




Article

A Bio-Inspired Method for Mathematical Optimization Inspired by Arachnida Salticidae

Hernán Peraza-Vázquez ^{1,*}, Adrián Peña-Delgado ^{2,*}, Prakash Ranjan ³, Chetan Barde ³, Arvind Choubey ³ and Ana Beatriz Morales-Cepeda ⁴

- ¹ Instituto Politécnico Nacional, Research Center for Applied Science and Advanced Technology (CICATA), km.14.5 Carretera Tampico-Puerto Industrial Altamira, Altamira 89600, Tamaulipas, Mexico
- ² Departamento de Mecatrónica y Energías Renovables, Universidad Tecnológica de Altamira, Boulevard de los Ríos km.3 + 100, Puerto Industrial Altamira, Altamira 89601, Tamaulipas, Mexico
- ³ Department of Electronics and Communication Engineering, Indian Institute of Information Technology Bhagalpur, Bhagalpur 813210, Bihar, India; pranjan.ece@iiitbh.ac.in (P.R.); 2017rsec004@nitjsr.ac.in (C.B.); achoubey.ece@nitjsr.ac.in (A.C.)
- ⁴ Division of Graduate Studies and Research, Instituto Tecnológico de Ciudad Madero (TecNM), Juventino Rosas y Jesús Urueta s/n, Col. Los Mangos, Cd. Madero 89318, Tamaulipas, Mexico; ana.mc@cdmadero.tecnm.mx
- * Correspondence: hperaza@ipn.mx (H.P.-V.); apea@utaltamira.edu.mx (A.P.-D.)

Abstract: This paper proposes a new meta-heuristic called Jumping Spider Optimization Algorithm (JSOA), inspired by Arachnida Salticidae hunting habits. The proposed algorithm mimics the behavior of spiders in nature and mathematically models its hunting strategies: search, persecution, and jumping skills to get the prey. These strategies provide a fine balance between exploitation and exploration over the solution search space and solve global optimization problems. JSOA is tested with 20 well-known testbench mathematical problems taken from the literature. Further studies include the tuning of a Proportional-Integral-Derivative (PID) controller, the Selective harmonic elimination problem, and a few real-world single objective bound-constrained numerical optimization problems taken from CEC 2020. Additionally, the JSOA's performance is tested against several well-known bio-inspired algorithms taken from the literature. The statistical results show that the proposed algorithm outperforms recent literature algorithms and is capable to solve challenging real-world problems with unknown search space.

Keywords: bio-inspired algorithm; meta-heuristics; constrained optimization; global optimization



Citation: Peraza-Vázquez, H.; Peña-Delgado, A.; Ranjan, P.; Barde, C.; Choubey, A.; Morales-Cepeda, A.B. A Bio-Inspired Method for Mathematical Optimization Inspired by Arachnida Salticidae. *Mathematics* **2022**, *10*, 102. <https://doi.org/10.3390/math10010102>

Academic Editor: Akemi Galvez Tomida

Received: 16 November 2021

Accepted: 23 December 2021

Published: 29 December 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 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

Currently, meta-heuristic algorithms are widely used as one of the main techniques to obtain an optimal solution (near to optimal) in various complex problems in several engineering and scientific research areas. Practically, it is possible to use them to solve any linear or non-linear optimization problem through one or several objective functions subject to several intrinsic restrictions. For example, they are quite useful in solving very complex problems where deterministic algorithms get caught up in a local optimum. Nowadays, meta-heuristics have become effective alternatives for solving NP-hard problems due to their versatility to find several local solutions, as in real-world applications, e.g., optimal design of structural engineering problems [1,2], logistics and industrial manufacture [3], renewable energy systems [4], Deep Neural Networks (DNNs) models optimization [5], among other applications. Metaheuristics generate several search agents, through stochastic processes within the solution space to find the optimal or close to the optimal value. The goal is to achieve efficient exploitation (intensification) and exploration (diversification) of the search space. Where, exploitation and exploration are associated with local and global search, respectively. In any metaheuristic design a fine balance of these search processes

should be a desired objective. Metaheuristics are not perfect, one weakness is that the quality of the solution depends on the number of search agents and the stop condition of the algorithm, commonly determined by the number of iterations.

On the other hand, the scientific community continues developing new metaheuristic algorithms due to its inability to get good results in all engineering and research fields. Good results can be obtained by optimizing specific problems in a particular field, but failing to find the global optimum in other fields [6,7].

Many metaheuristics have been developed in recent years; they can be classified into the following categories:

Swarm-Based Algorithms, which include Particle Swarm Optimization (PSO) [8], Salp Swarm Algorithm (SSA) [9], Whale Optimization Algorithm (WOA) [10], Chameleon Swarm Algorithm (CSA) [11], Orca Predation Algorithm (OPA) [12], African Vultures Optimization Algorithm (AVOA) [13], Dingo Optimization Algorithm (DOA) [14], Black widow Optimization Algorithm (BWOA) [15], Coot Bird Algorithm (COOT) [16], Mexican Axolotl Optimization (MAO) [17], Golden Eagle Optimizer (GEO) [18], Ant Lion Optimizer (ALO) [19], Coronavirus Optimization (CRO) [20], Archimedes Optimization algorithm (AOA) [21], Arithmetic Optimization Algorithm (ArOA) [22], Gradient-base Optimizer (GBO) [23], Hunger Game Search (HGS) [24], Henry Gas Solubility Optimization (HGSO) [25], Harris Hawks Optimization (HHO) [26], among others.

Evolutionary Algorithms, which include Differential evolution (DE) [27], Genetic Algorithm (GA) [28], Genetic Programming (GP) [29], Biogeography based Optimizer (BBO)[30], among others.

Physics-Based Algorithms, which include Multi-verse Optimization (MVO) [31], Water Wave optimization (WWO) [32], Thermal Exchange Optimization (TEO) [33], Cyclical Parthenogenesis Algorithm (CPA) [34], Magnetic Charged System Search (MCSS) [34], Colliding Bodies Optimization (CBO) [34], among others.

Human-Based Algorithms are Harmony Search (HS) [35,36], Ali Baba and the forty thieves algorithm (AFT) [37], Firework Algorithm (FWA) [38,39], Soccer-Inspired (SI) [40], among others.

Math's Based Algorithms, which are Sine Cosine Algorithm (SCA) [41], Chaos Game Optimization (CGO) [42], Stochastic Fractal Search (SFS) [43], Hyper-Spherical Search (HSS)[44] algorithm, are among the most prominent.

The generic structure of a swarm-based bio-inspired algorithm is depicted in Figure 1. It consists of four main phases, described as follows:

Phase 1: A set of vectors is randomly generated; the cardinality of the set is called the size of the initial population. The length of the vector is the number of variables in the problem (dimension). The vector represents the search agent, e.g., they can be used to represent some animals (reptiles, mammals, birds, amphibians, or insects), even some physical or chemical phenomenon. This initial population will evolve on each iteration.

Phase 2: Vector recombination functions (search agents). In this phase, the mathematical model that represents the behavior of the modeled living being is proposed, e.g., hunting, breeding, mating. Furthermore, the model must have a good balance (good ratio between) between exploration (diversification) and exploitation (intensification) of the search solution space (over the search solution space).

Phase 3: In this phase, the set of vectors are evaluated in the objective function, with or without restrictions, where the result of the evaluation of each vector is called fitness, for instance, the lowest fitness value is the best vector when the objective is to minimize.

Phase 4: The best fitness of the previous iteration (generation) is compared with the best fitness of the current iteration. If an obtained value exceeds (minimize) the best fitness obtained so far, the value and its respective vector are updated. In this way, in each iteration in the worst case, the fitness will be equal to that of the previous iteration, but it will never get worse. It is essential to highlight that the performance of a bio-inspired algorithm can be affected by the size of the population and the number of iterations.

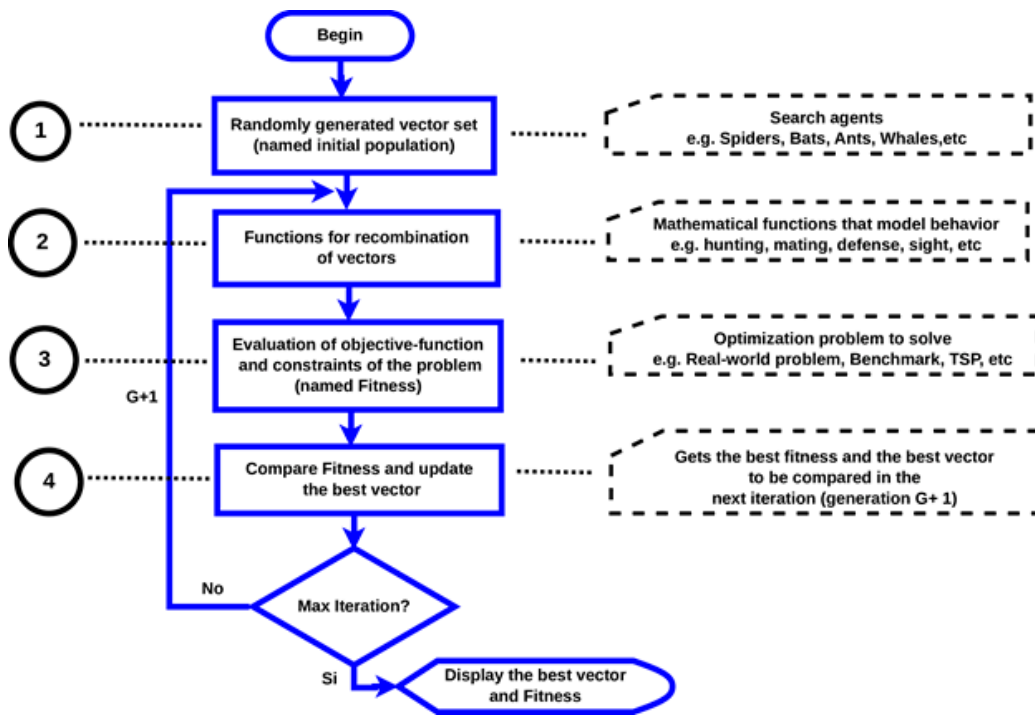


Figure 1. General structure of a bio-inspired algorithm.

Here, a new population-based bio-inspired algorithm, namely Jumping Spider Optimization Algorithm (JSOA), is proposed for solving optimization problems. JSOA mathematically models the persecution, search and jumping on the prey hunting strategies of jumping spiders. The rest of the paper is arranged as follows:

Section 2 illustrates the JSOA details, including the inspiration, mathematical model, time complexity, and a population size analysis. The proficiency and robustness of the proposed approach are benchmarked with several numerical examples, and their comparison with state-of-the-art metaheuristics are presented in Section 3. The results and discussion of the proposed algorithm are shown in Section 4. Real-world applications are solved in Section 5. The final section of the paper summarizes the conclusions and future work.

2. Jumping Spider Optimization Algorithm (JSOA)

In this section, the inspiration of the proposed method is first discussed. Then, the mathematical model is provided.

2.1. Biological Fundamentals

Jumping spiders Salticidae are found in a wide range of microhabitats [45]. Family Salticidae is the largest family of spiders; they are agile and dexterous jumpers of a few millimeters in length. They move at high speed and are capable of long and accurate jumps. They are generally active diurnal hunters and do not build webs. Their body appears to be covered with hairs that are both scaly, sometimes iridescent. The palps of males, but not females, are often large and showy, used during courtship. The front legs are somewhat larger and used to hold the prey when they fall on it. Thus, running and jumping on the prey are their primary hunting method. When they move from one side to the other and especially before jumping, they glue a silk thread where they are perched to protect themselves in case the jump fails. In that case, they climb back up the thread. The silk threads are impregnated with pheromones that play a role in reproductive and social—communication and possibly navigation. In addition, jumping spiders have complex eyes and acute vision. They are known for their bright colors and elaborate ornamentation, where males are generally brighter than females. Some species can remember and recognize

colors and adapt their hunting style accordingly. Further details about the Jumping Spiders' biology and locomotion behavior can be found in [46,47]. A photograph of the jumping spider can be seen in Figure 2.

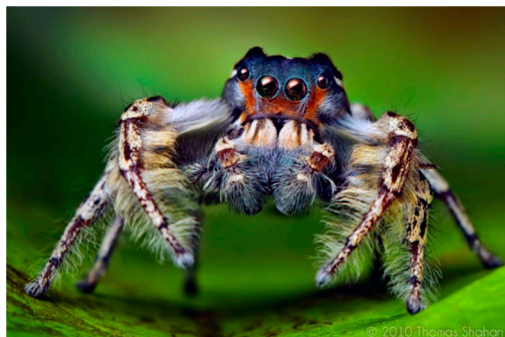


Figure 2. Salticidae Jumping Spider. Photography by Thomas Shahan (published under a CC BY 2.0 license).

2.2. Mathematical Model and Optimization Algorithm

In this section, the mathematical model of Jumping spider's different hunting strategies is first provided. The JSOA algorithm is then proposed. The hunting strategies considered are attacking by persecution, search, and Jumping on the prey. In addition, a model is also considered to represent the pheromone rate of the spider.

2.2.1. Strategy 1: Persecution

When the spider is not within a distance where it can catch its prey by jumping, it will move closer by doing some stealthy movements until it is at an achievable distance where it can jump and catch the prey. The persecution strategy can be represented by the uniformly accelerated rectilinear motion, see Equation (1). Thereby, the spider moves along the coordinate axis and the velocity increases (or decreases) linearly with time with constant acceleration.

$$x_i = \frac{1}{2}at^2 + v_0t \quad (1)$$

where x_i shows the position of i th follower spider, t is the time, v_0 is the initial speed. The acceleration is given by $a = \frac{v}{t}$, where $v = x - x_0$.

Here, for the optimization, each iteration is considered as the time, where the difference between iterations is equal to 1 and the initial speed is set to zero, $v_0 = 0$. Thereby, Equation (1) can be re-defined as follows:

$$x_i(g+1) = \frac{1}{2}(\vec{x}_i(g) - \vec{x}_r(g)) \quad (2)$$

where $\vec{x}_i(g+1)$ is the new position of a search agent (jumping spider) for the generation $g+1$, $\vec{x}_i(g)$ is the current i th search agent in generation g , and $\vec{x}_r(g)$ is the r th search agent randomly selected, with $i \neq r$, where r is a random integer number generated in the interval from 1 to the size of a maximum of search agents. A representation of this strategy can be seen in Figure 3.

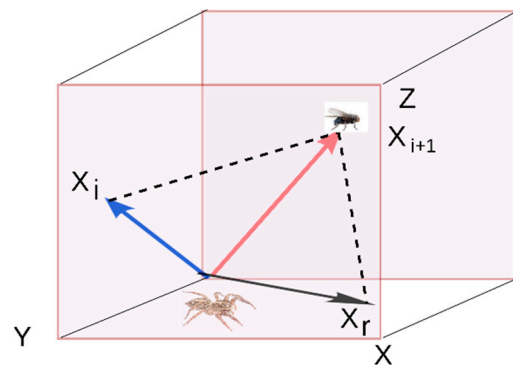


Figure 3. Representation of Jumping Spider’s persecution strategy.

2.2.2. Strategy 2: Jumping on the Prey

The jumping spider follows its prey and pounces on it. The hunting strategy of jumping on the prey can be represented as a projectile motion; see Figure 4.

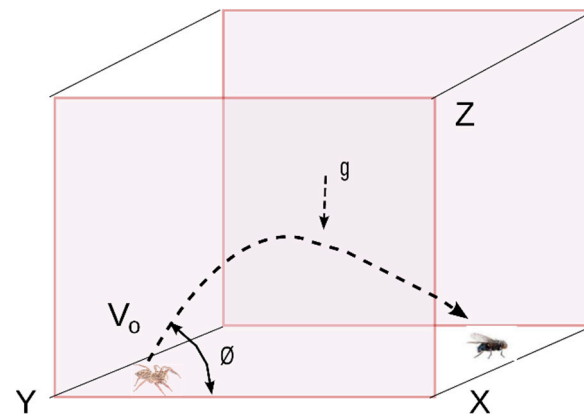


Figure 4. Jumping on the prey [47].

