mathematics

MDPI

Article

Circle Search Algorithm: A Geometry-Based Metaheuristic Optimization Algorithm

Mohammed H. Qais [1], Hany M. Hasanien [2,*], Rania A. Turky [3], Saad Alghuwainem [4], Marcos Tostado-Véliz [5] and Francisco Jurado [5]

1 Centre for Advances in Reliability and Safety, Hong Kong; mohqais@cairs.hk
2 Electrical Power and Machines Department, Faculty of Engineering, Ain Shams University, Cairo 11517, Egypt
3 Electrical Engineering Department, Faculty of Engineering and Technology, Future University in Egypt, Cairo 11835, Egypt; rania.turky@fue.edu.eg
4 Electrical Engineering Department, College of Engineering, King Saud University, Riyadh 11421, Saudi Arabia; saadalgh@ksu.edu.sa
5 Department of Electrical Engineering, University of Jaén, 23700 Linares, Spain; mtostado@ujaen.es (M.T.-V.); fjurado@ujaen.es (F.J.)
* Correspondence: hanyhasanien@ieee.org

Abstract: This paper presents a novel metaheuristic optimization algorithm inspired by the geometrical features of circles, called the circle search algorithm (CSA). The circle is the most well-known geometric object, with various features including diameter, center, perimeter, and tangent lines. The ratio between the radius and the tangent line segment is the orthogonal function of the angle opposite to the orthogonal radius. This angle plays an important role in the exploration and exploitation behavior of the CSA. To evaluate the robustness of the CSA in comparison to other algorithms, many independent experiments employing 23 famous functions and 3 real engineering problems were carried out. The statistical results revealed that the CSA succeeded in achieving the minimum fitness values for 21 out of the tested 23 functions, and the p-value was less than 0.05. The results evidence that the CSA converged to the minimum results faster than the comparative algorithms. Furthermore, high-dimensional functions were used to assess the CSA's robustness, with statistical results revealing that the CSA is robust to high-dimensional problems. As a result, the proposed CSA is a promising algorithm that can be used to easily handle a wide range of optimization problems.

Keywords: algorithms; circle search algorithm; metaheuristics; numerical optimization; optimization methods

MSC: 49J55; 93E20

Citation: Qais, M.H.; Hasanien, H.M.; Turky, R.A.; Alghuwainem, S.; Tostado-Véliz, M.; Jurado, F. Circle Search Algorithm: A Geometry-Based Metaheuristic Optimization Algorithm. *Mathematics* **2022**, *10*, 1626. https://doi.org/10.3390/math10101626

Academic Editor: Mihai Postolache

Received: 18 April 2022
Accepted: 6 May 2022
Published: 10 May 2022

1. Introduction

Human life is becoming more intelligent, and the surrounding systems are growing, which is resulting in higher complexity and cost. Optimization algorithms are important in addressing these complex systems and maintaining the trade-off between quality and cost. In the past decades, conventional optimization algorithms, such as the Nelder and Mead algorithm [1] and the Hooke and Jeeve algorithm [2], have been used to solve small nonlinear problems. However, these algorithms suffer from several disadvantages, such as (1) their dependency on the initial conditions of the optimization problem; (2) their accuracy being based on the differential equation solver in the existing tools; and (3) their possibility of getting stuck at a local minimum point rather than the global minimum point, especially when the problem under study is a heavy nonlinear or nonconvex problem. Accordingly, metaheuristic algorithms have emerged as a viable option for solving complex problems. These algorithms are population-based and stochastic algorithms, which use the principle of random operators. Metaheuristic algorithms have a significant potential to solve constrained and nonlinear problems, which consider the

optimization problem as a black-box problem. In addition, metaheuristic algorithms can be classified into three main categories on the basis of the manner of inspiration: biology-based, nature-based, and physics-based.

Biology-based algorithms simulate the behavior of creatures in reproduction and evolution, swarming, or in search of food. The genetic algorithm (GA) [3] and particle swarm optimization (PSO) method [4] were two of the most popular stochastic algorithms that emerged in the last decades [5]. The GA is based on organisms reproducing their best offspring, whereas the PSO is based on bird and fish swarming. These two algorithms are regarded as the most commonly used algorithms. Following that, numerous algorithms emerged to mimic the evolution and swarming behavior of other creatures, including differential evolution (DE) that mimics the species evolution [6], artificial bee colony (ABC) that simulates bees foraging [7,8] and ant colony optimization (ACO) that stimulates ants' bio-communication in order to find the lowest path between the colony and food [9]. The bat algorithm (BA) mimics the echo positioning behavior of bats [10], symbiotic organisms search (SOS) replicates organisms' symbiotic interaction techniques for survival and reproduction in an ecology [11], and butterfly optimizer (ABO) replicates the mate-finding approach of several butterfly species [12]. The salp swarm algorithm (SSA) simulates salps' swarm behavior while traversing and foraging in the seas [13], the artificial gorilla troops optimizer (GTO) mimics the society of silverback gorillas [14], the grasshopper optimizer (GOA) mimics the grasshoppers' swarming behavior [15], and the moth–flame optimizer (MFO) simulates the flight of moths in a straight line toward the moon [16].

