. *Application of Linear Programming to the Theory of the Firm: Including an Analysis of Monopolistic Firms by Non-Linear Programming*; University of California Press: Oakland, CA, USA, 2022.
11. Avriel, M. Nonlinear programming. In *Mathematical Programming for Operations Researchers and Computer Scientists*; CRC Press: Boca Raton, FL, USA, 2020; pp. 271–367.
12. Pellerin, R.; Perrier, N.; Berthaut, F. A survey of hybrid metaheuristics for the resource-constrained project scheduling problem. *Eur. J. Oper. Res.* **2020**, *280*, 395–416. [\[CrossRef\]](#)
13. Silva, M.A.L.; de Souza, S.R.; Souza, M.J.F.; de Franca Filho, M.F. Hybrid metaheuristics and multi-agent systems for solving optimization problems: A review of frameworks and a comparative analysis. *Appl. Soft Comput.* **2018**, *71*, 433–459. [\[CrossRef\]](#)
14. Gabis, A.B.; Meraihi, Y.; Mirjalili, S.; Ramdane-Cherif, A. A comprehensive survey of sine cosine algorithm: Variants and applications. *Artif. Intell. Rev.* **2021**, *54*, 5469–5540. [\[CrossRef\]](#)
15. Zhan, Z.H.; Wang, Z.J.; Jin, H.; Zhang, J. Adaptive distributed differential evolution. *IEEE Trans. Cybern.* **2019**, *50*, 4633–4647. [\[CrossRef\]](#)
16. Dash, D.R.; Dash, P.; Bisoi, R. Short term solar power forecasting using hybrid minimum variance expanded RVFLN and Sine-Cosine Levy Flight PSO algorithm. *Renew. Energy* **2021**, *174*, 513–537. [\[CrossRef\]](#)
17. Zhao, M.; Wang, X.; Yu, J.; Bi, L.; Xiao, Y.; Zhang, J. Optimization of construction duration and schedule robustness based on hybrid grey Wolf optimizer with sine cosine algorithm. *Energies* **2020**, *13*, 215. [\[CrossRef\]](#)
18. Hussain, K.; Neggaz, N.; Zhu, W.; Houssein, E.H. An efficient hybrid sine-cosine Harris hawks optimization for low and high-dimensional feature selection. *Expert Syst. Appl.* **2021**, *176*, 114778. [\[CrossRef\]](#)
19. Rizk-Allah, R.M. Hybridizing sine cosine algorithm with multi-orthogonal search strategy for engineering design problems. *J. Comput. Des. Eng.* **2018**, *5*, 249–273. [\[CrossRef\]](#)
20. Zhao, M.; Hou, R.; Li, H.; Ren, M. A hybrid grey wolf optimizer using opposition-based learning, sine cosine algorithm and reinforcement learning for reliable scheduling and resource allocation. *J. Syst. Softw.* **2023**, *205*, 111801. [\[CrossRef\]](#)
21. Kamble, V.; Dale, M.; Bairagi, V. A Hybrid Model by Combining Discrete Cosine Transform and Deep Learning for Children Fingerprint Identification. *Int. J. Adv. Comput. Sci. Appl.* **2023**, *14*, 780–787. [\[CrossRef\]](#)
22. Yousef, A.M.; Ebeed, M.; Abo-Elyousr, F.K.; Elnozohy, A.; Mohamed, M.; Mohamed Abdelwahab, S.A. Optimization of PID controller for Hybrid Renewable Energy System using Adaptive Sine Cosine Algorithm. *Int. J. Renew. Energy Res.* **2020**, *10*, 669–677.