The equations of projectile motion, which result from the composition of a uniform motion along the X-axis and a uniformly accelerated motion along the Y-axis, are as follows: The horizontal axis and its derivative with respect to time is represented by Equation (3).

$$\begin{aligned} \vec{x}_i &= v_o \cos(\alpha)t \vec{i} \\ \frac{dx}{dt} &= \vec{V}_x = v_o \cos(\alpha) \vec{i} \end{aligned} \tag{3}$$

Similarly, the vertical axis, and its derivative with respect to time is represented by Equation (4).

$$\begin{aligned} \vec{y}_i &= \left(v_o \sin(\alpha)t - \frac{1}{2}gt^2 \right) \vec{j} \\ \frac{dy}{dt} &= \vec{V}_y = (v_o \sin(\alpha) - gt) \vec{j} \end{aligned} \tag{4}$$

The time is represented similarly to strategy 1. Thereby, we obtain the equation of the trajectory, as seen in Equation (5).

$$y = x \tan(\alpha) - \frac{gx^2}{2V_o^2 \cos^2(\alpha)} \tag{5}$$

Finally, the trajectory depicted in Figure 4 can be expressed as follows:

$$\begin{aligned} \vec{x}_i(g+1) &= \vec{x}_i(g) \tan(\alpha) - \frac{g \vec{x}_i^2(g)}{2V_0^2 \cos^2(\alpha)} \\ \alpha &= \frac{\phi\pi}{180} \end{aligned} \tag{6}$$

where $\vec{x}_i(g+1)$ is the new position of a search agent, indicating jumping spiders' movement), $\vec{x}_i(g)$ is the current i th search agent, V_0 is set to 100 mm/seg, g is the gravity (9.80665 m/s²), and the α angle is calculated by an ϕ angle value randomly generated between (0, 1).

2.2.3. Strategy 3: Searching for Prey

The Jumping spider performs a random search around the environment to locate prey. Two mathematical functions, local and global search, are proposed for the modeling of this search technique, see Figure 5.

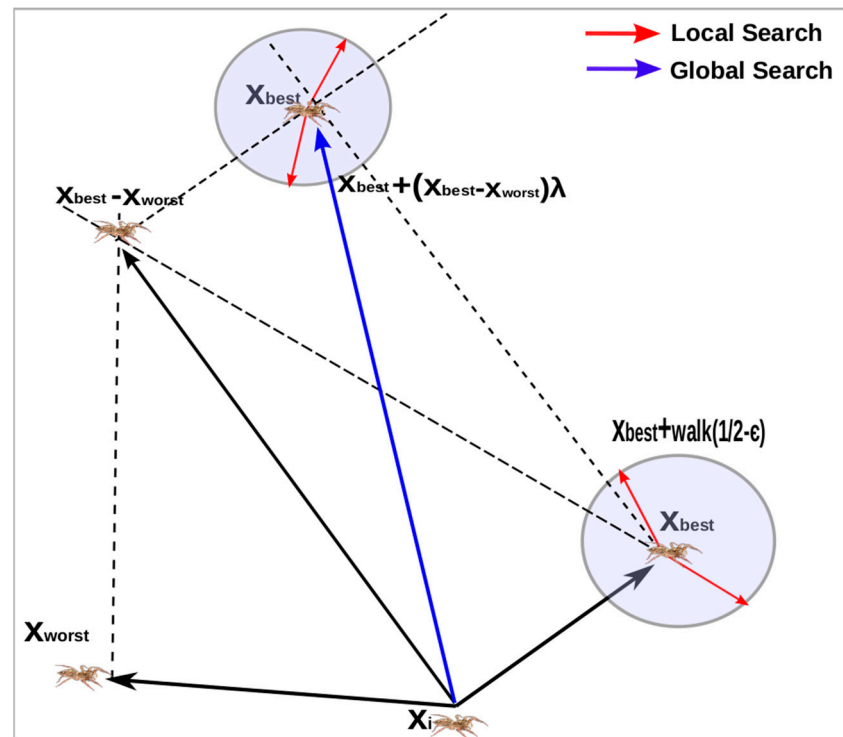


Figure 5. Local and global search.

The local search is described in Equation (7)

$$\vec{x}_i(g+1) = \vec{x}_{best}(g) + walk\left(\frac{1}{2} - \epsilon\right) \tag{7}$$

where $\vec{x}_i(g+1)$ is the new position of a search agent, $\vec{x}_{best}(g)$ is the best search agent found from the previous iteration, $walk$ is a pseudo-random number uniformly distributed in the interval of $(-2, 2)$, whereas ϵ is a normally distributed pseudo-random number in the interval of $(0, 1)$.

On the other hand, the Global search is formulated by Equation (8).

$$\vec{x}_i(g+1) = \vec{x}_{best}(g) + \left(\vec{x}_{best}(g) - \vec{x}_{worst}(g)\right)\lambda \tag{8}$$

where, $\vec{x}_i(g+1)$ is the new position of a search agent, $\vec{x}_{best}(g)$ and $\vec{x}_{worst}(g)$ are the best and worst search agent found from the previous iteration, respectively, and λ is a Cauchy random number with μ set to 0 and θ set to 1.

2.2.4. Strategy 4: Jumping Spider' Pheromone Rates

The pheromones are chemical substances produced and secreted to the outside by an individual, they are olfactory perceived by other individuals of the same species, and they cause a behavior change. Pheromones are produced by many animals, among which are insects, including spiders. In some spiders, like black widow spiders, the pheromone has a notable role in the courtship-mating. Whereas, in the Jumping spider the courtship-mating is based on its striking colors. Nevertheless, they also produce pheromones; the modeling of the rate of pheromones is taken from [15] and defined in the following equation:

$$pheromone(i) = \frac{Fitness_{max} - Fitness(i)}{Fitness_{max} - Fitness_{min}} \quad (9)$$

where $Fitness_{max}$ and $Fitness_{min}$ are the worst and the best fitness value in the current generation, respectively, whereas $Fitness(i)$ is the current fitness value of the i th search agent. Equation (9) normalize the fitness value in the interval (0, 1) where 0 is the worst pheromone rate, whereas 1 is the best.

The criteria consist that for low pheromones rates values equal or less than 0.3, the following equation is then applied [15]:

$$\vec{x}_i(g) = \vec{x}_{best}(g) + \frac{1}{2} \left(x_{r_1}^{\rightarrow}(g) - (-1)^{\sigma} x_{r_2}^{\rightarrow}(g) \right) \quad (10)$$

where $\vec{x}_i(g)$ is the search agent (jumping spider) with low pheromone rate that will be updated, r_1 and r_2 are random integer numbers generated in the interval from 1 to the maximum size of search agents, with $r_1 \neq r_2$, whereas $\vec{x}_{r_1}^{\rightarrow}(g)$ and $\vec{x}_{r_2}^{\rightarrow}(g)$ are the r_1, r_2 th search agents selected, $\vec{x}_{best}(g)$ is the best search agent found from the previous iteration and σ is a binary number generated, $\sigma \in \{0, 1\}$. The pheromone procedure is shown in Algorithm 1.

Algorithm 1 Pheromone procedure

```

1: Begin procedure
2: Compute pheromone rate for all spiders (search agents) by Equation (9)
3: for  $i = 1$  to sizePopulation do
4:   if  $pheromone(i) \leq 0.3$  then
5:     search agent updated by Equation (10)
6:   end if
7: end for
8: return  $\vec{x}$ 
9: End procedure

```

2.2.5. Pseudo Code for JSOA

The pseudocode for the JSOA is explained in Algorithm 2, whereas the overall flow is shown in Figure 6.

Algorithm 2 Jumping Spider Optimizer Algorithm

```

1: Procedure JSOA
2: Generate the initial population randomly (A set of spiders, search agents)
3: while iteration < Max Number of Iterations do
4:   if random < 0.5 then Attack or Search?
5:     if random < 0.5 then
6:       Strategy 1: Attack by persecution, Equation (2)
7:     Else
8:       Strategy 2: Attack by jumping on the prey, Equation (6)
9:     End if
10:  else
11:    if random < 0.5 then
12:      Strategy 3: Search for prey by local search, Equation (7).
13:    else
14:      Strategy 3: Search for prey by global search, Equation (8).
15:    end if
16:  end if
17: Update search agents that have low pheromone rate (computed by Equations (9) and (10)).
    See Algorithm 1.
18: Calculate  $x_{new}$ , the fitness value of the search agents
19: if  $x_{new} < x_*$  then
20:    $x_* = x_{new}$ 
21: end if
22: Iteration = Iteration + 1
23: end while
24: Display  $x_*$ , the best optimal solution
25: End procedure
    
```

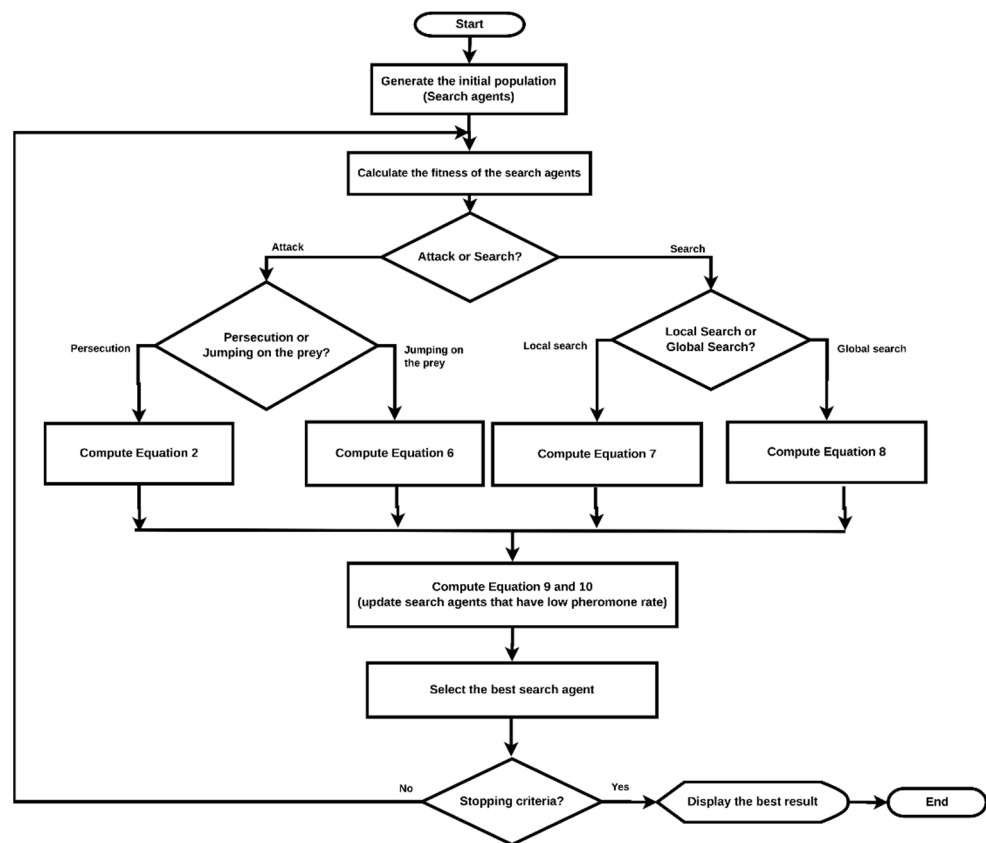


Figure 6. JSOA flowchart.

2.3. JSOA Algorithm Analysis

In this section, an in-depth analysis of the JSOA algorithm is carried out. This analysis includes the JSOA’s time complexity and the associated effects of the population size in the algorithm’s performance.

2.3.1. Time Complexity

Without any loss of generality, let f be any optimization problem and suppose that $O(f)$ is the computational time complexity of evaluating its function value. Thereby, the JSOA computational time complexity is defined as $O(f * tMax * nSpiders)$, where $tMax$ is the maximum number of iterations, and $nSpiders$ is the number of spiders (population size).

2.3.2. Population Size Analysis

The effects of the population size on the performance of the JSOA algorithm are studied by fixing the number of iterations to 100 and then varying the population size initially at 50, then 100, 200, and 500 for the Griewank function (F2). The output of these tests is summarized in Figure 7, where a convergence graph is used to determine whether the measured quantity (fitness) converged acceptably. The results were graphically compared with the setup mentioned above. Whereas Figure 8 shows the JSOA algorithm compared with the four most recent bio-inspired algorithms from the first half-year of 2021, these are Coot Bird Algorithm (COOT) [16], Chaos Game Optimization (CGO) [42], Mexican Axolotl Optimization (MAO) [17], and Golden Eagle Optimizer (GEO) [18].

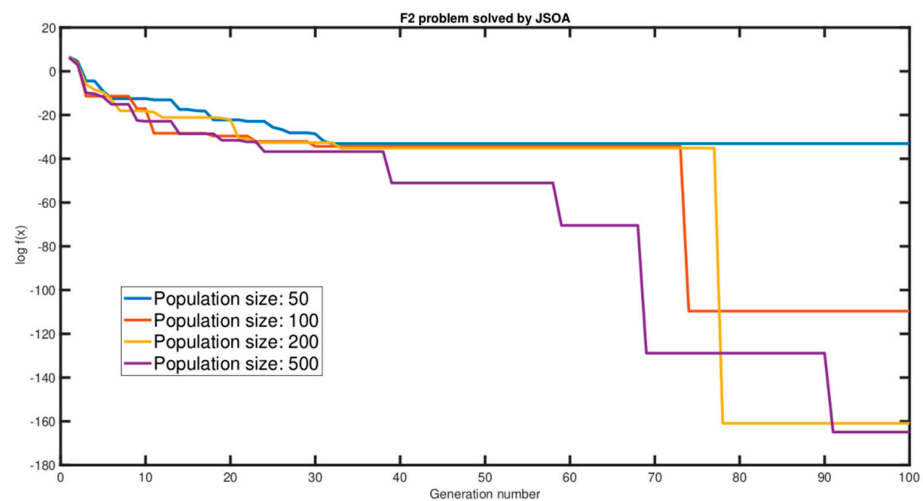


Figure 7. Convergence graph of population size analysis.

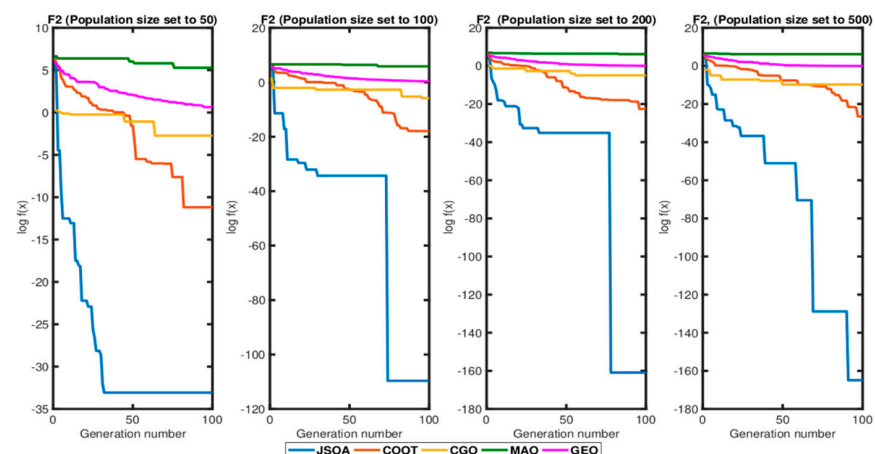


Figure 8. JSOA, COOT, CGO, MAO, and GEO Convergence graph population size analysis.

Additionally, the algorithms were tested as micro-algorithms with a reduced number of iterations and population size of 30 and 50 respectively. Figure 9 shows the results of this analysis.

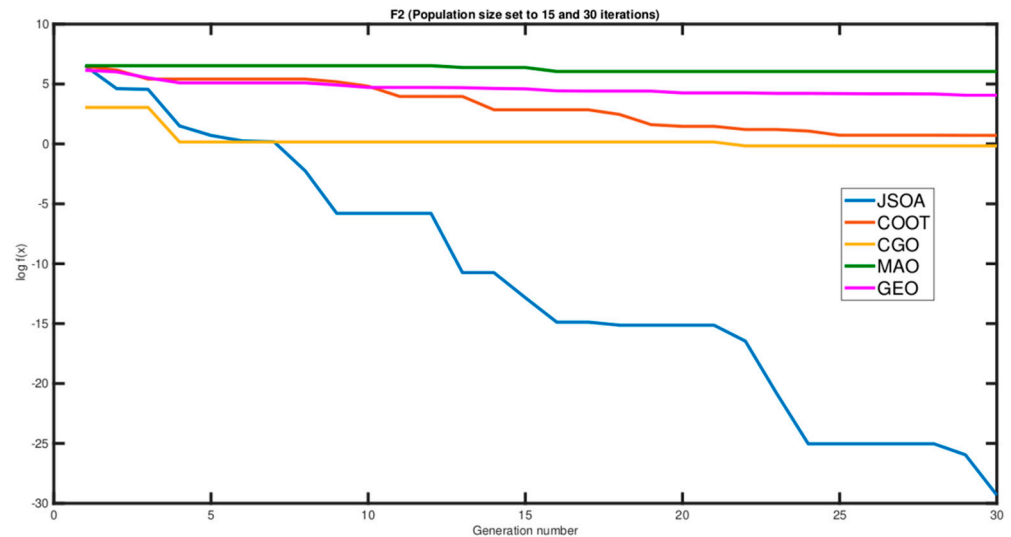


Figure 9. Convergence graph of the population size analysis as micro-algorithms.