Other biology-based algorithms mimic the hunting behavior of wild creatures, including the hunting search algorithm (HuS), which mimics the hunting process of wild animals, such as lions and tigers, that surround a prey and catch it [17]; grey wolf optimization (GWO), which mimics the hierarchical order of grey wolves' herd to hunt prey [18]; the whale optimization algorithm (WOA), which mimics the spiral hunting of humpback whales [19], and the Harris hawks optimizer (HHO), which mimics hawks' collaboration to pounce on prey [20]. Moreover, the coyote optimizer algorithm (COA) mimics the social environment of these species of wolves [21], and the marine predators algorithm (MPA) mimics the Levy and Brownian motions of oceanic predators as well as the ecological interaction between predators and prey [22]. There are other biology-based algorithms, such as dolphin echolocation [23], the social spider algorithm [24], the shuffled frog-leaping algorithm [25], the honey badger algorithm [26], the cuckoo search algorithm [27], the artificial butterfly optimizer [12], the bird mating algorithm [28], the tunicate swarm algorithm [29,30], the pity beetle algorithm [31], the spotted hyena optimizer [32], and golden eagle optimization [33].

Nature-based metaheuristic algorithms imitate the regular natural performance of plants, water, and so on. The sunflower optimizer (SFO) simulates the position tracking behavior of sunflowers [34], the tree seed algorithm (TSA) imitates the relationship between trees growth and their seeds [35], and the flower pollination algorithm (FPA) simulates the pollination process of flowers [36]. The water cycle algorithm (WCA) simulates the water flow from rivers and waterways to the sea [37], the water evaporation optimizer (WEO) simulates the evaporation of a small number of the water components to a solid surface with varying water content [38], and the heat transfer search (HTS) mimics the interaction of heated molecules with each other and with their surroundings in order to achieve thermal equilibrium [39]. There are other nature-based metaheuristic algorithms, such as farmland fertility [40], the water strider algorithm [41], and the rain optimization algorithm [42].

Physics-based metaheuristic algorithms mimic the famous physical theories and phenomena [43]. The gravitational search algorithm (GSA) mimics the rule of gravity and masses interactions [44,45], the Lichtenberg algorithm (LA) mimics the Lichtenberg figures patterns [46], and Henry gas solubility optimization (HGSO) mimics Henry's rule of dissolved gases [47–49]. Thermal exchange optimizer (TEO) mimics the heat exchange between objects and surroundings [50], the colliding bodies optimizer (CBO) mimics the

state of two collided moving objects before and after collision [51], and the atom search optimizer (ASO) mimics the interaction forces between atoms [52]. Charged system search (CSS) simulates the electrostatics Coulomb law and the mechanics Newtonian laws [53], the ray optimization algorithm (ROA) simulates the traveling light from the bright medium to dark medium [54], and the transient search optimizer (TSO) simulates the transient response of electrical circuits that include the energy storage devices [55]. There are other physics-based metaheuristic algorithms, including Galactic swarm optimization [56], the central force optimizer [57], the ions motion algorithm [58], the newton metaheuristic algorithm [59], the gradient-based optimizer [60], atomic orbital search [61], chaos game optimization (CGO) [62], and the simulated annealing algorithm [63].

All metaheuristic algorithms compete to find the optimal solution; even a little change in the answer may result in substantial cost savings. All of these algorithms, however, started in their simplest form and were later enhanced or hybridized with other algorithms. The metaheuristic algorithms that can be applied easily and efficiently have attracted researchers to widely utilize them. PSO is one of the most utilized, hybridized, and modified to solve a broad range of optimization problems [64]. GWO, WOA, TSO, and SSA are modified and applied to design an optimal control system for wind power plants [65–69]. TSO, HHO, COA, and SFO are applied to estimate the electrical parameters of photovoltaic modules [70–74]. At the same time, the no-free-lunch theorem states that no algorithm can solve all problems [75], wherein one algorithm can efficiently solve a certain problem while failing to solve another problem. This encourages researchers to model new effective algorithms that can solve more optimization problems. Table 1 summarizes the most recent emerging metaheuristic algorithms, and it can be indicated that there is a need for new competitive and effortless algorithms.

Table 1. State of the art metaheuristic algorithms.

Algorithm	Name	Reference	Classification	Mimicking
Orca predation algorithm	OPA	[76]	Biology-based (hunters)	The orcas' hunting habit.
Komodo mlipir algorithm	KMA	[77]	Biology-based	Komodo dragons and miliper foraging and reproduction
Integrated optimization algorithm	IOA	[78]	Evolutionarily based	Follower search, leader search, wanderer search, crossover search, and role learning are all terms used to construct IOA
Reptile search algorithm	RSA	[79]	Biology-based (hunters)	Crocodiles' hunting habit
African vultures optimization	AVOA	[80]	Biology-based	African vultures' feeding and navigational behaviors
Elephant clan optimization	ECO	[81]	Biology-based	Elephants' clan behavior.
Cooperation search algorithm	CoSA	[82]	Human-learning-based	The behaviors of teamwork in contemporary business
Group teaching optimization	GTOA	[83]	Human-learning-based	The relationship between the instructor and his or her pupils
Mayfly algorithm	MA	[84]	Biology-based	Mayfly flying and mating behavior
Search and rescue optimization algorithm	SAR	[85]	Human-learning-based	The study of human behavior during search and rescue missions

This article proposes a novel metaheuristic algorithm, called CSA, which can be categorized as a geometry-based metaheuristic algorithm. Within this category, the sine cosine algorithm (SCA) is a geometry-based method that emulates sinusoidal waveforms [86]. Geometry is the science that deals with the characteristics of figures in space, such as their dimensions, relative position, distance, form, and size. The circle is the most frequently

used geometric figure because it possesses unique characteristics such as a diameter, a perimeter, a center point, and a tangent line. The radius that crosses the tangent point is perpendicular to the tangent line, and the orthogonal function is the ratio of the radius to the perpendicular tangent line. The orthogonal function varies significantly with a tiny angle change, which may speed up the CSA exploration phase.