23. Bansal, S.; Wadhawan, S. A Hybrid of Sine Cosine and Particle Swarm Optimization (HSPS) for solving heterogeneous fixed fleet vehicle routing problem. *Int. J. Appl. Metaheuristic Comput.* **2021**, *12*, 41–65. [\[CrossRef\]](#)
24. Li, R.; Zhuang, Q.; Yu, N.; Li, R.; Zhang, H. Improved Hybrid Particle Swarm Optimizer with Sine-Cosine Acceleration Coefficients for Transient Electromagnetic Inversion. *Curr. Bioinform.* **2022**, *17*, 60–76. [\[CrossRef\]](#)
25. Begum, M.; Ferdush, J.; Uddin, M.S. A Hybrid robust watermarking system based on discrete cosine transform, discrete wavelet transform, and singular value decomposition. *J. King Saud Univ. Comput. Inf. Sci.* **2022**, *34*, 5856–5867. [\[CrossRef\]](#)
26. Khajezadeh, M.; Sobhani, A.; Alizadeh, S.M.S.; Eslami, M. A Novel Hybrid Particle Swarm Optimization and Sine Cosine Algorithm for Seismic Optimization of Retaining Structures. *Period. Polytech. Civ. Eng.* **2022**, *66*, 96–111. [\[CrossRef\]](#)
27. Hassan, A.T.; Banakhr, F.A.; Mahmoud, M.M.; Mosaad, M.I.; Rashwan, A.F.; Mosa, M.R.; Hussein, M.M.; Mohamed, T.H. Adaptive Load Frequency Control in Microgrids Considering PV Sources and EVs Impacts: Applications of Hybrid Sine Cosine Optimizer and Balloon Effect Identifier Algorithms. *Int. J. Robot. Control Syst.* **2024**, *4*, 941–957. [\[CrossRef\]](#)
28. Talebi, M.; Borhani, M.A.; Zadeh, S.M.; Ansarifard, G.R. Experimental analysis of forced convection heat transfer of Hybrid Nanofluids in a vertical annulus with cosine heat flux. *Prog. Nucl. Energy* **2022**, *153*, 104438. [\[CrossRef\]](#)
29. Hans, R.; Kaur, H. Hybrid binary Sine Cosine Algorithm and Ant Lion Optimization (SCALO) approaches for feature selection problem. *Int. J. Comput. Mater. Sci. Eng.* **2020**, *9*, 1950021. [\[CrossRef\]](#)
30. Song, H.; Wang, J.; Cheng, Z.; Chang, T. Hybrid improved sine cosine algorithm for mixed-integer nonlinear programming problems. *Soft Comput.* **2023**, *27*, 14909–14933. [\[CrossRef\]](#)
31. Gu, M.; Xie, R.; Jin, G.; Shao, L. A hybrid compression method for the NMR data based on window averaging and Discrete Cosine Transform. *Comput. Geosci.* **2021**, *157*, 104914. [\[CrossRef\]](#)
32. Rawa, M. Towards Avoiding Cascading Failures in Transmission Expansion Planning of Modern Active Power Systems Using Hybrid Snake-Sine Cosine Optimization Algorithm. *Mathematics* **2022**, *10*, 1323. [\[CrossRef\]](#)
33. Rosli, S.J.; Rahim, H.A.; Rani, K.N.A.; Ngadiran, R.; Ahmad, R.B.; Yahaya, N.Z.; Abdulmalek, M.; Jusoh, M.; Yasin, M.N.M.; Sabapathy, T.; et al. A hybrid modified method of the sine cosine algorithm using latin hypercube sampling with the cuckoo search algorithm for optimization problems. *Electronics* **2020**, *9*, 1786. [\[CrossRef\]](#)
34. Naik, J.; Dash, P.; Bisoi, R. Optimized droop controller based energy management for stand-alone micro-grid using hybrid monarch butterfly and sine-cosine algorithm. *Sustain. Energy Technol. Assess.* **2021**, *46*, 101310. [\[CrossRef\]](#)
35. Gupta, S.; Deep, K. Hybrid sine cosine artificial bee colony algorithm for global optimization and image segmentation. *Neural Comput. Appl.* **2020**, *32*, 9521–9543. [\[CrossRef\]](#)
36. Beşkirli, A.; Dağ, İ. I-CPA: An Improved Carnivorous Plant Algorithm for Solar Photovoltaic Parameter Identification Problem. *Biomimetics* **2023**, *8*, 569. [\[CrossRef\]](#) [\[PubMed\]](#)
37. Beşkirli, A.; Dağ, İ.; Kiran, M.S. A Multi-Strategy Tree Seed Algorithm for Parameter Estimation of Solar Photovoltaic Models. *Appl. Soft Comput.* **2024**, *167*, 112220. [\[CrossRef\]](#)

38. Luo, W.; Lin, X.; Li, C.; Yang, S.; Shi, Y. Benchmark functions for CEC 2022 competition on seeking multiple optima in dynamic environments. *arXiv* **2022**, arXiv:2201.00523.
39. Yazdani, D.; Branke, J.; Omidvar, M.N.; Li, X.; Li, C.; Mavrovouniotis, M.; Nguyen, T.T.; Yang, S.; Yao, X. IEEE CEC 2022 competition on dynamic optimization problems generated by generalized moving peaks benchmark. *arXiv* **2021**, arXiv:2106.06174.