Notice that JSOA outperforms all the other algorithms, even as a micro-algorithm. Based on this, for the rest of the paper, JSOA tests will be conducted with a population size of 200 for testbench functions and 100 for real-world problems.

3. Experimental Setup

The numerical efficiency, effectiveness, and stability of the JSOA algorithm developed in this study were tested by solving 20 classical well-known benchmark optimization functions reported in the literature [48]. The classification of the testbench functions can be observed in Table A1. The unimodal functions allow testing the exploitation ability (local search) since they only have one global optimum [14]. In contrast, multimodal functions can test the exploration ability (global search) since they include many local optima [14]. Table A2 summarizes these testbench functions where Dim indicates the dimension of the function, Interval is the boundary of the function's search space and f_{\min} is the optimum value. The shapes of the testbench functions considered in this study are shown in appendix A. The JSOA algorithm was compared with ten state-of-the-art bio-inspired algorithms taken from the literature; these are Coot Bird Algorithm (COOT) [16], Chaos Game Optimization (CGO) [42], Mexican Axolotl Optimization (MAO) [17], Golden Eagle Optimizer (GEO) [18], Archimedes Optimization algorithm (AOA) [21], Arithmetic Optimization Algorithm (ArOA) [22], Gradient-based Optimizer (GBO) [23], Hunger Game Search (HGS) [24], Henry Gas Solubility Optimization (HGSO) [25] and, Harris Hawks Optimization (HHO) [26]. For each testbench function, the eleven algorithms were run 30 times, the size of the population (search agents) and the number of iterations were set to 30 and 200, respectively. The entire parameter settings for all algorithms are shown in Table 1.

Table 1. Parameter settings.

Algorithms	Parameter	Value
for All Algorithms	Population Size for All Problems	30
	Maximum Iterations for Testbench Functions	200
	Number of Replications for Testbench Functions	30
	Maximum Iterations for Real-World Problems	100
JSOA	Does not use additional parameters	–
CGO	Does not use additional parameters	–
COOT	Does not use additional parameters	–
GEO	p_a : Propensity to attack p_c : Propensity to croise	[0.5, 2.0] [1.0, 0.5]
MAO	Crossover probability (cop)	0.5
	Damage probability (dp)	0.5
	Regeneration probability (rp)	0.1
	Tournament size (k) Lamba value (λ)	2 0.5
AOA	Object Number $C_1 = 2, C_2 = 6$ $C_3 = 2$ and $C_4 = 0.5$ (CEC and engineering problems)	30
ArOA	α	5
	μ	0.5
GBO	β_{min}, β_{max}	0.2, 0.6
	pr	0.5
HGSO	Gases number	50
	Cluster number	5
	M1 and M2	0.1, 0.2
	$\beta, \alpha,$ and K	1, 1, 1
	l_1, l_2 and l_3 are constants fixed for benchmark function	$5 \times 10^{-3},$ 100,
	l_1, l_2 and l_3 are constants fixed for engineering problems.	1×10^{-2} 1, 10, 1
HHO	Harris Hawk Number E_0 variable changes from -1 to 1 (Default)	30
HGS	Does not use additional parameters	-

Our approach is implemented in MATLAB R2018a. All computations were carried out on a standard PC (Linux Kubuntu 20.04 LTS, Intel core i7, 2.50 GHz, 32 GB).

4. Results and Discussion

The comparison results are shown in Table 2, where f_{min} , the best, the mean and standard deviation values are shown. The convergence graphs of all functions versus all algorithms selected in this study are summarized in Figure 10. In order to investigate the significant differences between the results of the proposed JSOA and the other algorithms, the Wilcoxon rank-sum non-parametric statistical test with a 5% degree of significance was carried out. This statistical test returns a parameter called p -values that determines the significance level of two algorithms. In this experimentation, an algorithm is statistically significant if and only if the calculated p -value is less than 0.05. Table 3 summarizes the result of this test.

Table 2. JSOA—Performance and comparison.

F	f_{\min}	Algorithms														
		JSOA			COOT			CGO			MAO			GEO		
		Best	Ave	Std	Best	Ave	Std	Best	Ave	Std	Best	Ave	Std	Best	Ave	Std
F1	0	4.1113×10^{-76}	8.36×10^{-67}	2.74×10^{-66}	1.44×10^{-11}	2.62×10^{-11}	1.0×10^{-11}	1.36×10^{-8}	1.87×10^{-5}	4.4×10^{-5}	1.98×10^2	2.77×10^4	5.96×10^4	4.30×10^{-1}	7.05×10^{-1}	2.7×10^{-1}
F2	0	0.0000	0.0000	0.0000	1.32×10^{-8}	3.20×10^{-8}	1.61×10^{-7}	0.0000	5.54×10^2	9.16×10^{-2}	7.98×10	3.04×10^2	8.31×10	1.05×10^0	1.11×10	5.2×10^{-2}
F3	0	3.54×10^{-51}	5.54×10^{-39}	1.90×10^{-38}	1.9910×-5	4.13×10^{-4}	2.07×10^{-3}	2.32×10^{-11}	2.80×10^{-1}	3.60×10^{-1}	4.92×10^2	7.80×10^2	1.14×10^2	1.06×10	2.83×10	1.19×10
F4	0	0.0000	5.35×10^{-40}	2.02×10^{-39}	1.04×10^{-4}	3.31×10^{-5}	1.57×10^{-4}	5.22×10^{-2}	1.91×10^{-2}	1.81×10^{-2}	5.79×10	7.52×10	6.9105	3.1912	5.5714	1.3723
F5	0	4.448×10^{-45}	4.448×10^{-45}	1.390×10^{-39}	1.85×10^{-5}	5.55×10^{-1}	3.04123	4.97×10^{-14}	3.79×10^{-1}	3.80×10^{-1}	7.37×10^{26}	3.59×10^{38}	1.47×10^{39}	1.10×10^2	1.06×10^7	5.80×10^7
F6	0	0.0000	5.87×10^{-296}	0.0000	3.14×10^{-44}	1.05×10^{-45}	5.7×10^{-45}	1.77×10^{-24}	4.21×10^{-23}	2.1×10^{-22}	2.78×10^7	6.68×10^8	7.15×10^8	1.25×10^{-6}	2.50×10^{-3}	8.2×10^{-3}
F7	0	9.12×10^{-79}	3.61×10^{-66}	1.46×10^{-65}	3.43×10^{-11}	9.72×10^{-9}	5.30×10^{-8}	1.93×10^{-4}	4.02×10^{-5}	5.58×10^{-5}	3.91×10	9.40×10	2.57×10	1.53×10^{-2}	3.63×10^{-2}	2.3×10^{-2}
F8	0	1.57×10^{-80}	1.20×10^{-66}	4.30×10^{-66}	8.69×10^{-7}	2.90×10^{-8}	1.59×10^{-7}	1.17×10^{-3}	4.40×10^{-3}	1.54×10^{-2}	1.90×10^3	4.94×10^3	1.27×10^3	1.86×10	7.48×10	3.87×10
F9	-1	-1.0000	-1.0000	0.0000	-1.0000	-2.64×10^{-1}	9.92×10^{-1}	-1.0000	-1.0000	7.43×10^{-5}	9.99×10^{-1}	9.99×10^{-1}	1.36×10^{-4}	9.95×10^{-1}	9.95×10^{-1}	4.9×10^{-6}
F10	0	3.78×10^{-74}	4.93×10^{-62}	2.62×10^{-61}	2.98×10^{-10}	8.12×10	4.45×10	4.16×10^{-2}	9.55×10^{-1}	3.78×10	4.60×10^7	6.59×10^7	2.34×10^8	2.55×10	5.42×10	1.80×10
F11	0	-8.88×10^{-16}	-8.88×10^{-16}	0.0000	2.40×10^{-6}	1.82×10^{-4}	9.85×10^{-4}	-8.88×10^{-16}	2.28×10^{-2}	2.14×10^{-2}	1.78×10	1.91×10	7.81×10^{-1}	1.6900	2.4800	4.0×10^{-1}
F12	-4.59	-4.5900	-4.5900	1.29×10^{-15}	-4.5900	-4.5900	1.60×10^{-3}	-4.5900	-4.4200	2.24×10^{-1}	-4.0600	4.09×10^{-1}	2.8200	1.78×10^2	2.89×10^2	5.21×10
F13	0.9	0.9000	0.9000	4.52×10^{-16}	0.9000	1.5100	1.3400	0.9000	9.00×10^{-1}	3.39×10^{-4}	8.7700	9.5700	9.94×10^{-1}	1.9600	4.8400	1.8300
F14	0	1.70×10^{-6}	4.46×10^{-4}	4.01×10^{-4}	1.93×10^{-3}	1.53×10^{-2}	1.13×10^{-2}	2.21×10^{-4}	5.56×10^{-3}	4.07×10^{-3}	3.53×10	3.93×10	2.49×10	2.42×10^{-2}	4.94×10^{-2}	2.2×10^{-2}
F15	0	0.0000	0.0000	0.0000	1.05×10^{-5}	1.74×10^{-4}	9.41×10^{-4}	2.06×10^{-3}	4.84×10^{-3}	5.07×10^{-3}	2.89×10^2	3.45×10^2	3.68×10	2.98×10	7.61×10	3.62×10
F16	0	5.33×10^{-21}	3.32×10^{-1}	7.32×10^{-1}	2.88×10	3.17×10	9.6600	1.23×10^{-6}	2.49×10^{-3}	3.78×10^{-3}	2.36×10^4	7.18×10^4	3.02×10^4	3.14×10	3.99×10	1.05×10
F17	0	2.17×10^{-45}	3.40×10^{-38}	4.32×10^{-38}	9.99×10^{-2}	7.56×10^{-2}	4.32×10^{-2}	5.21×10^{-9}	8.54×10^{-2}	8.10×10^{-2}	1.43×10	1.83×10	2.5200	1.1100	1.6000	2.3×10^{-1}
F18	0	3.85×10^{-40}	3.16×10^{-28}	9.61×10^{-28}	1.73×10^{-3}	7.43×10^{-5}	3.17×10^{-4}	2.26×10^{-4}	2.63×10^{-4}	3.60×10^{-4}	1.28×10^7	5.60×10^{11}	2.68×10^{12}	4.83×10^{-4}	1.07×10^{-2}	2.5×10^{-2}
F19	0	0.0000	1.17×10^{-13}	6.41×10^{-13}	1.91×10^{-11}	7.39×10^{-12}	5.4×10^{-12}	3.51×10^{-12}	3.51×10^{-12}	1.1×10^{-15}	1.54×10^{-4}	5.31×10^{-4}	6.99×10^{-4}	3.01×10^{-11}	4.78×10^{-10}	2.4×10^{-9}
F20	-1	-1.0000	-1.0000	0.0000	-9.99×10^{-1}	-4.91×10^{-1}	4.88×10^{-1}	-1.0000	-9.55×10^{-1}	4.19×10^{-2}	3.31×10^{-9}	4.35×10^{-9}	3.71×10^{-9}	6.08×10^{-13}	6.60×10^{-13}	2.1×10^{-13}

F	Algorithms								
	AOA			GBO			HGSO		
	Best	Ave	Std.	Best	Ave	Std.	Best	Ave	Std.
F1	6.40×10^{-35}	1.87×10^{-33}	6.02×10^{-33}	1.11×10^{-44}	1.13×10^{-42}	5.34×10^{-42}	1.14×10^{-72}	1.27×10^{-61}	6.97×10^{-61}
F2	0.0000	4.17×10^{-3}	2.29×10^{-2}	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
F3	5.84×10^{-17}	8.53×10^{-17}	2.64×10^{-16}	3.96×10^{-21}	3.35×10^{-17}	8.92×10^{-17}	5.04×10^{-41}	1.53×10^{-33}	8.09×10^{-33}
F4	6.15×10^{-19}	2.81×10^{-16}	7.59×10^{-16}	1.07×10^{-17}	2.73×10^{-17}	1.35×10^{-16}	3.00×10^{-31}	2.30×10^{-28}	1.26×10^{-27}

Table 2. Cont.

	Algorithms								
	AOA			GBO			HGSO		
	Best	Ave	Std.	Best	Ave	Std.	Best	Ave	Std.
F5	1.02×10^{-21}	4.53×10^{-17}	1.72×10^{-16}	1.59×10^{-20}	2.07×10^{-18}	4.19×10^{-18}	6.95×10^{-41}	5.22×10^{-35}	2.67×10^{-34}
F6	5.69×10^{-149}	1.88×10^{-147}	1.03×10^{-146}	2.21×10^{-178}	4.10×10^{-174}	0.0000	0.0000	0.0000	0.0000
F7	7.26×10^{-48}	3.96×10^{-34}	2.04×10^{-33}	7.29×10^{-45}	2.01×10^{-41}	7.12×10^{-41}	2.5010^{-77}	1.46×10^{-61}	7.99×10^{-61}
F8	3.03×10^{-43}	3.38×10^{-34}	9.76×10^{-34}	2.02×10^{-39}	2.45×10^{-37}	8.83×10^{-37}	5.72×10^{-75}	1.43×10^{-56}	7.86×10^{-56}
F9	0.0000	0.0000	0.0000	-1.0000	-1.0000	1.04×10^{-10}	0.0000	0.0000	0.0000
F10	6.89×10^{-12}	1.84×10^{-7}	1.01×10^{-6}	4.63×10^{-29}	4.75×10^{-25}	1.95×10^{-24}	3.45×10^{-29}	8.63×10^{-21}	4.72×10^{-20}
F11	2.00×10	1.93×10	3.6500	-8.88×10^{-16}	-8.88×10^{-16}	0.0000	-8.88×10^{-16}	-4.14×10^{-16}	1.23×10^{-15}
F12	-4.5100	-4.5500	5.00×10^{-2}	-4.5900	-4.5900	1.81×10^{-15}	-4.5900	-4.5100	2.22×10^{-1}
F13	6.0300	7.6100	1.6200	9.00×10^{-1}	9.11×10^{-1}	4.26×10^{-2}	8.4400	8.4700	3.0400
F14	1.33×10^{-3}	2.44×10^{-3}	1.58×10^{-3}	3.12×10^{-3}	3.02×10^{-3}	2.56×10^{-3}	3.37×10^{-4}	8.49×10^{-4}	7.63×10^{-4}
F15	1.70×10^2	5.6900	3.12×10	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
F16	2.89×10	2.89×10	8.71×10^{-2}	2.79×10	2.68×10	6.48×10^{-1}	2.77×10	2.86×10	3.41×10^{-1}
F17	2.00×10^{-1}	1.10×10^{-1}	3.05×10^{-2}	1.49×10^{-6}	6.76×10^{-3}	2.53×10^{-2}	9.99×10^{-2}	1.01×10^{-1}	9.53×10^{-4}
F18	1.18×10^{-10}	9.89×10^{-5}	4.14×10^{-4}	5.54×10^{-24}	5.48×10^{-11}	2.62×10^{-10}	4.22×10^{-33}	3.3810^{-28}	1.52×10^{-27}
F19	3.36×10^{-11}	2.90×10^{-11}	3.12×10^{-12}	3.53×10^{-12}	4.34×10^{-12}	1.89×10^{-12}	3.38×10^{-11}	3.38×10^{-11}	2.63×10^{-26}
F20	4.77×10^{-11}	6.09×10^{-11}	2.70×10^{-11}	-1.0000	-1.0000	1.08×10^{-3}	6.21×10^{-10}	4.07×10^{-10}	2.31×10^{-10}
	Algorithms								
	HHO			HGS			ArOA		
	Best	Ave	Std.	Best	Ave	Std.	Best	Ave	Std.
F1	7.70×10^{-51}	6.49×10^{-43}	3.48×10^{-42}	1.93×10^{-60}	8.17×10^{-54}	3.12×10^{-53}	2.31×10^{-1}	5.1800	1.8400
F2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.15×10^{-2}	4.90×10^{-1}	2.90×10^{-1}
F3	1.45×10^{-20}	3.01×10^{-20}	1.22×10^{-19}	1.95×10^{-26}	1.93×10^{-23}	5.41×10^{-23}	2.45×10^{-31}	7.30×10^{-12}	2.85×10^{-11}
F4	6.63×10^{-24}	1.12×10^{-20}	4.14×10^{-20}	3.17×10^{-20}	2.47×10^{-13}	1.35×10^{-12}	5.81×10^{-20}	3.28×10^{-2}	1.84×10^{-2}
F5	9.14×10^{-23}	3.76×10^{-19}	1.95×10^{-18}	2.66×10^{-26}	6.74×10^{-20}	6.74×10^{-20}	1.52×10^{-32}	1.43×10^{-10}	5.07×10^{-10}
F6	3.42×10^{-233}	4.96×10^{-212}	0.0000	0.0000	7.22×10^{-88}	3.96×10^{-87}	0.0000	0.0000	0.0000
F7	1.93×10^{-51}	3.26×10^{-41}	1.19×10^{-40}	9.00×10^{-63}	7.46×10^{-25}	4.08×10^{-24}	0.0000	0.0000	0.0000
F8	2.49×10^{-57}	5.81×10^{-42}	1.51×10^{-41}	1.57×10^{-68}	3.45×10^{-27}	1.89×10^{-26}	0.0000	3.40×10^{-205}	0.0000
F9	-1.0000	-1.0000	0.0000	-1.0000	-6.01×10^{-1}	8.12×10^{-1}	-1.0000	-7.34×10^{-1}	6.91×10^{-1}

Table 2. Cont.