The major contributions of this work are listed below:

1. Introducing a novel geometry-based optimization method, called CSA.
2. Presenting a mathematical model for the proposed CSA, including the states of exploration and exploitation processes.
3. Applying the proposed CSA and other comparative algorithms to determine the optimal solution of 23 well-known functions and three engineering design issues
4. Applying the CSA to solve high-dimensional functions (100 and 1000 dimensions).
5. Testing the superiority and significance of the CSA in comparison with other algorithms, performed by using a variety of statistical tests, including the mean, standard deviation, rank test, and *p*-values.

The rest of the paper is structured as follows: Section 2 provides the specifics of the inspiration and the modeling of the CSA. Section 3 shows and contrasts the optimum results of the 23 standard functions that are obtained using the CSA and other experienced algorithms. Section 4 illustrates the use of the CSA to optimally design three well-known engineering issues. Section 5 provides a short conclusion regarding this study.

2. Circle Search Algorithm

2.1. Background

The geometrical circle is a fundamental closed curve with the same distance between all points and the center. As shown in Figure 1, the diameter is defined as the line connecting two points on the curve that intersect the center (x_c). The radius (R) is the line that links any point on the circle to the center. The perimeter of a circle is equal to the length of the curve that surrounds it. As observed in Figure 1, the tangent line segment is a straight line that intersects the circle at a single point (x_t) and is perpendicular to the radius intersecting this point. According to Pythagorean equations, the orthogonal function (Tan) of the right triangle is the ratio between the radius and the perpendicular tangent line segment. It is obvious that the radius is defined as the distance between x_t and x_c, whereas the tangent line segment is the distance between points x_t and x_p; then, the orthogonal function (Tan) is expressed as in the following equations:

$$\mathrm{Tan}(\theta) = \frac{x_t - x_c}{x_p - x_t} \tag{1}$$

$$x_t - x_c = (x_p - x_t) \times \mathrm{Tan}(\theta) \tag{2}$$

$$x_t = x_c + (x_p - x_t) \times \mathrm{Tan}(\theta) \tag{3}$$

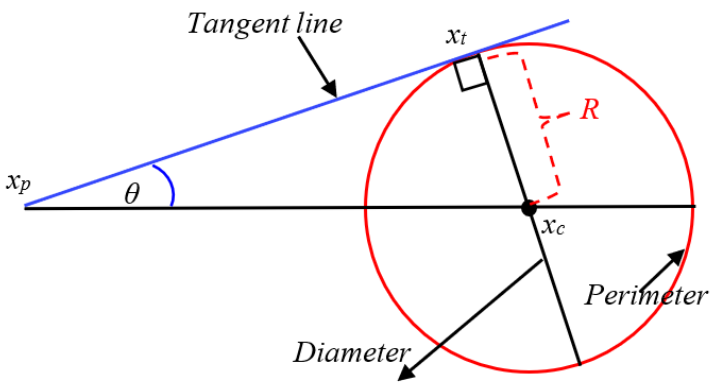
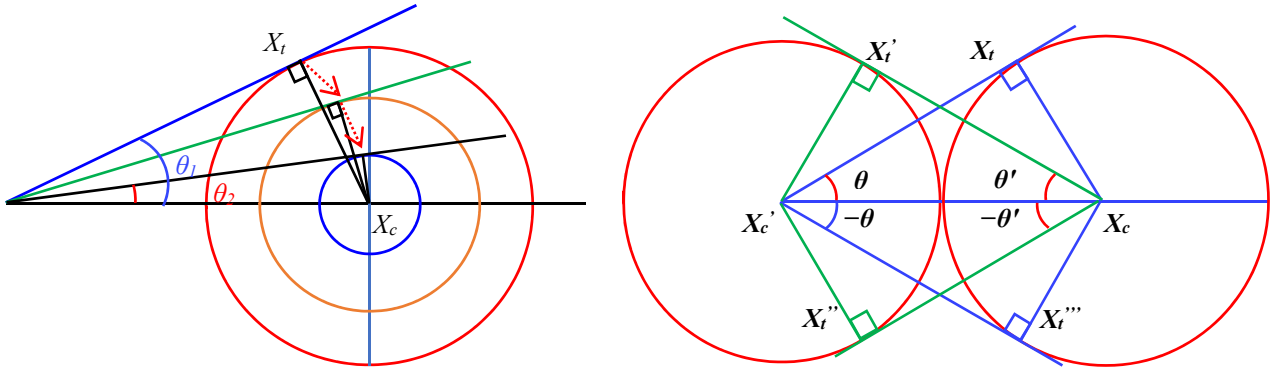


Figure 1. Terminologies of the geometric circle.