40. Alzoubi, S.; Abualigah, L.; Sharaf, M.; Daoud, M.S.; Khodadadi, N.; Jia, H. Synergistic swarm optimization algorithm. *Comput. Model. Eng. Sci.* **2024**, *139*, 2557–2604. [[CrossRef](#)]
41. Falahah, I.A.; Al-Baik, O.; Alomari, S.; Bektemysova, G.; Gochhait, S.; Leonova, I.; Malik, O.P.; Werner, F.; Dehghani, M. Frilled Lizard Optimization: A Novel Nature-Inspired Metaheuristic Algorithm for Solving Optimization Problems. *Preprints* **2024**. [[CrossRef](#)]
42. Zhang, F.; Wu, S.; Cen, P. The past, present and future of the pangolin in Mainland China. *Glob. Ecol. Conserv.* **2022**, *33*, e01995. [[CrossRef](#)]
43. Jahn, J. (Ed.) *Vector Optimization*; Springer: Berlin/Heidelberg, Germany, 2009.
44. Fakhouri, H.N.; Hamad, F.; Alawamrah, A. Success history intelligent optimizer. *J. Supercomput.* **2022**, *78*, 6461–6502. [[CrossRef](#)]
45. Mohapatra, S.; Mohapatra, P. American zebra optimization algorithm for global optimization problems. *Sci. Rep.* **2023**, *13*, 5211. [[CrossRef](#)]
46. Bairwa, A.K.; Joshi, S.; Singh, D. Dingo optimizer: A nature-inspired metaheuristic approach for engineering problems. *Math. Probl. Eng.* **2021**, *2021*, 2571863. [[CrossRef](#)]
47. Jia, H.; Peng, X.; Lang, C. Remora optimization algorithm. *Expert Syst. Appl.* **2021**, *185*, 115665. [[CrossRef](#)]
48. Abualigah, L.; Yousri, D.; Abd Elaziz, M.; Ewees, A.A.; Al-Qaness, M.A.; Gandomi, A.H. Aquila optimizer: A novel meta-heuristic optimization algorithm. *Comput. Ind. Eng.* **2021**, *157*, 107250. [[CrossRef](#)]
49. Khishe, M.; Mosavi, M.R. Chimp optimization algorithm. *Expert Syst. Appl.* **2020**, *149*, 113338. [[CrossRef](#)]
50. Singh, A.; Sharma, A.; Rajput, S.; Mondal, A.K.; Bose, A.; Ram, M. Parameter extraction of solar module using the sooty tern optimization algorithm. *Electronics* **2022**, *11*, 564. [[CrossRef](#)]
51. Dhiman, G.; Kumar, V. Seagull optimization algorithm: Theory and its applications for large-scale industrial engineering problems. *Knowl.-Based Syst.* **2019**, *165*, 169–196. [[CrossRef](#)]
52. Wu, D.; Rao, H.; Wen, C.; Jia, H.; Liu, Q.; Abualigah, L. Modified sand cat swarm optimization algorithm for solving constrained engineering optimization problems. *Mathematics* **2022**, *10*, 4350. [[CrossRef](#)]
53. Mirjalili, S.; Mirjalili, S.M.; Hatamlou, A. Multi-verse optimizer: A nature-inspired algorithm for global optimization. *Neural Comput. Appl.* **2016**, *27*, 495–513. [[CrossRef](#)]
54. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [[CrossRef](#)]
55. Mirjalili, S. SCA: A sine cosine algorithm for solving optimization problems. *Knowl.-Based Syst.* **2016**, *96*, 120–133. [[CrossRef](#)]
56. Mirjalili, S. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowl. Based Syst.* **2015**, *89*, 228–249. [[CrossRef](#)]
57. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [[CrossRef](#)]
58. Wang, D.; Tan, D.; Liu, L. Particle swarm optimization algorithm: An overview. *Soft Comput.* **2018**, *22*, 387–408. [[CrossRef](#)]
59. Nikolaev, A.G.; Jacobson, S.H. Simulated annealing. In *Handbook of Metaheuristics*; Springer: Boston, MA, USA, 2010; pp. 1–39.
60. Mathew, T.V. *Genetic Algorithm*; IIT Bombay: Mumbai, India, 2012; Volume 53.

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.