	Algorithms								
	HHO			HGS			ArOA		
	Best	Ave	Std.	Best	Ave	Std.	Best	Ave	Std.
F10	1.43×10^{-22}	1.09×10^{-10}	4.45×10^{-10}	2.26×10^{-13}	5.50×10	9.98×10	2.11×10^2	3.89×10^2	9.14×10
F11	-8.88×10^{-16}	-8.88×10^{-16}	0.0000	-8.88×10^{-16}	2.90×10^{-15}	1.56×10^{-14}	-8.88×10^{-16}	-7.70×10^{-16}	6.49×10^{-16}
F12	-4.5900	-4.5900	3.69×10^{-5}	-4.5900	-4.5600	1.61×10^{-1}	-4.5900	-4.5900	1.46×10^{-5}
F13	9.00×10^{-1}	9.00×10^{-1}	4.52×10^{-16}	9.00×10^{-1}	9.00×10^{-1}	6.49×10^{-16}	9.00×10^{-1}	9.00×10^{-1}	4.52×10^{-16}
F14	3.28×10^{-4}	4.82×10^{-4}	6.36×10^{-4}	4.8610×-4	5.78×10^{-3}	9.35×10^{-3}	3.24×10^{-6}	1.42×10^{-4}	1.42×10^{-4}
F15	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
F16	1.06×10^{-2}	4.64×10^{-2}	6.23×10^{-2}	7.63×10^{-2}	1.82×10	1.26×10	2.87×10	2.88×10	3.76×10^{-2}
F17	4.48×10^{-23}	1.68×10^{-20}	7.96×10^{-20}	5.28×10^{-28}	1.43×10^{-2}	4.34×10^{-2}	9.99×10^{-2}	9.99×10^{-2}	1.88×10^{-7}
F18	2.34×10^{-29}	1.45×10^{-10}	7.83×10^{-10}	4.02×10^{-11}	2.61×10^{-3}	8.28×10^{-3}	0.0000	0.0000	0.0000
F19	3.51×10^{-12}	3.57×10^{-12}	2.40×10^{-13}	3.54×10^{-12}	1.38×10^{-11}	1.02×10^{-11}	1.58×10^{-7}	7.8610×10^{-5}	1.55×10^{-4}
F20	-1.0000	-1.0000	0.0000	-1.0000	-2.98×10^{-1}	4.63×10^{-1}	1.22×10^{-8}	1.43×10^{-7}	1.13×10^{-7}

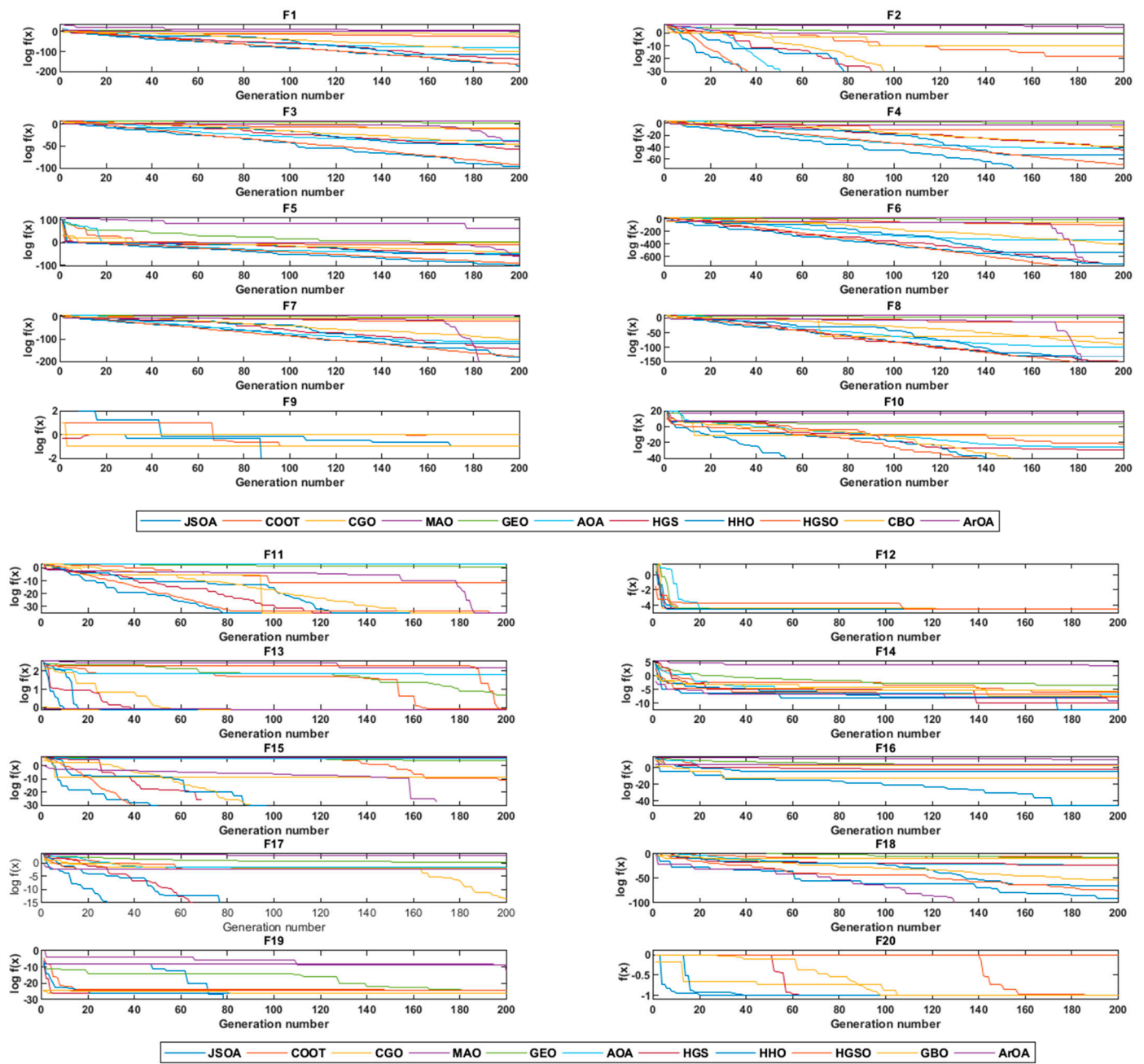


Figure 10. Convergence curves.

Moreover, to evaluate the quality of the results the Mean Absolute Error (MAE) criteria was used to measure the error between the best-known f_{\min} (fitness) versus the mean fitness computed as described in Equation (11). See Table A2.

$$MAE = \frac{\sum_{i=1}^n |m_i - p_i|}{n} \tag{11}$$

where m_i indicates the mean of the optimal values (computed), p_i is the corresponding global optimal value (observed), and n represents the number of test functions. Note that the arithmetic average of the absolute error is $e_i = |m_i - p_i|$. It shows how far the results are from actual values. The ten algorithms were ranked by computing their Mean Absolute Error (MAE). Table 4 shows the average error rates obtained in the 20 testbench functions,

while the ranking of all the algorithms based on their MAE calculations is illustrated in Table 5.

Table 3. The *p*-values of the Wilcoxon rank-sum test with 5% significance for JSOA vs. other algorithms (20 benchmark functions with 30 dimensions). NaN means “Not a Number” returned by the test.

	Algorithms									
	CGA	COOT	GEO	MAO	AOA	GBO	HGSO	HHO	HGS	ArOA
F1	1.929×10^{-3}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	1.09×10^{-10}	3.02×10^{-11}	1.018×10^{-1}	3.02×10^{-11}
F2	1.306×10^{-7}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	4.193×10^{-2}	NaN	NaN	NaN	NaN	1.21×10^{-12}
F3	1.791×10^{-2}	3.01×10^{-11}	3.01×10^{-11}	3.01×10^{-11}	3.01×10^{-11}	3.01×10^{-11}	2.708×10^{-2}	3.01×10^{-11}	1.253×10^{-2}	3.01×10^{-11}
F4	6.708×10^{-5}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.112×10^{-1}	3.02×10^{-11}	6.710×10^{-5}	3.02×10^{-11}
F5	2.195×10^{-2}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	6.669×10^{-3}	3.02×10^{-11}	3.261×10^{-1}	3.02×10^{-11}
F6	2.548×10^{-3}	1.249×10^{-6}	1.249×10^{-6}	1.249×10^{-6}	1.249×10^{-6}	1.249×10^{-6}	1.045×10^{-3}	1.249×10^{-6}	3.361×10^{-1}	1.152×10^{-4}
F7	3.947×10^{-4}	3.02×10^{-11}	3.02×10^{-11}	3.020×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	7.043×10^{-7}	3.02×10^{-11}	7.952×10^{-1}	1.21×10^{-12}
F8	3.947×10^{-4}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	5.07×10^{-10}	3.02×10^{-11}	9.941×10^{-1}	1.52×10^{-10}
F9	1.09×10^{-12}	3.90×10^{-13}	1.09×10^{-12}	1.21×10^{-12}	1.68×10^{-14}	NaN	5.63×10^{-13}	NaN	NaN	2.157×10^{-2}
F10	2.72×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	6.710×10^{-5}	3.02×10^{-11}
F11	1.13×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	4.61×10^{-13}	NaN	3.337×10^{-1}	NaN	3.337×10^{-1}	3.337×10^{-1}
F12	3.96×10^{-11}	2.33×10^{-11}	2.36×10^{-12}	2.95×10^{-12}	6.02×10^{-11}	1.608×10^{-1}	5.42×10^{-11}	1.124×10^{-9}	1.608×10^{-1}	1.019×10^{-9}
F13	5.852×10^{-9}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	2.158×10^{-2}	5.90×10^{-10}	NaN	NaN	NaN
F14	9.75×10^{-10}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	1.473×10^{-7}	7.088×10^{-8}	7.172×10^{-1}	3.711×10^{-1}	5.012×10^{-2}	9.514×10^{-6}
F15	3.453×10^{-7}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	4.193×10^{-2}	NaN	NaN	NaN	NaN	NaN
F16	1.413×10^{-1}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	6.100×10^{-1}	2.491×10^{-6}	3.02×10^{-11}
F17	9.460×10^{-6}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.351×10^{-5}	3.02×10^{-11}
F18	2.691×10^{-2}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.69×10^{-11}	3.02×10^{-11}	1.091×10^{-5}	8.89×10^{-10}	3.02×10^{-11}	1.21×10^{-12}
F19	1.891×10^{-6}	1.64×10^{-12}	1.72×10^{-12}	1.72×10^{-12}	1.72×10^{-12}	2.41×10^{-12}	2.70×10^{-14}	1.01×10^{-11}	8.68×10^{-11}	1.72×10^{-12}
F20	1.701×10^{-8}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.701×10^{-8}	1.21×10^{-12}	NaN	3.337×10^{-1}	1.21×10^{-12}

Table 4. Error rates results (20 benchmark functions).

	Algorithms										
	JSOA	COOT	CGO	MAO	GEO	AOA	GBO	HGSO	HHO	HGS	ArOA
F1	8.36×10^{-67}	2.62×10^{-12}	1.87×10^{-5}	2.77×10^4	7.05×10^{-1}	1.87×10^{-33}	1.13×10^{-42}	1.27×10^{-61}	6.49×10^{-43}	8.17×10^{-54}	5.1800
F2	0.0000	3.20×10^{-8}	5.54×10^{-2}	3.04×10^2	1.1100	4.17×10^{-3}	0.0000	0.0000	0.0000	0.0000	4.90×10^{-1}
F3	5.54×10^{-39}	4.13×10^{-4}	2.80×10^{-1}	7.80×10^2	2.83×10	8.53×10^{-17}	3.35×10^{-17}	1.53×10^{-33}	3.01×10^{-20}	1.93×10^{-23}	7.30×10^{-12}
F4	5.35×10^{-40}	3.31×10^{-5}	1.91×10^{-2}	7.52×10	5.5700	2.81×10^{-16}	2.73×10^{-17}	2.30×10^{-28}	1.12×10^{-20}	2.47×10^{-13}	3.28×10^{-2}
F5	7.90×10^{-40}	5.55×10^{-1}	3.79×10^{-1}	3.59×10^{38}	1.06×10^7	4.53×10^{-17}	2.07×10^{-18}	5.22×10^{-35}	3.76×10^{-19}	6.74×10^{-20}	1.43×10^{-10}
F6	5.87×10^{-296}	1.05×10^{-45}	4.2×10^{-23}	6.68×10^8	2.50×10^{-3}	1.88×10^{-147}	4.1×10^{-174}	0.0000	4.96×10^{-212}	7.22×10^{-88}	0.0000
F7	3.61×10^{-66}	9.72×10^{-9}	4.02×10^{-5}	9.40×10	3.63×10^{-2}	3.96×10^{-34}	2.01×10^{-41}	1.46×10^{-61}	3.26×10^{-41}	7.46×10^{-25}	0.0000
F8	1.20×10^{-66}	2.90×10^{-8}	4.40×10^{-3}	4.94×10^3	7.4800	3.38×10^{-34}	2.45×10^{-37}	1.43×10^{-56}	5.81×10^{-42}	3.45×10^{-27}	3.40×10^{-205}
F9	0.0000	7.36×10^{-1}	6.00×10^{-5}	2.0000	2.0000	1.0000	0.0000	1.0000	0.0000	3.99×10^{-1}	2.66×10^{-1}
F10	4.93×10^{-62}	8.1200	9.55×10^{-1}	6.59×10^7	5.42×10	1.84×10^{-7}	4.75×10^{-25}	8.63×10^{-21}	1.09×10^{-10}	5.50×10	3.89×10^2
F11	8.88×10^{-16}	1.82×10^{-4}	2.28×10^{-2}	1.91×10	2.4800	1.93×10	8.88×10^{-16}	4.14×10^{-16}	8.88×10^{-16}	2.90×10^{-15}	7.70×10^{-16}
F12	1.63×10^{-6}	3.02×10^{-4}	1.70×10^{-1}	5.0000	2.94×10^2	3.78×10^{-2}	1.63×10^{-6}	7.54×10^{-2}	1.63×10^{-6}	2.94×10^{-2}	1.63×10^{-6}
F13	0.0000	6.12×10^{-1}	2.40×10^{-4}	8.6700	3.9400	6.7100	1.07×10^{-2}	7.5700	0.0000	0.0000	0.0000
F14	4.46×10^{-4}	1.53×10^{-2}	5.56×10^{-3}	3.93×10	4.94×10^{-2}	2.44×10^{-3}	3.02×10^{-3}	8.49×10^{-4}	4.82×10^{-4}	5.78×10^{-3}	1.42×10^{-4}
F15	0.0000	1.74×10^{-4}	4.84×10^{-3}	3.45×10^2	7.61×10	5.6900	0.0000	0.0000	0.0000	0.0000	0.0000
F16	3.32×10^{-1}	3.17×10	2.49×10^{-3}	7.18×10^4	3.99×10	2.89×10	2.68×10	2.86×10	4.64×10^{-2}	1.82×10	2.88×10
F17	3.40×10^{-38}	7.56×10^{-2}	8.54×10^{-2}	1.83×10	1.6000	1.10×10^{-1}	6.76×10^{-3}	1.01×10^{-1}	1.68×10^{-20}	1.43×10^{-2}	9.99×10^{-2}
F18	3.16×10^{-28}	7.43×10^{-5}	2.63×10^{-4}	5.60×10^{11}	1.07×10^{-2}	9.89×10^{-5}	5.48×10^{-11}	3.38×10^{-28}	1.45×10^{-10}	2.61×10^{-3}	0.0000
F19	1.17×10^{-13}	7.39×10^{-12}	3.5×10^{-12}	5.31×10^{-4}	4.78×10^{-10}	2.90×10^{-11}	4.34×10^{-12}	3.38×10^{-11}	3.57×10^{-12}	1.38×10^{-11}	7.86×10^{-5}
F20	0.0000	5.09×10^{-1}	4.49×10^{-2}	1.0000	1.0000	1.0000	2.00×10^{-4}	1.0000	0.0000	7.02×10^{-1}	1.0000

Table 5. Rank of algorithms using MAE.