2.2. CSA Formulation

The CSA looks for the optimum answer inside random circles in order to broaden the scope of the search area. Using the center of the circle as a target point, the angle of the contacting point of the tangent line and the circumference of the circle progressively decreases until it approaches the center of the circle, as shown in Figure 2a. Due to the possibility that this circle has been trapped in the local solution, the angle at which the tangent line touches the point is altered at random, as shown in Figure 2b. The touching point X_t is considered the search agent of the CSA, and the center point X_c is assumed to be the best position in the algorithm. As shown in Figure 2, the CSA updates the search agent in response to the movement of the touching point toward the center. Nevertheless, to prevent the CSA from being stuck in a local solution, the contact point will be updated randomly by changing the angle in a random manner. The main steps of the CSA optimizer are explained below:



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

Figure 2. The processes of the CSA algorithm (**a**) exploitation; (**b**) exploration.

Step 1: Initialization: This step is important in the CSA, where whole dimensions of each search agent should be equally randomized, as depicted in Algorithm 1. Most of the previous published code randomizes the dimensions unequally, which sometimes make the algorithms surprisingly obtain the best results. Then, the search agents are initialized between the upper limit values (UB) and lower limit values (LB) of the search space as in Equation (4):

$$\mathbf{X_t} = \mathrm{LB} + r \times (\mathrm{UB} - \mathrm{LB}) \tag{4}$$

where r is a random vector that lies between [0, 1].

Step 2: Update search agent position: the position of the search agents $\mathbf{X_t}$ is updated according to the evaluated best position $\mathbf{X_c}$ as shown in Equation (5):

$$\mathbf{X_t} = \mathbf{X_c} + (\mathbf{X_c} - \mathbf{X_t}) \times \tan(\theta) \tag{5}$$

where the angle θ plays an important role in the exploration and exploitation of the CSA and can be calculated as follows:

$$\theta = \begin{cases} w \times rand & Iter > (\mathrm{c} \times Maxiter) \ \text{(escape from local stagnation)} \\ w \times p & \text{otherwise} \end{cases} \tag{6}$$

$$w = w \times rand - w \tag{7}$$

$$a = \pi - \pi \times \left(\frac{Iter}{Maxiter}\right)^2 \tag{8}$$

$$p = 1 - 0.9 \times \left(\frac{Iter}{Maxiter}\right)^{0.5} \tag{9}$$

where *rand* is a random number lies between 0 and 1, *Iter* stands for the iteration counter, *Maxiter* represents the maximum number of iterations, and c is a constant between 0 and 1 that represents the percentage of maximum iterations. Equation (7) shows that the variable w changes from $-\pi$ to 0 during the increasing of the iterations number. The variable a changes from π to 0 according to Equation (8). The variable p changes from 1 to 0 as in Equation (9). Accordingly, the angle θ changes from $-\pi$ to 0.

There are two cases can be achieved for the CSA as follows:

1. ***Case 1:*** *Iter* > (*c.Maxiter*): this case means that the angle $\theta = w \times rand$ all the time, which can applied to improve the exploration process of the CSA and escape the local stagnation.
2. ***Case 2:*** *Iter* < (*c.Maxiter*): this case makes the angle $\theta = w \times p$ all the time, which can be used to improve the exploitation process of the CSA.

The pseudo code of the CSA is summarized in the Algorithm 2. Furthermore, Figure 3 exhibits the flowchart of the CSA.

Algorithm 1 Initialization of the CSA

Input **LB** *and* **UB**.
Do *for all search agents*
r = random number between [0, 1].
Use Equation (4) to initialize the search agent $\mathbf{X_t}$.
End Do

Algorithm 2 Pseudo-code of the CSA

Initialize the search agents $\mathbf{X_t}$ *using Algorithm 1*
Input the constant value c*, Iter = 0, and Maxiter*
While *Iter less than Maxiter*
Use Equation (8) to find the value of a
Do *for all search agents*
Use Equation (7) to find the value of w
Use Equation (9) to find the value of p
Use Equation (6) to find the value of the angle θ
Use Equation (5) to update the search agent $\mathbf{X_t}$
if *the updated search agents are out of the boundaries* ***then*** *set search agents equal to the boundaries*
find the fitness function f($\mathbf{X_t}$*)*
End Do
Evaluate the f($\mathbf{X_t}$*) with the stored best solution f(*$\mathbf{X_c}$*)*
Update f($\mathbf{X_c}$*) and* $\mathbf{X_c}$
Iter = Iter + 1
End While
Output f($\mathbf{X_c}$*) and* $\mathbf{X_c}$

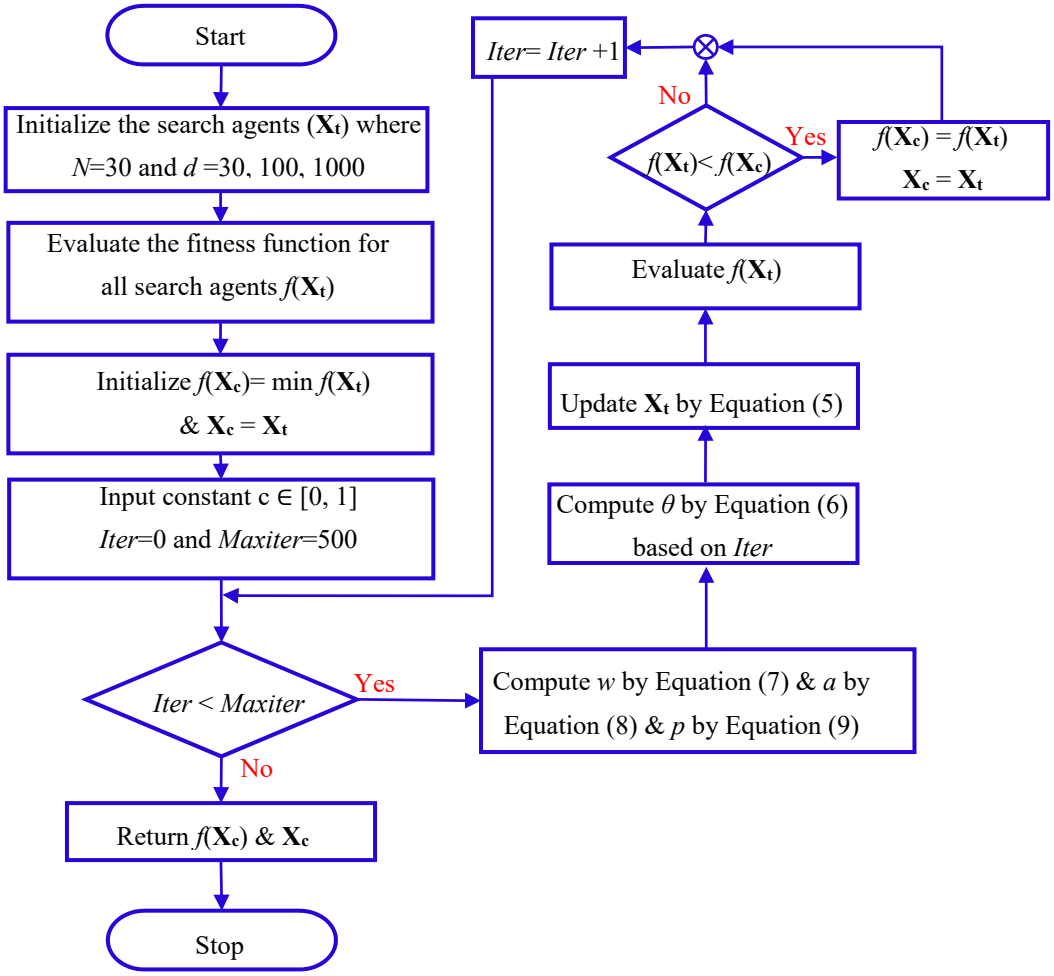


Figure 3. Flowchart of the CSA.

3. Computational Complexity of the CSA

The Big O Notation can be used to quantify the runtime of the CSA. The initialization is based on one loop as described in Algorithm 1, so it requires $O(N)$, where N denotes the search agents' number. The computational complexity of the update process is $O(Maxiter \times N \times D)$. The computational complexity of the fitness evaluation is $O(Maxiter \times N)$. Therefore, the total complexity is $O(N(1 + Maxiter \times D + Maxiter)$, where the computational complexity of the CSA is $O(N \times Maxiter \times D)$.

4. Experimental Results and Discussion

In this article, the optimization processes were performed using MATLAB R2019b, Windows 10 64 bits on Core i7, 16 GB RAM PC to find the best solution of the famous 23 standard functions.

4.1. Standard Functions

These standard functions are well known and used widely as benchmark functions, including the unimodal, multimodal, and fixed-dimension multimodal functions. Table 2 shows the list of the unimodal functions (F1–F7), which have one optimal best solution, and they were used to check the deep exploitation of the optimization algorithms. Tables 3 and 4 list the non-fixed multimodal functions (F8–F13) and fixed dimensions of the multimodal functions (F14–F23), which trap the algorithms into many local optimal solutions. These multimodal functions are used to examine the exploration of the algorithms and their ability to escape from the local optima.

Table 2. Unimodal standard famous functions.

Function	Expression	Dimension (*d*)	Solution Space	Best Solution
F1	$f(x) = \sum_{i=1}^{d} {x_i}^2$	30, 100, 1000	$[-100, 100]^d$	0
F2	$f(x) = \sum_{i=1}^{d} \lvert x_i \rvert + \prod_{i=1}^{d} \lvert x_i \rvert$	30, 100, 1000	$[-10, 10]^d$	0
F3	$f(x) = \sum_{i=1}^{d} (\sum_{j=1}^{i} x_j)^2$	30, 100, 1000	$[-100, 100]^d$	0
F4	$f(x) = \max_i \{\lvert x_i \rvert, 1 \le i \le d\}$	30, 100, 1000	$[-100, 100]^d$	0
F5	$f(x) = \sum_{i=1}^{d} [100({x_{i+1}} - {x_i}^2)^2 + (x_i - 1)^2]$	30, 100, 1000	$[-30, 30]^d$	0
F6	$f(x) = \sum_{i=1}^{d} (x_i + 0.5)^2$	30, 100, 1000	$[-100, 100]^d$	0
F7	$f(x) = \sum_{i=1}^{d} i.{x_i}^4 + random[0, 1)$	30, 100, 1000	$[-1.28, 1.28]^d$	0

Table 3. Multimodal standard famous functions.