Algorithms	MAE	Rank
HHO	2.35×10^{-3}	1
JSOA	1.66×10^{-2}	2
CGO	1.01×10^{-1}	3
GBO	1.3400	4
HGSO	1.9200	5
COOT	2.1200	6
AOA	3.1400	7
HGS	3.7200	8
ArOA	2.12×10	9
GEO	5.31×10^5	10
MAO	1.80×10^{37}	11

According to the statistical results given in Table 2, the Jumping Spider Optimization Algorithm (JSOA) can provide overtopping results. In the exploitation analysis (unimodal functions), the JSOA outperforms all algorithms in the functions F1, F3, F4, F5, and F10. Whereas competitive results were found in the functions F2, F6, F7, F8 and F9 for the Algorithms Chaos Game Optimization (CGO), Archimedes Optimization algorithm (AOA), Gradient-base Optimizer (GBO), Hunger Game Search (HGS), Henry Gas Solubility Optimization (HGSO), Harris Hawks Optimization (HHO), and Arithmetic Optimization Algorithm (ArOA). Moreover, in the exploration analysis, multimodal functions, the JSOA outperforms all algorithms in functions F14, F16, F17, F18, and F19. Whereas it also shows competitive results with the COOT, CBO, HGSO, HHO, HGS, and ArOA, for F11, F12, F3, F15 and F20 functions. On the other hand, in Wilcoxon rank-sum test the p -values in Table 3 confirm the meaningful advantage of JSOA compared to other bio-inspired algorithms for many cases. Additionally, in the Mean Absolute Error (MAE) analysis, the JSOA algorithm appears ranked in the second position with a slight difference of 1.43×10^{-2} with the first position, as shown in Table 5.

In all the carried out tests, the number of iterations was set at 200, this being a lower value than that commonly used in the literature (500 or 1000) [16–18]. Therefore, the JSOA algorithm is more efficient in finding optimal (near-to optimal) solutions with a smaller number of iterations.

5. Real-World Applications

In this section, constrained optimization problems are considered. The JSOA algorithm was tested with four Real-World Single Objective Bound Constrained Numerical Optimization problems, Process Flow Sheeting, Process Synthesis, Optimal Design of an Industrial Refrigeration System and Welded Beam Design, taken from the CEC2020 special session [49].

On the other hand, the JSOA algorithm was also tested to find the optimal tuning parameters of a Proportional-Integral-Derivative (PID) controller and to solve the Selective Harmonics Elimination Problem, taken from [14,15], respectively.

For all the real-world application problems solved, JSOA was tested against COOT, CGO, MAO, GEO, and the best ranked in the MAE test (HHO), with population size and maximum iteration equal to 30 and 100, respectively.

5.1. Constraint Handling

The constraint handling method used is based on Penalization of Constraints (PCSt) [14] and in computing the Mean Constraint Violation (MCV) [49].

The PCSt handling is based on the penalization of infeasible solutions, that is at least one constraint is violated. This method is formulated in Equation (12) as follows:

$$F(\vec{x}) = \begin{cases} f(\vec{x}), & \text{if } v(\vec{x}) \leq 0 \\ f_{\max} + v(\vec{x}), & \text{otherwise.} \end{cases} \tag{12}$$

where f_{\max} is the fitness function value of the worst feasible solution in the population, whereas $f(\vec{x})$ is the fitness function value of a feasible solution and $v(\vec{x})$ is the value of MCV. The MCV handling is depicted in Equation (13).

$$v(\vec{x}) = \frac{\sum_{i=1}^p G_i(\vec{x}) + \sum_{j=1}^m H_j(\vec{x})}{p + m} \tag{13}$$

where $G_i(\vec{x})$ is the sum of the p inequality constraints, whereas $H_i(\vec{x})$ is the sum of the m equality constraints. $G_i(\vec{x})$ and $H_i(\vec{x})$ are formulated in Equations (14) and (15), respectively.

$$G_i(\vec{x}) = \begin{cases} 0, & \text{if } g_i(\vec{x}) \leq 0 \\ g_i(\vec{x}), & \text{otherwise.} \end{cases} \tag{14}$$

$$H_j(\vec{x}) = \begin{cases} 0, & \text{if } |h_j(\vec{x})| - \delta \leq 0 \\ |h_j(\vec{x})|, & \text{otherwise.} \end{cases} \tag{15}$$

In the inequality constraint $g_i(\vec{x})$ or the equality constraint $h_i(\vec{x})$ is not violated, a zero value is returned, else its self-value is returned (the value of the constraint is violated). That is to say, the constraint handling method used is based on the fitness of an infeasible solution punished by the worst feasible solution in the current population plus the mean value of the constraints violated. In Equation (15), the δ value is set to 0.0001.

5.2. Process Flow Sheetting Problem

This problem is formulated as a non-convex constrained optimization problem [49]. In this test, there are three decision variables with three inequality constraints. It is formulated as shown in Equation (16). The best known feasible objective function value is $f(\vec{x}) = 1.0765430833$.

$$\begin{aligned} & \text{Minimize} && f(\vec{x}) = -0.7x_3 + 5(0.5 - x_1)^2 + 0.8 \\ & \text{Subject to} && g_1(\vec{x}) = -e^{(x_1-0.2)} - x_2 \leq 0 \\ & && g_2(\vec{x}) = x_2 + 1.1x_3 \leq -1.0 \\ & && g_3(\vec{x}) = x_1 - x_3 \leq 0.2 \\ & \text{with bounds :} && 0.2 \leq x_1 \leq 1, -2.22554 \leq x_2 \leq 1, x_3 \in \{0, 1\} \end{aligned} \tag{16}$$

In Table 6, the comparison results show that JSOA, CGO, COOT, and HHO algorithms reported feasible solutions, whereas MAO and GEO are infeasible, as seen in the convergence graph in Figure 11. An infeasible solution does not satisfy one or more constraints, so it is not valid for the problem. The JSOA algorithm is ranked as the first best-obtained solution. The difference with the best-known feasible objective function value is 1.17347×10^{-5} .

Table 6. Process Flow Sheeting problem—Comparison Results.

Algorithm	Optimal Values for Variables			f_{\min}
	x_1	x_2	x_3	
JSOA	0.94194	−2.10000	1	1.076554818
MAO	0.22351	−1.0331	0	1.1822336005 (infeasible *)
CGO	0.20000	−1.0000	0	1.2500000000
COOT	0.20000	−1.0000	0	1.2500000000
GEO	0.36341	−1.5510	1	0.1932841405 (infeasible *)
HHO	0.20000	−1.0000	0	1.2500000000

* The solution does not satisfy one or more constraints.

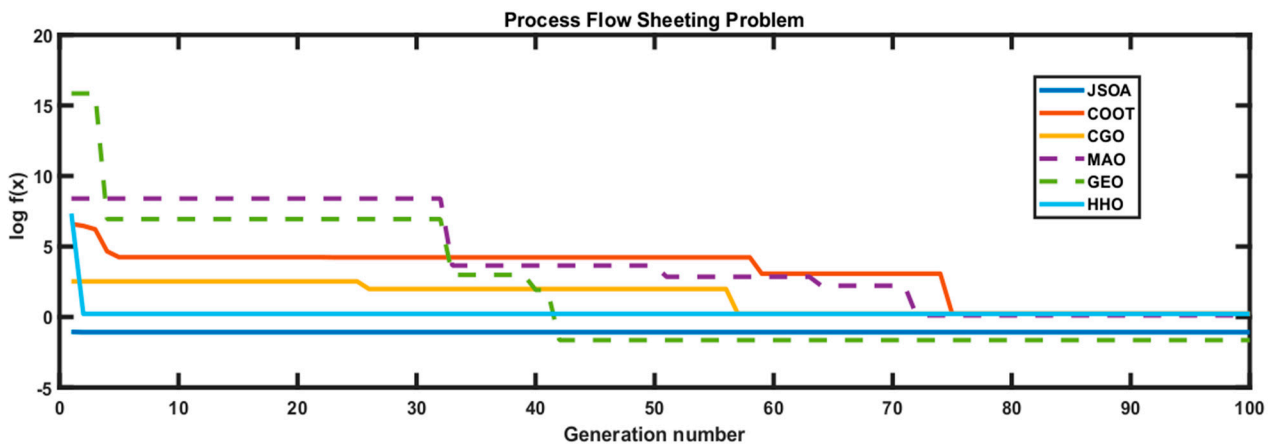


Figure 11. Convergence graph of the Process Flow Sheeting Problem. The dotted line represents an infeasible solution shown by an algorithm.

5.3. Process Synthesis Problem

This problem has non-linearities in real and binary variables [50]. Here, there are seven decision variables and nine inequality constraints. The mathematical formulation of the optimization problem is described in Equation (17). The best known feasible objective function value is $f(\vec{x}) = 2.9248305537$.

$$\begin{aligned}
 & \text{Minimize} && f(\vec{x}) = (1 - x_1)^2 + (2 - x_2)^2 + (3 - x_3)^2 + (1 - x_4)^2 + \\
 & && (1 - x_5)^2 + (1 - x_6)^2 - \ln(1 + x_7) \\
 & \text{Subject to} && g_1(\vec{x}) = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \leq 5 \\
 & && g_2(\vec{x}) = x_1^2 + x_2^2 + x_3^2 + x_6^2 \leq 5.5 \\
 & && g_3(\vec{x}) = x_1 + x_4 \leq 1.2 \\
 & && g_4(\vec{x}) = x_2 + x_5 \leq 1.8 \\
 & && g_5(\vec{x}) = x_3 + x_6 \leq 2.5 \\
 & && g_6(\vec{x}) = x_1 + x_7 \leq 1.2 \\
 & && g_7(\vec{x}) = x_2^2 + x_5^2 \leq 1.64 \\
 & && g_8(\vec{x}) = x_3^2 + x_6^2 \leq 4.25 \\
 & && g_9(\vec{x}) = x_3^2 + x_5^2 \leq 4.64 \\
 & \text{with bounds} && 0 \leq x_1, x_2, x_3 \leq 1, \\
 & && x_4, x_5, x_6, x_7 \in \{0, 1\}
 \end{aligned} \tag{17}$$

In Table 7, the comparison results show that JSOA and COOT algorithms reported feasible solutions, whereas MAO, CGO, GEO, and HHO are infeasible, as seen in the

convergence graph in Figure 12. Note that the JSOA algorithm is ranked as the first best-obtained solution. The difference with the best-known feasible objective function value is 1.21×10^4 .

5.4. Optimal Design of an Industrial Refrigeration System

The problem is formulated as a non-linear inequality-constrained optimization problem [50]. Here, there are fourteen decision variables and 15th inequality constraints. The best-known feasible objective is $f(\vec{x}) = 3.22130008 \times 10^{-2}$. This problem can be stated as follows:

$$\begin{aligned}
 & \text{Minimize} && f(\vec{x}) = 63098.88x_2x_4x_{12} + 5441.5x_2^2x_{12} + 115055.5x_2^{1.664}x_6 + 6172.27x_2^2x_6 \\
 & && + 63098.88x_1x_3x_{11} + 5441.5x_1^2x_{11} + 115055.5x_1^{1.664}x_5 + 6172.27x_1^2x_5 + 140.53x_1x_{11} \\
 & && 281.29x_3x_{11} + 70.26x_1^2 + 281.29x_3^2 + 14437x_8^{1.8812}x_{12}^{0.3424}x_{10}x_{14}^{-1}x_1^2x_7x_9^{-1} \\
 & && + 20470.2x_7^{2.893}x_{11}^{0.316}x_1^2 \\
 & \text{Subject to} && g_1(\vec{x}) = 1.524x_7^{-1} \leq 1 \\
 & && g_2(\vec{x}) = 1.524x_8^{-1} \leq 1 \\
 & && g_3(\vec{x}) = 0.07789x_1 - 2x_7^{-1}x_9 \leq 1 \\
 & && g_4(\vec{x}) = 7.05305x_9^{-1}x_1^2x_{10}x_8^{-1}x_2^{-1}x_{14}^{-1} \leq 1 \\
 & && g_5(\vec{x}) = 0.0833x_{13}^{-1}x_{14} \leq 1 \\
 & && g_6(\vec{x}) = 47.136x_2^{0.333}x_{10}^{-1}x_{12} - 1.333x_8x_{13}^{2.1195} + 62.08x_{13}^{2.1195}x_{12}^{-1}x_8^{0.2}x_{10}^{-1} \leq 1 \\
 & && g_7(\vec{x}) = 0.04771x_{10}x_8^{1.8812}x_{12}^{0.3424} \leq 1 \\
 & && g_8(\vec{x}) = 0.0488x_9x_7^{1.893}x_{11}^{0.316} \leq 1 \\
 & && g_9(\vec{x}) = 0.0099x_1x_3^{-1} \leq 1 \\
 & && g_{10}(\vec{x}) = 0.0193x_2x_4^{-1} \leq 1 \\
 & && g_{11}(\vec{x}) = 0.0298x_1x_5^{-1} \leq 1 \\
 & && g_{12}(\vec{x}) = 0.056x_2x_6^{-1} \leq 1 \\
 & && g_{13}(\vec{x}) = 2x_9^{-1} \leq 1 \\
 & && g_{14}(\vec{x}) = 2x_{10}^{-1} \leq 1 \\
 & && g_{15}(\vec{x}) = x_{12}x_{11}^{-1} \leq 1 \\
 & \text{with bounds} && 0.001 \leq x_i \leq 5, i = 1, 2, \dots, 14
 \end{aligned} \tag{18}$$

Table 7. Comparison results for the Process Synthesis problem.

Algorithm	Optimal Values for Variables							f_{\min}
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	
JSOA	0.19468	1.2806	1.9559	1	0	0	1	2.92495120667677
MAO	7.41298	2.08955	13.6171	0	1	1	1	5.69266×10^{17} (infeasible *)
CGO	11.6400	32.7787	47.0926	1	1	0	1	3.9149×10^{22} (infeasible *)
COOT	0.14214	0.78882	1.9077	1	1	0	1	3.00121860556942
GEO	96.5572	47.3277	86.2538	1	0	1	0	3.6795×10^{43} (infeasible *)
HHO	0.38922	1.1288	2.018	0	0	0	0	4.7224 (infeasible *)

* The solution does not satisfy one or more constraints.

In Table 8, the comparison results show that JSOA and COOT algorithms reported feasible solutions, whereas MAO, CGO, GEO, and HHO are infeasible, as seen in the convergence graph illustrated in Figure 13. Note that the JSOA algorithm is ranked as the first best-obtained solution. The difference with the best-known feasible objective function value is 2.77×10^{-3} .

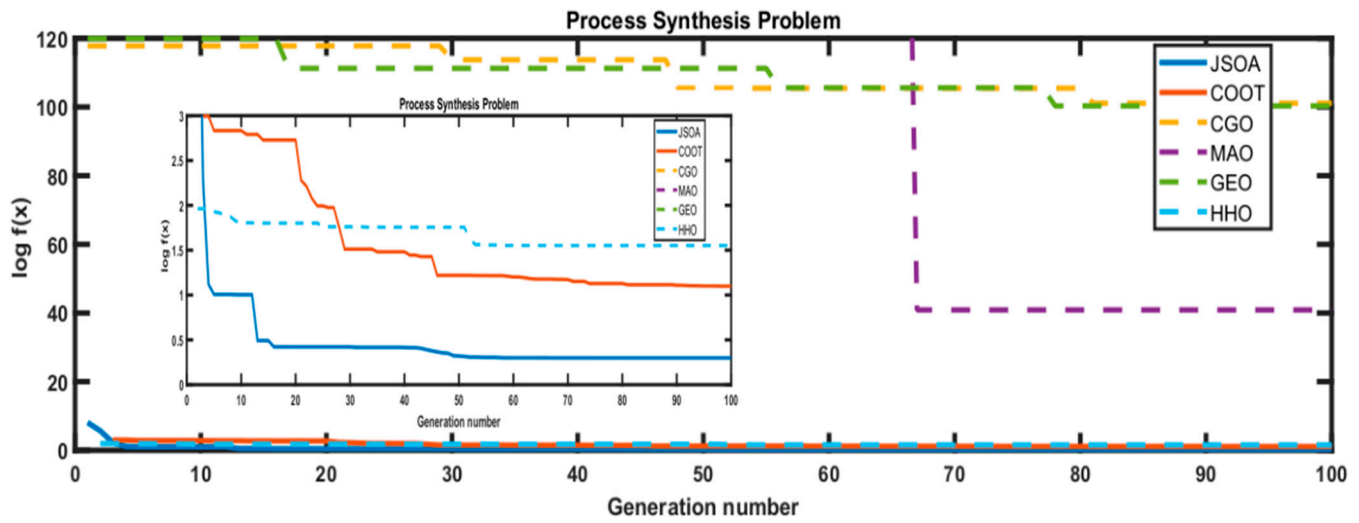


Figure 12. Convergence graph of the Process Synthesis Problem. The dotted line represents an infeasible solution shown by an algorithm.