Function	Expression	Dimension (*d*)	Solution Space	Best Solution
F8	$f(x) = \sum_{i=1}^{d} x_i \sin(\sqrt{\lvert x_i \rvert})$	30, 100, 1000	$[-500, 500]^d$	$-418.9829 \times d$
F9	$f(x) = \sum_{i=1}^{n} [{x_i}^2 - 10\cos(2\pi x_i) + 10]$	30, 100, 1000	$[-5.12, 5.12]^d$	0
F10	$f(x) = -20\exp(-0.2\sqrt{\frac{1}{d}\sum_{j=1}^{d} x_j}) - \exp(\frac{1}{n}\cos(2\pi x_j)) + 20 + e$	30, 100, 1000	$[-32, 32]^d$	0
F11	$f(x) = \frac{1}{4000}\sum_{i=1}^{d} {x_i}^2 - \prod_{i=1}^{d} \cos(\frac{x_i}{\sqrt{i}}) + 1$	30, 100, 1000	$[-600, 600]^d$	0
F12	$f(x) = \frac{\pi}{d}\{10\sin(\pi y_1) + \sum_{i=1}^{d} (y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_d - 1)^2\}$ $+ \sum_{i=1}^{d} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i + 1}{4}$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	30, 100, 1000	$[-50, 50]^d$	0
F13	$f(x) = 0.1 \times \{10\sin^2(3\pi x_1) + \sum_{i=1}^{d} (x_i - 1)^2[1 + \sin^2(3\pi x_i + 1)]$ $+ (x_d - 1)^2[1 + \sin^2(2\pi x_d)]\} + \sum_{i=1}^{d} u(x_i, 5, 100, 4)$	30, 100, 1000	$[-50, 50]^d$	0

Table 4. Fixed-dimension multimodal standard famous functions.

Function	Expression	Dimension (*d*)	Solution Space	Best Solution
F14	$f(x) = (\frac{1}{500} + \sum_{i=1}^{25} \frac{1}{i + \sum_{j=1}^{2} (x_i - a_{ij})^6})^{-1}$	2	$[-65, 65]^d$	1
F15	$f(x) = \sum_{i=1}^{11} [a_i - \frac{x_1({b_i}^2 + b_i x_2)}{{b_i}^2 + b_i x_3 + x_4}]^2$	4	$[-5, 5]^d$	0.00030
F16	$f(x) = 4{x_1}^2 - 2.1{x_1}^4 + \frac{1}{3}{x_1}^6 + x_1 x_2 - 4{x_2}^2 + 4{x_2}^4$	2	$[-5, 5]^d$	-1.0316
F17	$f(x) = (x_2 - \frac{5.1}{4\pi^2}{x_1}^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos(x_1) + 10$	2	$[-5, 5]^d$	0.398
F18	$f(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3{x_1}^2 - 14x_2 + 6x_1 x_2 + 3{x_2}^2)] \times$ $[30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12{x_1}^2$ $+ 48x_2 - 36x_1 x_2 + 27{x_2}^2)]$	2	$[-2, 2]^d$	3
F19	$f(x) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2)$	3	$[1, 3]^d$	-3.86

Table 4. *Cont.*

Function	Expression	Dimension (*d*)	Solution Space	Best Solution
F20	$f(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right)$	6	$[0, 1]^d$	−3.32
F21	$f(x) = -\sum_{i=1}^{5} \left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	4	$[0, 10]^d$	−10.1532
F22	$f(x) = -\sum_{i=1}^{7} \left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	4	$[0, 10]^d$	−10.4028
F23	$f(x) = -\sum_{i=1}^{10} \left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	4	$[0, 10]^d$	−10.5363

4.2. Comparative Algorithms

The CSA performance was compared with eight algorithms—the PSO, GWO, SSA, SCA, WOA, HHO, CGO, and TSO algorithms. The PSO and GWO algorithms are the most used algorithms, so they can be considered as the benchmark algorithms for all new algorithms. The WOA, CGO, and TSO algorithms are recent algorithms, which are applied widely due to their straightforward coding and application. The SSA and HHO are recent algorithms and have good capability to avoid being stuck in local optima. The SCA is a geometry-based algorithm, where the proposed CSA is also classified as a geometry-based algorithm. The settings of the applied algorithms in this paper are displayed in Table 5, which were tuned on the basis of the related literature. All algorithms had the same maximum number of iterations (500), and the search agents were 30 in number.

Table 5. Parameters of the applied algorithms.

Algorithm	Parameters
Proposed CSA	w decreased from 1.5 to 0 and constant c = 0.75 for F1–F13 and c = 0.3 for F14–F23
PSO	Inertia weight w decreased from 0.5 to 0.3, c_1 = 2, and c_2 = 2
GWO	The parameter a changed from 2 to 0
SSA	Probability update was 0.5
SCA	Constant a = 2 and probability update was 0.5
WOA	The parameter a changed from 2 to 0, b = 1
HHO	The decreasing energy $E1$ changed from 2 to 0
CGO	α, β, and γ were random numbers
TSO	The parameter a changed from 2 to 0

4.3. Statistical Analysis

Statistical analysis was carried out for 30 independent runs for all applied algorithms. For obtaining a fair comparison, the same initial population was used for all compared algorithms. The mean and standard deviation were the most used statistical tests to show the superiority of the algorithms to obtain the best minimum for different runs. Tables 6–8 show the minimum, mean, standard deviation (std), and ranks for all applied algorithms and functions. The best achieved mean solution was bolded for visual inspection, wherein the CSA obtained the best mean solution for 21 out of 23 functions. Furthermore, the standard deviations of the results of 30 independent runs for all functions were very small, especially for the Shwefel function (F8), where all algorithms obtained very big standard deviations for this function. The sum of individual ranks showed that the CSA obtained the first rank, whereas the SSA obtained the second rank. The balance between exploitation and exploration abilities of the algorithms can be tested using two contrary functions (Rosenbrock function (F5) and Rastrigin function (F9)), wherein most of the algorithms failed to solve both of them. However, the proposed CSA successfully obtained the best minimum solution for both.

Table 6. Statistical results of the optimized standard functions with $d = 30$.

		GWO	SCA	SSA	HHO	WOA	PSO	TSO	CGO	CSA
F1	Avg.	5.7838×10^{-38}	6.5806×10^{-11}	9.5464×10^{-08}	1.4759×10^{-109}	6.6351×10^{-69}	4.6281×10^{-07}	1.0826×10^{-03}	1.1934×10^{-136}	9.5326×10^{-219}
	STD.	9.1734×10^{-38}	2.7669×10^{-10}	5.2552×10^{-08}	8.0839×10^{-109}	3.6342×10^{-68}	1.0552×10^{-06}	3.9713×10^{-03}	4.7846×10^{-136}	0.0000×10^{00}
	Min	1.1944×10^{-40}	1.9641×10^{-15}	3.3386×10^{-08}	0.0000×10^{00}	6.6909×10^{-86}	2.3249×10^{-10}	8.9536×10^{-08}	0.0000×10^{00}	2.9648×10^{-278}
F2	Avg.	1.8357×10^{-22}	3.1710×10^{-08}	2.6003×10^{-01}	3.4043×10^{-53}	6.7368×10^{-50}	5.3551×10^{-04}	8.4374×10^{-06}	1.7566×10^{-71}	1.3380×10^{-92}
	STD.	3.6643×10^{-22}	4.5318×10^{-08}	1.8497×10^{-01}	1.2317×10^{-52}	3.3959×10^{-49}	8.9901×10^{-04}	1.2036×10^{-05}	7.9508×10^{-71}	7.3287×10^{-92}
	Min	1.1603×10^{-23}	1.1753×10^{-09}	4.5012×10^{-03}	9.7243×10^{-172}	1.1799×10^{-57}	2.8776×10^{-06}	6.5020×10^{-07}	0.0000×10^{00}	3.4777×10^{-140}
F3	Avg.	1.2328×10^{-08}	1.0265×10^{-07}	3.0191×10^{02}	5.3195×10^{-78}	1.7475×10^{-01}	1.0393×10^{03}	3.9684×10^{02}	1.0135×10^{-98}	3.5072×10^{-192}
	STD.	3.2665×10^{-08}	3.1441×10^{-07}	4.7355×10^{02}	2.0975×10^{-77}	4.9298×10^{-01}	9.4985×10^{02}	1.0181×10^{03}	2.6061×10^{-98}	0.0000×10^{00}
	Min	3.4277×10^{-12}	9.8383×10^{-13}	1.1474×10^{00}	4.6827×10^{-103}	7.1459×10^{-08}	3.8895×10^{-02}	4.5120×10^{-04}	0.0000×10^{00}	1.2646×10^{-274}
F4	Avg.	5.7254×10^{-01}	1.0814×10^{00}	1.0281×10^{00}	1.5750×10^{-119}	4.8675×10^{-02}	3.0927×10^{00}	7.2259×10^{-01}	6.5523×10^{-59}	1.2504×10^{-98}
	STD.	1.0996×10^{00}	2.5084×10^{00}	8.5943×10^{-01}	8.6266×10^{-119}	1.3524×10^{-01}	2.9911×10^{00}	6.7465×10^{-01}	1.2075×10^{-58}	6.8486×10^{-98}
	Min	4.2053×10^{-08}	9.9628×10^{-05}	2.8062×10^{-02}	8.0269×10^{-181}	6.5725×10^{-06}	2.8062×10^{-02}	6.0508×10^{-03}	0.0000×10^{00}	3.0109×10^{-139}
F5	Avg.	2.5381×10^{01}	7.0200×10^{01}	2.5675×10^{01}	1.8902×10^{-06}	8.5474×10^{00}	1.9573×10^{01}	1.1420×10^{01}	1.5351×10^{01}	0.0000×10^{00}
	STD.	1.5260×10^{01}	1.3659×10^{02}	2.4616×10^{01}	1.0192×10^{-05}	1.2854×10^{01}	2.3932×10^{01}	1.2831×10^{01}	5.4322×10^{00}	0.0000×10^{00}
	Min	7.5169×10^{-02}	7.5169×10^{-02}	9.3447×10^{-04}	0.0000×10^{00}	2.3611×10^{-07}	4.2949×10^{-05}	4.6622×10^{-03}	3.2114×10^{-06}	0.0000×10^{00}
F6	Avg.	5.4155×10^{-01}	5.3173×10^{00}	2.0211×10^{-07}	2.8436×10^{-07}	1.3848×10^{-02}	4.7315×10^{-07}	4.8451×10^{-01}	1.8244×10^{-18}	0.0000×10^{00}
	STD.	3.5419×10^{-01}	8.0594×10^{-01}	3.5045×10^{-07}	1.4406×10^{-06}	2.4111×10^{-02}	8.7532×10^{-07}	4.2675×10^{-01}	5.9259×10^{-18}	0.0000×10^{00}
	Min	2.8333×10^{-04}	1.2272×10^{00}	2.4022×10^{-08}	0.0000×10^{00}	7.5679×10^{-08}	5.0032×10^{-09}	3.5827×10^{-02}	6.2486×10^{-24}	0.0000×10^{00}
F7	Avg.	1.3202×10^{-03}	4.7389×10^{-04}	4.0651×10^{-02}	1.0839×10^{-04}	1.2714×10^{-03}	2.7046×10^{-02}	2.5284×10^{-03}	4.6345×10^{-04}	3.8180×10^{-04}
	STD.	6.4299×10^{-04}	4.4640×10^{-04}	4.0466×10^{-02}	1.0664×10^{-04}	2.4965×10^{-03}	2.5014×10^{-02}	2.8216×10^{-03}	3.4197×10^{-04}	6.6990×10^{-04}
	Min	2.8985×10^{-04}	4.5957×10^{-06}	2.1466×10^{-03}	1.3803×10^{-05}	7.2309×10^{-07}	7.7340×10^{-04}	1.1353×10^{-04}	7.3420×10^{-05}	2.6012×10^{-05}
F8	Avg.	-1.1269×10^{04}	-1.1117×10^{04}	-1.2096×10^{04}	-1.2569×10^{04}	-1.2427×10^{04}	-1.2071×10^{04}	-1.2223×10^{04}	-1.2451×10^{04}	-1.2569×10^{04}
	STD.	1.6040×10^{03}	1.5893×10^{03}	1.2285×10^{03}	2.3541×10^{-09}	6.5709×10^{02}	1.2389×10^{03}	9.5254×10^{02}	6.4871×10^{02}	1.9404×10^{-12}
	Min	-1.2557×10^{04}	-1.2551×10^{04}	-1.2569×10^{04}	-1.2569×10^{04}	-1.2569×10^{04}	-1.2569×10^{04}	-1.2569×10^{04}	-1.2569×10^{04}	-1.2569×10^{04}
F9	Avg.	4.1807×10^{01}	4.0559×10^{01}	9.9496×10^{00}	0.0000×10^{00}	1.8948×10^{-15}	3.0214×10^{01}	2.2908×10^{00}	0.0000×10^{00}	0.0000×10^{00}
	STD.	3.5269×10^{01}	4.2016×10^{01}	1.4311×10^{01}	0.0000×10^{00}	1.0378×10^{-14}	3.3687×10^{01}	1.2547×10^{01}	0.0000×10^{00}	0.0000×10^{00}
	Min	0.0000×10^{00}	0.0000×10^{00}	2.3251×10^{-08}	0.0000×10^{00}	0.0000×10^{00}	2.7281×10^{-08}	1.7390×10^{-08}	0.0000×10^{00}	0.0000×10^{00}