Table 8. Comparison results for the Optimal Design of an Industrial Refrigeration System.

Algorithm	Optimal Values for Variables														f_{min}
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	
JSOA	0.001	0.0010003	0.001	0.001	0.001	0.001	1.58	1.5241	4.9599	2.1421	0.001	0.001	0.007508	0.089784	0.034988
MAO	3.446	4.0369	4.0025	1.5223	0.29135	0.40969	1.4476	4.61	3.4963	4.7813	1.2534	4.5339	4.007	3.1936	8.07×10^6 (infeasible *)
* CGO	5	2.8898	5	2.3383	3.3192	5	3.2098	5	3.2024	5	5	3.8764	5	3.4982	6.19×10^7 (infeasible *)
COOT	0.001	0.001	0.001	0.001	4.7635	4.839	2.0353	1.6635	3.3789	4.8519	2.0843	0.58734	0.11329	1.1238	12.659065
GEO	3.0574	1.9180	2.2207	3.3653	1.9315	1.8528	2.1755	3.7480	1.8249	2.3242	0.0010	2.0866	3.1755	2.1960	5.94×10^6 (infeasible *)
HHO	0.001	0.001	0.001	0.21205	1.8724	2.6267	2.795	2.7998	3.0903	4.9901	4.9908	4.9937	1.9368	2.7519	7.55×10 (infeasible *)

* The solution does not satisfy one or more constraints.

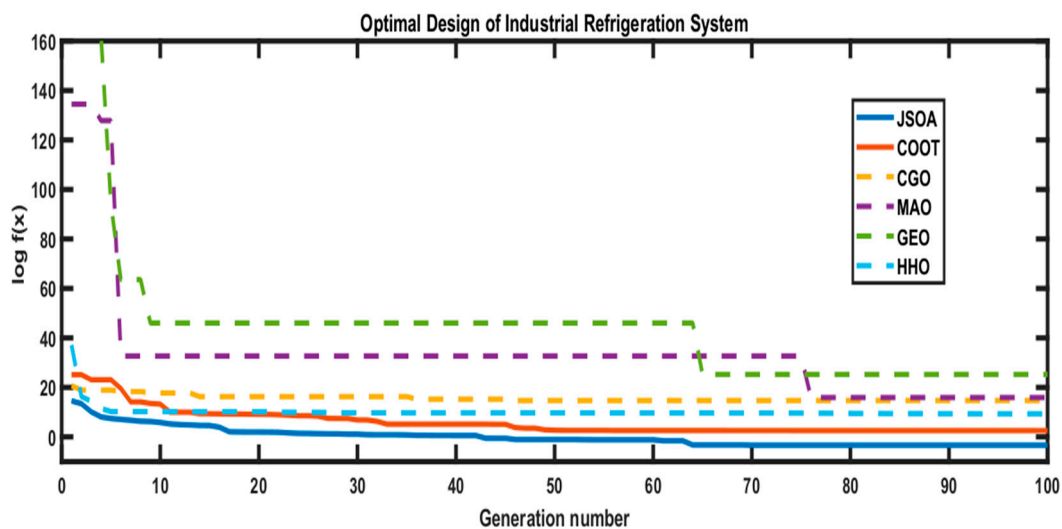


Figure 13. Convergence graph of the Optimal Design of an Industrial Refrigeration System. The dotted line represents an infeasible solution shown by an algorithm.

5.5. Welded Beam Design

The main objective of this problem is to design a welded beam with minimum cost [50]. This problem contains five inequality constraints and four decision variables that are used to develop a welded beam [51]. The best known feasible objective function value is

$f(\vec{x}) = 1.6702177263$. Therefore, the mathematical description of this problem can be defined as follows:

$$\begin{aligned}
 & \text{Minimize } f(\vec{x}) = 1.10471 x_1^2 x^2 + 0.04811 x_3 x_4 (x_2 + 14) \\
 & \text{Subject to } g_1(\vec{x}) = x_1 - x_4 \leq 0 \\
 & \quad g_2(\vec{x}) = \delta(\vec{x}) - \delta_{\max} \leq 0 \\
 & \quad g_3(\vec{x}) = P \leq P_c(\vec{x}) \\
 & \quad g_4(\vec{x}) = \tau_{\max} \geq \tau(\vec{x}) \\
 & \quad g_5(\vec{x}) = \sigma(\vec{x}) - \sigma_{\max} \leq 0
 \end{aligned}$$

where,

$$\tau = \sqrt{\tau^l{}^2 + \tau^u{}^2 + 2\tau^l\tau^u \frac{x^2}{2R}}, \tau^u = \frac{RM}{J}, \tau^l = \frac{P}{\sqrt{2x_2x_1}}, M = P(\frac{x_2}{2} + L), \tag{19}$$

$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1+x_3}{2}\right)^2}, J = 2\left(\left(\frac{x_2^2}{4} + \left(\frac{x_1+x_3}{2}\right)^2\right)\sqrt{2x_1x_2}\right), \sigma(\vec{x}) = \frac{6PL}{x_4x_3^2},$$

$$L = 14in, P = 6000 lb, E = 30.10^6 psi, \sigma_{\max} = 30,000 psi, \tau_{\max} = 13,600 psi, \delta_{\max} = 0.25in,$$

with bounds :

$$\begin{aligned}
 & 0.125 \leq x_1 \leq 2, \\
 & 0.1 \leq x_2, x_3 \leq 10, \\
 & 0.1 \leq x_4 \leq 2
 \end{aligned}$$

In Table 9, the comparison results show that JSOA, MAO, COOT and GEO algorithms reported feasible solutions, whereas GEO and HHO are infeasible, as seen in the convergence graph in Figure 14. Furthermore, the COOT algorithm showed a competitive result, whereas the JSOA algorithm is ranked as the first best-obtained solution. The difference with the best-known feasible objective function value is 2.27×10^{-8} .

Table 9. Comparison results for Welded Beam Design.

Algorithm	Optimal Values for Variables				f_{\min}
	x_1	x_2	x_3	x_4	
JSOA	0.198832312221973	3.337365260666560	9.192024209255530	0.198832312331652	1.67021774898897
MAO	1.0365	2.0756	6.1919	1.1649	8.04184550085786
CGO	0.5131	6.8515	2.5371	0.22107	2.5553 (infeasible *)
COOT	0.19808	3.384	9.1731	0.19977	1.67928562217482
GEO	0.93008	3.6876	5.6868	0.93798	8.06300000000000
HHO	0.34167	2.307	7.0138	0.34153	2.17679623002226 (infeasible *)

* The solution does not satisfy one or more constraints.

5.6. Tuning of a Proportional-Integral-Derivative (PID) Controller: Sloshing Dynamics Problem

This problem is taken from [14]. The goal is to tune a Proportional-Integral-Derivative (PID) controller for the Sloshing dynamics phenomenon (SDP). The SDP is a well-known problem in fluid dynamics. It is related to the movement of a liquid inside another object, altering the system dynamics [14]. Sloshing is an important effect on mobile vehicles carrying liquids, e.g., ships, spacecraft, aircraft, and trucks. A deficient sloshing control causes instability and accidents.

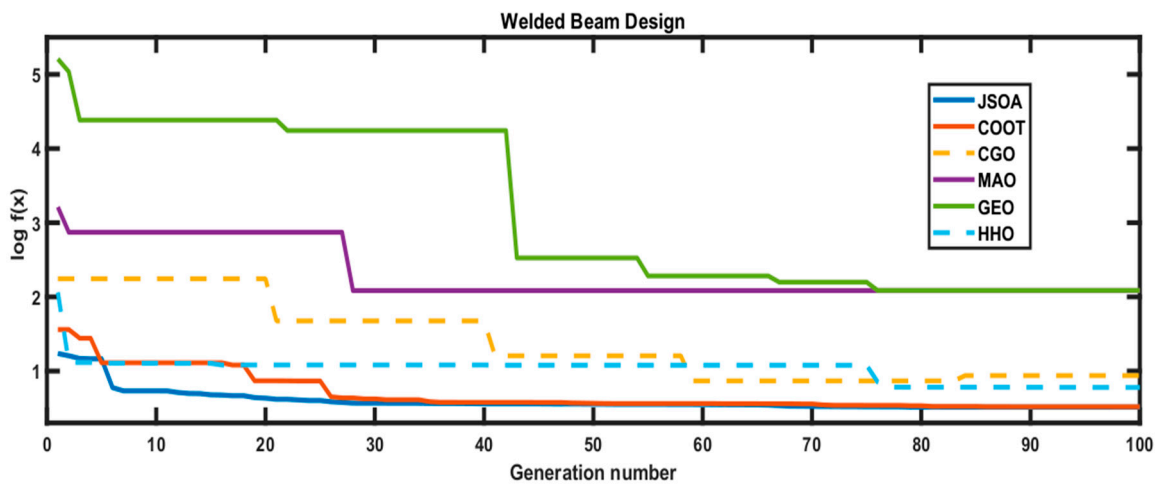


Figure 14. Convergence graph of Welded Beam Design. The dotted line represents an infeasible solution shown by an algorithm.

Sloshing dynamics can be depicted as a Ball and Hoop System (BHS). This effect illustrates the dynamics of a steel ball that is free to roll on the inner surface of a rotating circular hoop [52]. The ball exhibits an oscillatory motion caused by the continuously rotated hoop through a motor. The ball will tend to move in the direction of the hoop rotation and will fall back, at some point, when gravity overcomes the frictional forces [14]. Seven variables can describe the BHS behavior: hoop radius (R), hoop angle (θ), input torque to the hoop ($T(t)$), ball position on the hoop (y), ball radius (r), ball mass (m) and ball angles with vertical (slosh angle) (ψ) [14]. A schematic representation is shown in Figure 15. The transfer function of the BHS system, taken from [14], is formulated in Equation (20). Where θ is, the input and y are the output of the BHS system.

$$G_{BHS}(s) = \frac{y(s)}{\theta(s)} = \frac{1}{s^4 + 6s^3 + 11s^2 + 6s} \tag{20}$$

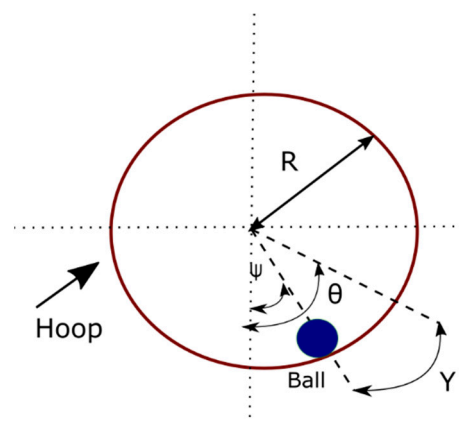


Figure 15. Schema of the ball and hoop system [14].

Table 10 summarizes the comparison results at solving the BHS for the optimal tuning of a Proportional-Integral Derivative (PID) controller. The transient response parameters of the PID controller are Rise time, Settling time, Peak time, and Peak overshoot. The PID controller is designed to minimize the overshoot and settling time so that the liquid can remain as stable as possible under any perturbation, and if it moves, it can rapidly go back to its steady-state [14]. It is to be noticed that the JSOA and HHO show competitive results and are better than the rest of the algorithms part of this test, obtaining the lowest rising

and settling time value, as well as the peak overshoot, see Table 11. In addition, the PID controller step response for the six algorithms is shown in Figure 16. This figure shows that the JSOA (blue line) and HHO (magenta line) are more stable with very fine control without exceeding the setpoint (dotted line).

Table 10. Comparison results of optimized PID parameters.

Algorithm	Parameter		
	Kp	Ki	Kd
JSOA	3.6729	0.0058	4.9697
MAO	3.9916	0.59242	4.4857
CGO	2.5303	0.35241	5
COOT	3.8496	0.57292	5
GEO	3.6799	0.30945	2.2922
HHO	3.6504	0.01	4.9338

Table 11. Comparison results of transient response parameters.

Algorithm	Transient Parameters			
	Rise Time (s)	Settling Time (s)	Peak Time (s)	Peak Overshoot (%)
JSOA	2.0059	3.2915	4.027	0
MOA	1.7484	15.3316	4.9209	22.1115
CGO	2.4723	23.9662	7.2793	52.2769
COOT	1.7455	16.1494	5.659	17.9202
GEO	2.0863	22.9262	5.3124	27.432
HHO	2.0205	3.3181	4.0793	0

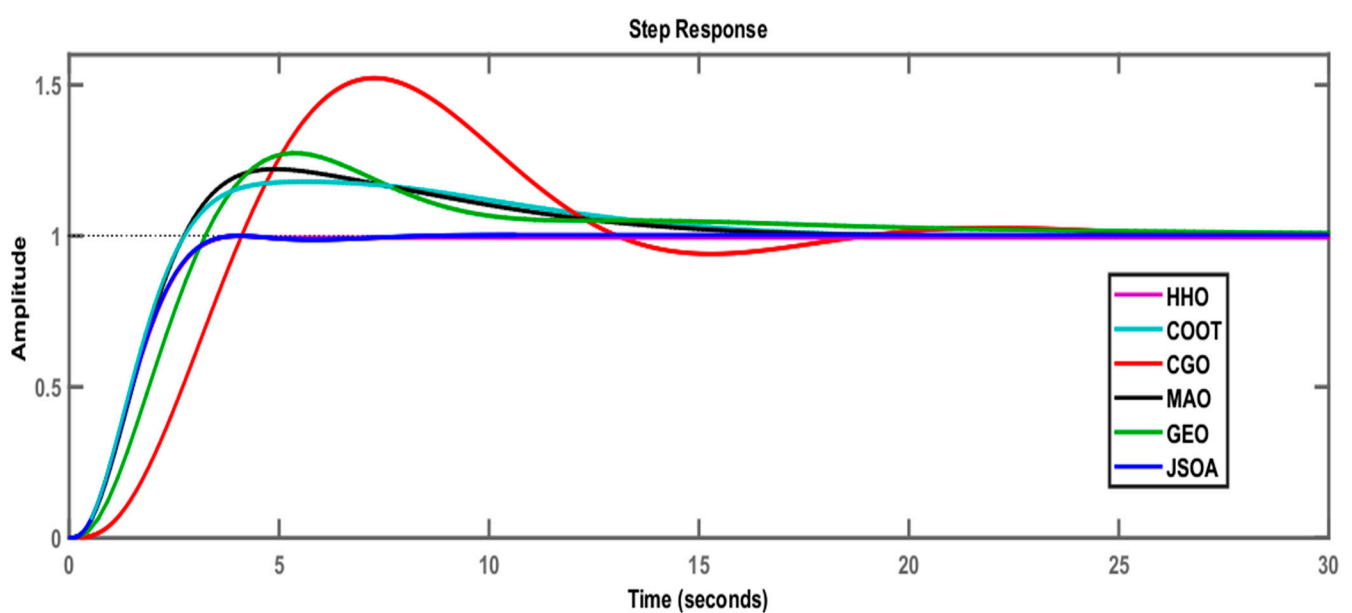


Figure 16. Comparative results of step response for the PID controller.

5.7. Selective Harmonics Elimination (SHE) Problem

The selective harmonic elimination (SHE) problem is taken from [15]; it is a highly used control strategy applied to multilevel inverters (MLI) that aims for the elimination of

unwanted low order harmonics by setting them equal to zero. In contrast, the fundamental component is kept equal to the desired amplitude. Figure 17 shows the typical staircase output waveform of a single-phase multilevel inverter. This waveform is generated by the correct synchronization and angle switching of the MLI power semiconductor devices, part of cascaded H-bridge inverter modules with isolated direct current (dc) sources, connected either in cascaded or series.

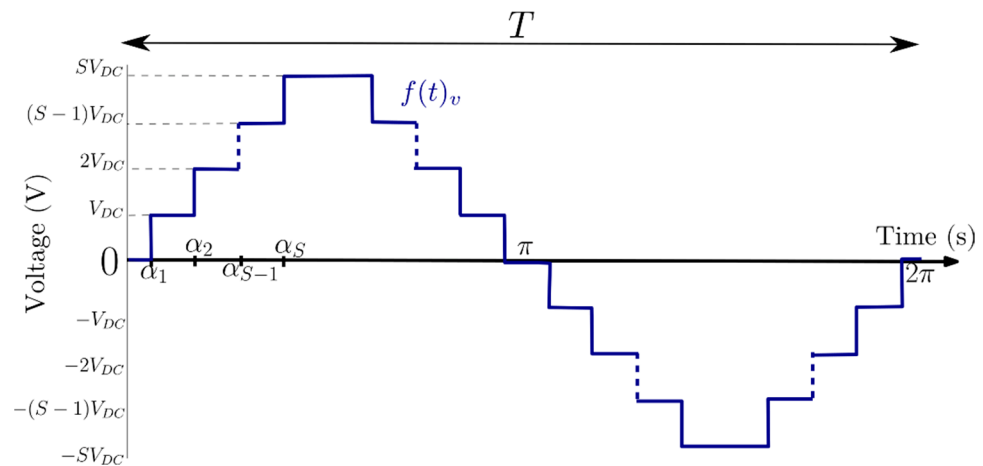


Figure 17. Single phase multilevel inverter staircase output waveform [15].

The Fourier series expansion of this waveform is defined in Equation (21). Due to the quarter-wave symmetry nature of the waveform, the dc component, and the Fourier coefficient A_n will be both equal to 0.

$$f(t) = \underbrace{A_0}_{dc} + \underbrace{\sum_{n=1}^{\infty} (A_n \cos(n\alpha) + B_n \sin(n\alpha))}_{ac} \Bigg|_{\omega_0 = \frac{2\pi}{T}} \quad (21)$$

By Fourier transforming the staircase output waveform, it can be described in Equation (22).

$$f(t)_V^+ = \begin{cases} \frac{4V_{dc}}{n\pi} \cos(n\alpha_1) + \dots + \cos(n\alpha_n) & \text{for odd } n \\ 0 & \text{for even } n \end{cases} \quad (22)$$

Therefore, a series of nonlinear equations must be solved for unknown angles. The mathematical relationship between the MLI isolated dc sources (S) and the number of levels (n) is shown in Equation (23).

$$n = 2S + 1 \quad (23)$$

where the switching angles (α) are subjected to:

$$0 \leq \alpha_1 \leq \alpha_2 \leq \dots \leq \alpha_{(s-1)} \leq \alpha_s \leq 90^\circ \quad (24)$$

As a case of study, a 3ϕ eleven-levels multilevel inverter is selected with an index of modulation equal to 1. Thus, the selective harmonic elimination set of equations that eliminates the fifth, seventh, eleventh and thirteenth harmonic can be rewritten as below:

$$\begin{aligned} \cos(\alpha_1) + \cos(\alpha_2) + \dots + \cos(\alpha_5) &= M \\ \cos(5\alpha_1) + \cos(5\alpha_2) + \dots + \cos(5\alpha_5) &= 0 \\ \cos(7\alpha_1) + \cos(7\alpha_2) + \dots + \cos(7\alpha_5) &= 0 \\ \cos(11\alpha_1) + \cos(11\alpha_2) + \dots + \cos(11\alpha_5) &= 0 \\ \cos(13\alpha_1) + \cos(13\alpha_2) + \dots + \cos(13\alpha_5) &= 0 \end{aligned} \quad (25)$$

where $M = (V_1^*) / (4V_{dc}\pi)$ and the modulation index is defined as $m = (M/5)$ for $0 \leq m \leq 1$. V_1^* is defined as the desired peak voltage and V_{dc} is equal to the direct voltage of the isolated dc power supplies.

Therefore, the objective function [15], is defined as:

$$\min f(\alpha_1, \alpha_2, \dots, \alpha_5) = \left[\sum_{i=1}^5 \cos(\alpha_i) - M \right]^2 + \left[\sum_{i=1}^5 \cos(5\alpha_i) \right]^2 + \dots + \left[\sum_{i=1}^5 \cos(13\alpha_i) \right]^2 \tag{26}$$

Subjected to the switching angles described in Equation (24). Some of the algorithms that are chosen for comparison are Whale Optimization Algorithm (WOA), Modified Grey Wolf Optimization Algorithm (MGWOA) and Black Widow Optimization Algorithm (BWOA). Table 5 was taken from [15] and updated with results of the following algorithms: Coot Bird Algorithm (COOT) [16], Chaos Game Optimization (CGO) [42], Mexican Axolotl Optimization (MAO) [17], Golden Eagle Optimizer (GEO) [18], Harris Hawks Optimization (HHO)[26], and Jumping Spider Optimization Algorithm (JSOA). The obtained near-optimal angles are then fed to a Simulink code that retrieves the staircase output waveform. Then, the Total Harmonic Distortion (THD) is calculated, and a Fourier spectrum is determined to show the correct elimination of the unwanted low order harmonics. From Table 12, it can be seen that JSOA gets the best fitness values, whereas Figure 18 shows the correct elimination of the fifth, seventh, eleventh, and thirteenth order harmonics.

Table 12. Comparison results for the selective harmonic elimination problem.

Algorithm	Angles					THD%	f_{\min}
	α_1	α_2	α_3	α_4	α_5		
JSOA	7.86	19.36	29.65	47.68	63.20	5.01	3.49×10^{-31}
BWOA	7.86	19.37	29.65	47.68	63.21	5.01	1.29×10^{-28}
WOA	4.19	20.29	22.12	41.97	61.15	6.9	3.93×10^{-2}
MGWOA	0.49	14.74	25.61	40.57	89.16	5.71	16.04×10^{-2}
MAO	9.67	22.37	33.04	51.91	64.63	5.26	3.64×10^{-1}
CGO	13.25	26.37	45.93	46.05	87.22	13.99	9.76×10^{-2}
COOT	7.97	19.46	29.79	47.74	63.20	5.04	3.43×10^{-4}
GEO	8.62	10.60	38.76	76.59	83.17	14.83	4.64×10^{-1}
HHO	8.16	19.50	30.15	48.21	63.35	5.16	2.69×10^{-4}

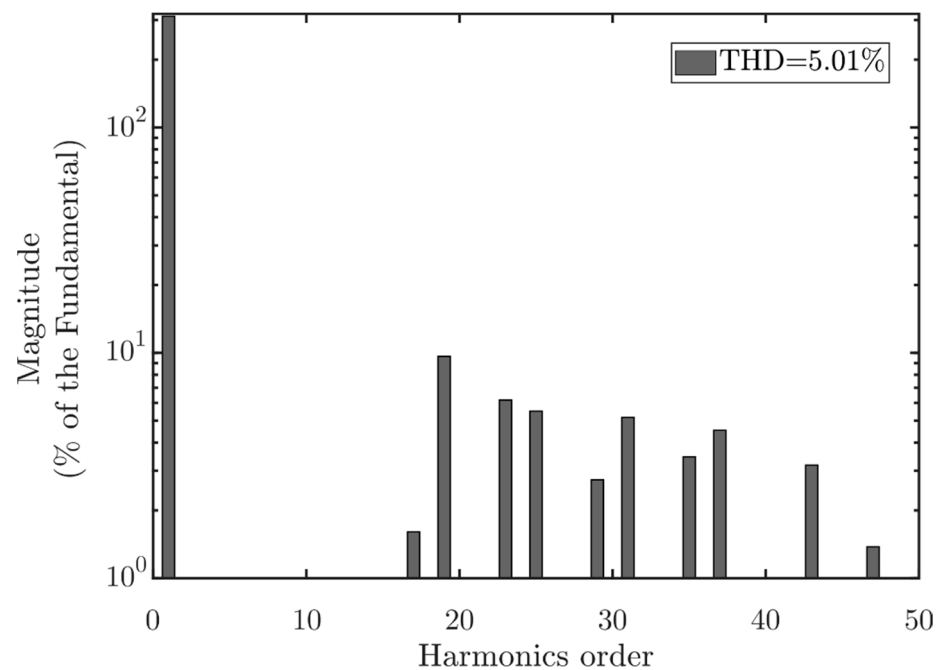


Figure 18. Fourier transform spectrum calculated from the JSOA set of angles described in Table 12.

6. Conclusions

This work presented a new swarm-based optimization algorithm inspired by the hunting behavior Arachnida Salticidae, named Jumping Spider Optimization Algorithm (JSOA). The proposed method included three operators to simulate the spider hunting methods (search, persecution, and jumping on the prey). These strategies work as mathematical recombination functions of vectors (spiders, also named search agents), that provide a fine balance between exploitation and exploration over the solution search space. The algorithm's performance was benchmarked on 20 test functions, four real-world optimization problems, tuning of a Proportional-Integral-Derivative (PID) controller, and the selective harmonic elimination problem. In addition, the performance of the proposed JSOA algorithm is compared with the ten most recent algorithms in the literature: COOT, CGO, MAO, GEO, AOA, ArOA, GBO, HGS, HGSO, and HHO. The statistical results show that the JSOA algorithm outperforms or has competitive results in these algorithms. This study has the following conclusions:

- The JSOA algorithm does not have parameters to configure affecting its performance.
- Exploitation and exploration of the JSOA are intrinsically high on problems involving unimodal and multimodal test functions, respectively.
- The algorithm has outstanding results with few iterations, 200 for the testbench functions and 100 for the real-world problems.
- The JSOA algorithm models the pheromone of spiders whose criteria are used to repair vectors with a low pheromone level. The vectors (spiders, also named search agents) with the worst fitness are replaced in each iteration. This repair helps get a better performance in the exploration of the search space.
- The Wilcoxon rank-sum test p -values confirm the meaningful advantage of JSOA compared to other bio-inspired algorithms for many cases. In addition, the MAE statistical results show that the JSOA is ranked among the highest position compared to the other algorithms.
- JSOA can tune a Proportional-Integral-Derivative (PID) controller with very fine control without exceeding the setpoint, with zero percent of peak overshoot for the Sloshing dynamics Problem.
- JSOA can solve the Selective Harmonic Elimination problem with the best fitness value results compared to WOA, MGWOA, BWOA, COOT, CGO, MAO, GEO, and HHO

algorithms and competitive results with the BWOA algorithm regarding the Total Harmonic Distortion (THD)

- JSOA can solve real-world problems with unknown search spaces.

The evaluation criterion of the pheromone rate lacks a sensitivity analysis of the fixed 0.3 value previously determined empirically. An updated version of the JSOA algorithm addressing this issue and that solves multi and many-objective functions is currently under development for future work

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

Funding: This research was funded by Instituto Politécnico Nacional (IPN) through grant number SIP-20211364.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The source code used to support the findings of this study has been deposited in the MathWorks repository at <https://www.mathworks.com/matlabcentral/fileexchange/104045-a-bio-inspired-method-inspired-by-arachnida-salticidade> (accessed on 16 October 2021).

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

Appendix A

Table A1. Classification of Testbench functions.

ID	Function Name	Unimodal	Multimodal	n-Dimensional	Non-Separable	Convex	Differentiable	Continuous	Non-Convex	Non-Differentiable	Separable	Random
F1	Brown	X		X	X	X	X					
F2	Griewank	X		X	X			X	X			
F3	Schwefel 2.20	X		X	X		X	X	X			
F4	Schwefel 2.21	X		X		X		X		X	X	
F5	Schwefel 2.22	X		X		X		X		X	X	
F6	Schwefel 2.23	X		X		X	X	X			X	
F7	Sphere	X		X		X	X	X			X	
F8	Sum Squares	X		X		X	X	X			X	
F9	Xin-She Yang N. 3	X		X	X		X		X			
F10	Zakharov	X		X		X		X				
F11	Ackley		X	X			X	X	X			
F12	Ackley N. 4		X	X	X		X		X			
F13	Periodic		X	X			X	X	X	X		
F14	Quartic		X	X			X	X			X	X
F15	Rastrigin		X	X		X	X	X			X	
F16	Rosenbrock		X	X	X		X	X	X			
F17	Salomon		X	X	X		X	X	X			
F18	Xin-She Yang		X	X					X	X	X	
F19	Xin-She Yang N. 2		X	X	X				X	X		
F20	Xin-She Yang N. 4		X	X	X				X	X		

Table A2. Description of the Testbench Functions.

ID	Function	Dim	Interval	f_{\min}
F1	$f(\mathbf{x}) = f(x_1, \dots, x_n) = \sum_{i=1}^{n-1} (x_i^2)^{(x_{i+1}^2+1)} + (x_{i+1}^2)^{(x_i^2+1)}$	30	$[-1, 4]$	0
F2	$f(\mathbf{x}) = f(x_1, \dots, x_n) = 1 + \sum_{i=1}^n \frac{x_i^2}{4000} - \prod_{i=1}^n \cos(\frac{x_i}{\sqrt{i}})$	30	$[-600, 600]$	0
F3	$f(\mathbf{x}) = f(x_1, \dots, x_n) = \sum_{i=1}^n x_i $	30	$[-100, 100]$	0
F4	$f(\mathbf{x}) = f(x_1, \dots, x_n) = \max_{i=1, \dots, n} x_i $	30	$[-100, 100]$	0
F5	$f(\mathbf{x}) = f(x_1, \dots, x_n) = \sum_{i=1}^n x_i + \prod_{i=1}^n x_i $	30	$[-100, 100]$	0
F6	$f(\mathbf{x}) = f(x_1, \dots, x_n) = \sum_{i=1}^n x_i^{10}$	30	$[-10, 10]$	0
F7	$f(\mathbf{x}) = f(x_1, x_2, \dots, x_n) = \sum_{i=1}^n x_i^2$	30	$[-5.12, 5.12]$	0
F8	$f(\mathbf{x}) = f(x_1, \dots, x_n) = \sum_{i=1}^n ix_i^2$	30	$[-10, 10]$	0
F9	$f(\mathbf{x}) = f(x_1, \dots, x_n) = \exp(-\sum_{i=1}^n (x_i/\beta)^{2m}) - 2\exp(-\sum_{i=1}^n x_i^2) \prod_{i=1}^n \cos^2(x_i)$	30	$[-2\pi, 2\pi], m = 5, \beta = 15$	-1
F10	$f(\mathbf{x}) = f(x_1, \dots, x_n) = \sum_{i=1}^n x_i^2 + (\sum_{i=1}^n 0.5ix_i)^2 + (\sum_{i=1}^n 0.5ix_i)^4$	30	$[-5, 10]$	0
F11	$f(\mathbf{x}) = f(x_1, \dots, x_n) = -a.\exp(-b\sqrt{\frac{1}{n}\sum_{i=1}^n x_i^2}) - \exp(\frac{1}{d}\sum_{i=1}^n \cos(cx_i)) + a + \exp(1)$	30	$[-32, 32], a = 20, b = 0.3, c = 2\pi$	0
F12	$f(\mathbf{x}) = f(x_1, \dots, x_n) = \sum_{i=1}^{n-1} (e^{-0.2\sqrt{x_i^2 + x_{i+1}^2}} + 3(\cos(2x_i) + \sin(2x_{i+1})))$	2	$[-35, 35]$	-5.901×10^{14}
F13	$f(\mathbf{x}) = f(x_1, \dots, x_n) = 1 + \sum_{i=1}^n \sin^2(x_i) - 0.1e^{(\sum_{i=1}^n x_i^2)}$	30	$[-10, 10]$	0.9
F14	$f(\mathbf{x}) = f(x_1, \dots, x_n) = \sum_{i=1}^n ix_i^4 + \text{random}[0, 1)$	30	$[-1.28, 1.28]$	0 + random noise
F15	$f(x, y) = 10n + \sum_{i=1}^n (x_i^2 - 10 \cos(2\pi x_i))$	30	$[-5.12, 5.12]$	0
F16	$f(x_1 \cdots x_n) = \sum_{i=1}^{n-1} (100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2)$	30	$[-5, 10]$	0
F17	$f(\mathbf{x}) = f(x_1, \dots, x_n) = 1 - \cos(2\pi\sqrt{\sum_{i=1}^D x_i^2}) + 0.1\sqrt{\sum_{i=1}^D x_i^2}$	30	$[-100, 100]$	0
F18	$f(\mathbf{x}) = f(x_1, \dots, x_n) = \sum_{i=1}^n \epsilon_i x_i ^i$	30	$[-5, 5], \epsilon$ random	0
F19	$f(\mathbf{x}) = f(x_1, \dots, x_n) = (\sum_{i=1}^n x_i) \exp(-\sum_{i=1}^n \sin(x_i^2))$	30	$[-2\pi, 2\pi]$	0
F20	$f(\mathbf{x}) = f(x_1, \dots, x_n) = \left(\sum_{i=1}^n \sin^2(x_i) - e^{-\sum_{i=1}^n x_i^2} \right) e^{-\sum_{i=1}^n \sin^2 \sqrt{ x_i }}$	30	$[-10, 10]$	-1

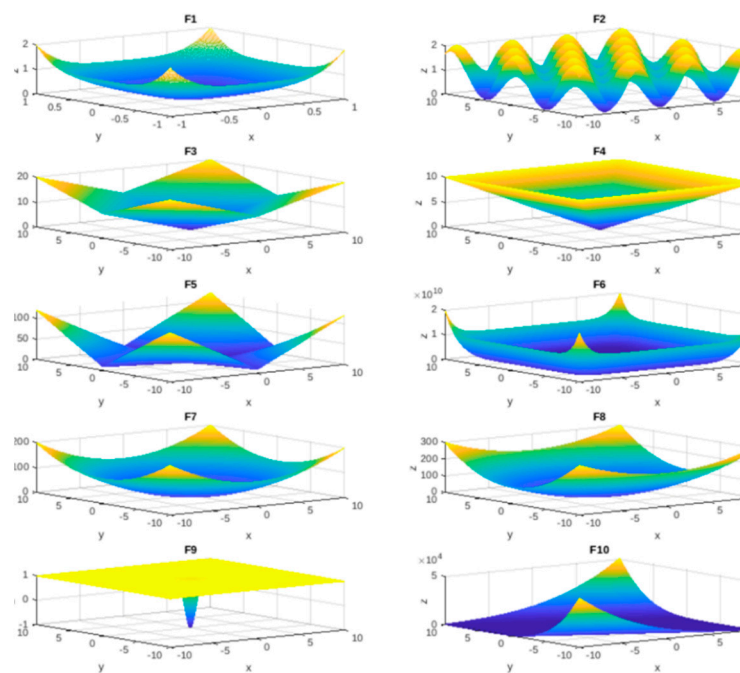


Figure A1. Testbench Unimodal functions.

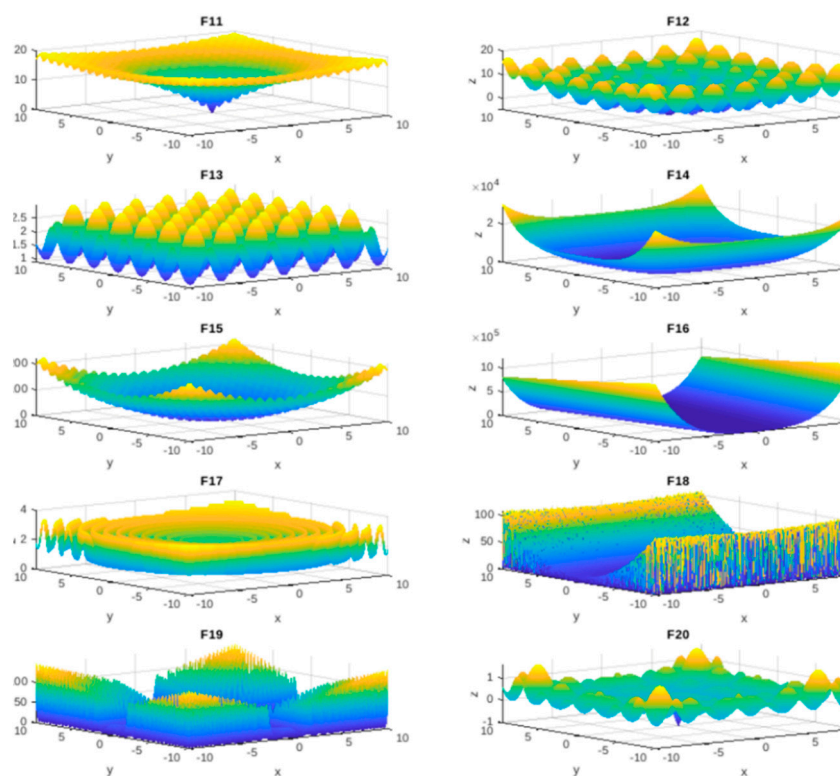


Figure A2. Testbench Multimodal functions.

References

1. Kumar, A. Application of nature-inspired computing paradigms in optimal design of structural engineering problems—A review. *Nat.-Inspired Comput. Paradig. Syst.* **2021**, *63–74*. [[CrossRef](#)]
2. Shaukat, N.; Ahmad, A.; Mohsin, B.; Khan, R.; Khan, S.U.-D.; Khan, S.U.-D. Multiobjective Core Reloading Pattern Optimization of PARR-1 Using Modified Genetic Algorithm Coupled with Monte Carlo Methods. *Sci. Technol. Nucl. Install.* **2021**, *2021*, 1–13. [[CrossRef](#)]

3. Lodewijks, G.; Cao, Y.; Zhao, N.; Zhang, H. Reducing CO₂ Emissions of an Airport Baggage Handling Transport System Using a Particle Swarm Optimization Algorithm. *IEEE Access* **2021**, *9*, 121894–121905. [[CrossRef](#)]
4. Malik, H.; Iqbal, A.; Joshi, P.; Agrawal, S.; Bakhsh, F.I. (Eds.) *Metaheuristic and Evolutionary Computation: Algorithms and Applications*; Springer: Berlin/Heidelberg, Germany, 2021; Volume 916. [[CrossRef](#)]
5. Elaziz, M.A.; Dahou, A.; Abualigah, L.; Yu, L.; Alshinwan, M.; Khasawneh, A.M.; Lu, S. Advanced metaheuristic optimization techniques in applications of deep neural networks: A review. *Neural Comput. Appl.* **2021**, *33*, 14079–14099. [[CrossRef](#)]
6. Wolpert, D.H.; Macready, W.G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [[CrossRef](#)]
7. Ho, Y.; Pepyne, D. Simple Explanation of the No-Free-Lunch Theorem and Its Implications. *J. Optim. Theory Appl.* **2002**, *115*, 549–570. [[CrossRef](#)]
8. Mirjalili, S.; Song Dong, J.; Lewis, A.; Sadiq, A.S. Particle Swarm Optimization: Theory, Literature Review, and Application in Airfoil Design. In *Nature-Inspired Optimizers: Theories, Literature Reviews and Applications*; Mirjalili, S., Song Dong, J., Lewis, A., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 167–184.
9. Castelli, M.; Manzoni, L.; Mariot, L.; Nobile, M.S.; Tangherloni, A. Salp Swarm Optimization: A critical review. *Expert Syst. Appl.* **2021**, *189*, 116029. [[CrossRef](#)]
10. Mirjalili, S.; Lewis, A. The Whale Optimization Algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [[CrossRef](#)]
11. Braik, M.S. Chameleon Swarm Algorithm: A bio-inspired optimizer for solving engineering design problems. *Expert Syst. Appl.* **2021**, *174*, 114685. [[CrossRef](#)]
12. Jiang, Y.; Wu, Q.; Zhu, S.; Zhang, L. Orca predation algorithm: A novel bio-inspired algorithm for global optimization problems. *Expert Syst. Appl.* **2021**, *188*, 116026. [[CrossRef](#)]
13. Abdollahzadeh, B.; Gharehchopogh, F.S.; Mirjalili, S. African vultures optimization algorithm: A new nature-inspired metaheuristic algorithm for global optimization problems. *Comput. Ind. Eng.* **2021**, *158*, 107408. [[CrossRef](#)]
14. Peraza-Vázquez, H.; Peña-Delgado, A.F.; Echavarría-Castillo, G.; Morales-Cepeda, A.B.; Velasco-Álvarez, J.; Ruiz-Perez, F. A Bio-Inspired Method for Engineering Design Optimization Inspired by Dingoes Hunting Strategies. *Math. Probl. Eng.* **2021**, *2021*, 1–19. [[CrossRef](#)]
15. Peña-Delgado, A.F.; Peraza-Vázquez, H.; Almazán-Covarrubias, J.H.; Cruz, N.T.; García-Vite, P.M.; Morales-Cepeda, A.B.; Ramirez-Arredondo, J.M. A novel bio-inspired algorithm applied to selective harmonic elimination in a three-phase eleven-level inverter. *Math. Probl. Eng.* **2020**, *2020*, 1–10. [[CrossRef](#)]
16. Naruei, I.; Keynia, F. A new optimization method based on COOT bird natural life model. *Expert Syst. Appl.* **2021**, *183*, 115352. [[CrossRef](#)]
17. Villuendas-Rey, Y.; Velázquez-Rodríguez, J.; Alanis-Tamez, M.; Moreno-Ibarra, M.-A.; Yáñez-Márquez, C. Mexican Axolotl Optimization: A Novel Bioinspired Heuristic. *Mathematics* **2021**, *9*, 781. [[CrossRef](#)]
18. Mohammadi-Balani, A.; Nayeri, M.D.; Azar, A.; Taghizadeh-Yazdi, M. Golden eagle optimizer: A nature-inspired metaheuristic algorithm. *Comput. Ind. Eng.* **2021**, *152*, 107050. [[CrossRef](#)]
19. Abualigah, L.; Shehab, M.; Alshinwan, M.; Mirjalili, S.; Elaziz, M.A. Ant Lion Optimizer: A Comprehensive Survey of Its Variants and Applications. *Arch. Comput. Methods Eng.* **2021**, *28*, 1397–1416. [[CrossRef](#)]
20. Salehan, A.; Deldari, A. Corona virus optimization (CVO): A novel optimization algorithm inspired from the Corona virus pandemic. *J. Supercomput.* **2021**. [[CrossRef](#)]
21. Hashim, F.A.; Hussain, K.; Houssein, E.H.; Mabrouk, M.S.; Al-Atabany, W. Archimedes optimization algorithm: A new metaheuristic algorithm for solving optimization problems. *Appl. Intell.* **2021**, *51*, 1531–1551. [[CrossRef](#)]
22. Abualigah, L.; Diabat, A.; Mirjalili, S.; Elaziz, M.A.; Gandomi, A.H. The Arithmetic Optimization Algorithm. *Comput. Methods Appl. Mech. Eng.* **2021**, *376*, 113609. [[CrossRef](#)]
23. Ahmadianfar, I.; Bozorg-Haddad, O.; Chu, X. Gradient-based optimizer: A new metaheuristic optimization algorithm. *Inf. Sci.* **2020**, *540*, 131–159. [[CrossRef](#)]
24. Yang, Y.; Chen, H.; Heidari, A.A.; Gandomi, A.H. Hunger games search: Visions, conception, implementation, deep analysis, perspectives, and towards performance shifts. *Expert Syst. Appl.* **2021**, *177*, 114864. [[CrossRef](#)]
25. Hashim, F.A.; Houssein, E.H.; Mabrouk, M.S.; Al-Atabany, W.; Mirjalili, S. Henry gas solubility optimization: A novel physics-based algorithm. *Future Gener. Comput. Syst.* **2019**, *101*, 646–667. [[CrossRef](#)]
26. Heidari, A.A.; Mirjalili, S.; Faris, H.; Aljarah, I.; Mafarja, M.; Chen, H. Harris hawks optimization: Algorithm and applications. *Future Gener. Comput. Syst.* **2019**, *97*, 849–872. [[CrossRef](#)]
27. Singh, S.; Tiwari, A.; Agrawal, S. Differential Evolution Algorithm for Multimodal Optimization: A Short Survey. In *Advances in Intelligent Systems and Computing*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 745–756.
28. Katoch, S.; Chauhan, S.S.; Kumar, V. A review on genetic algorithm: Past, present, and future. *Multimed. Tools Appl.* **2021**, *80*, 8091–8126. [[CrossRef](#)] [[PubMed](#)]
29. Liu, W.-L.; Yang, J.; Zhong, J.; Wang, S. Genetic programming with separability detection for symbolic regression. *Complex Intell. Syst.* **2021**, *7*, 1185–1194. [[CrossRef](#)]
30. Sang, X.; Liu, X.; Zhang, Z.; Wang, L. Improved Biogeography-Based Optimization Algorithm by Hierarchical Tissue-Like P System with Triggering Ablation Rules. *Math. Probl. Eng.* **2021**, *2021*, 1–24. [[CrossRef](#)]

31. Fu, Y.; Zhou, M.; Guo, X.; Qi, L.; Sedraoui, K. Multiverse Optimization Algorithm for Stochastic Biobjective Disassembly Sequence Planning Subject to Operation Failures. In Proceedings of the Transactions on System, Man, and Cybernetics: Systems, Virtual. 17–20 October 2021. [[CrossRef](#)]
32. Kaur, A.; Kumar, Y. A new metaheuristic algorithm based on water wave optimization for data clustering. *Evol. Intell.* **2021**, 1–25. [[CrossRef](#)]
33. Kaveh, A.; Dadras, A. A novel meta-heuristic optimization algorithm: Thermal exchange optimization. *Adv. Eng. Softw.* **2017**, *110*, 69–84. [[CrossRef](#)]
34. Kaveh, A. *Advances in Metaheuristic Algorithms for Optimal Design of Structures*; Springer: Cham, Switzerland, 2021. [[CrossRef](#)]
35. Geem, Z.W.; Kim, J.H.; Loganathan, G.V. A new heuristic optimization algorithm: Harmony search. *Simulation* **2001**, *76*, 60–68. [[CrossRef](#)]
36. Abbasi, M.; Abbasi, E.; Mohammadi-Ivatloo, B. Single and multi-objective optimal power flow using a new differential-based harmony search algorithm. *J. Ambient. Intell. Humaniz. Comput.* **2020**, *12*, 851–871. [[CrossRef](#)]
37. Braik, M.; Ryalat, M.H.; Al-Zoubi, H. A novel meta-heuristic algorithm for solving numerical optimization problems: Ali Baba and the forty thieves. *Neural Comput. Appl.* **2021**, 1–47. [[CrossRef](#)]
38. Qi, Y.; Liu, J.; Yu, J. A Fireworks algorithm based path planning method for amphibious robot. In Proceedings of the 2021 IEEE International Conference on Real-time Computing and Robotics (RCAR), Xining, China, 15–19 July 2021. [[CrossRef](#)]
39. Tan, Y.; Zhu, Y. Fireworks Algorithm for Optimization. In *Lecture Notes in Computer Science*; Springer Science and Business Media LLC: Berlin/Heidelberg, Germany, 2010; pp. 355–364.
40. Osaba, E.; Yang, X.-S. Soccer-Inspired Metaheuristics: Systematic Review of Recent Research and Applications. *Appl. Optim. Swarm Intell.* **2021**, 81–102. [[CrossRef](#)]
41. 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](#)] [[PubMed](#)]
42. Talatahari, S.; Azizi, M. *Chaos Game Optimization: A Novel Metaheuristic Algorithm*; Springer: Berlin/Heidelberg, Germany, 2021; Volume 54. [[CrossRef](#)]
43. Sasmito, A.; Pratiwi, A.B. Stochastic fractal search algorithm in permutation flowshop scheduling problem. In Proceedings of the International Conference on Mathematics, Computational Sciences and Statistics 2020, Online, 29 September 2020; Volume 2329, p. 050003. [[CrossRef](#)]
44. Karami, H.; Sanjari, M.J.; Gharehpetian, G.B. Hyper-Spherical Search (HSS) algorithm: A novel meta-heuristic algorithm to optimize nonlinear functions. *Neural Comput. Appl.* **2014**, *25*, 1455–1465. [[CrossRef](#)]
45. Aguilar-Arguello, S.; Taylor, A.H.; Nelson, X.J. Jumping spiders attend to information from multiple modalities when preparing to jump. *Anim. Behav.* **2021**, *171*, 99–109. [[CrossRef](#)]
46. Götter, C. Locomotion of Spiders—What Robotics can Learn from Spiders and Vice Versa. Ph.D. Thesis, ETH Zurich, Zurich, Switzerland, 2021. [[CrossRef](#)]
47. Brandt, E.E.; Sasiharan, Y.; Elias, D.O.; Mhatre, N. Jump takeoff in a small jumping spider. *J. Comp. Physiol. A* **2021**, *207*, 153–164. [[CrossRef](#)] [[PubMed](#)]
48. GitHub-Mazhar-Ansari-Ardeh/BenchmarkFcns: A Collection of Mathematical Test Functions for Benchmarking Optimization Algorithms. Available online: <https://github.com/mazhar-ansari-ardeh/BenchmarkFcns> (accessed on 20 October 2021).
49. Suganthan, P.N.; Hansen, N.; Liang, J.J.; Deb, K.; Chen, Y.P.; Auger, A.; Tiwari, S. *Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization*; Nanyang Technological University: Singapore, 2005.
50. Kumar, A.; Wu, G.; Ali, M.Z.; Mallipeddi, R.; Suganthan, P.N.; Das, S. A test-suite of non-convex constrained optimization problems from the real-world and some baseline results. *Swarm Evol. Comput.* **2020**, *56*, 100693. [[CrossRef](#)]
51. Vazquez, H.P.; Torres-Huerta, A.M.; Flores-Vela, A. Self-Adaptive Differential Evolution Hyper-Heuristic with Applications in Process Design. *Comput. Syst.* **2016**, *20*, 173–193. [[CrossRef](#)]
52. Jain, N.; Parmar, G.; Gupta, R.; Khanam, I. Performance evaluation of GWO/PID approach in control of ball hoop system with different objective functions and perturbation. *Cogent Eng.* **2018**, *5*, 1465328. [[CrossRef](#)]