Table 6. *Cont.*

		GWO	SCA	SSA	HHO	WOA	PSO	TSO	CGO	CSA
F10	Avg.	1.1978×10^{-01}	2.5187×10^{-01}	9.4409×10^{-01}	8.8818×10^{-16}	5.0330×10^{-15}	9.1797×10^{-01}	1.3331×10^{-01}	2.7830×10^{-15}	8.8818×10^{-16}
	STD.	6.5604×10^{-01}	9.5974×10^{-01}	1.2595×10^{00}	0.0000×10^{00}	2.9626×10^{-15}	1.9652×10^{00}	7.2493×10^{-01}	1.8027×10^{-15}	0.0000×10^{00}
	Min	7.9936×10^{-15}	6.1332×10^{-09}	4.1912×10^{-05}	8.8818×10^{-16}	8.8818×10^{-16}	8.9230×10^{-06}	7.4523×10^{-06}	8.8818×10^{-16}	8.8818×10^{-16}
F11	Avg.	3.7391×10^{-03}	2.1526×10^{-07}	1.8495×10^{-02}	0.0000×10^{00}	0.0000×10^{00}	1.1234×10^{-02}	6.6144×10^{-02}	0.0000×10^{00}	0.0000×10^{00}
	STD.	1.0062×10^{-02}	1.1396×10^{-06}	1.4792×10^{-02}	0.0000×10^{00}	0.0000×10^{00}	1.3511×10^{-02}	1.5859×10^{-01}	0.0000×10^{00}	0.0000×10^{00}
	Min	0.0000×10^{00}	3.2196×10^{-15}	4.8962×10^{-04}	0.0000×10^{00}	0.0000×10^{00}	2.9305×10^{-09}	1.8765×10^{-08}	0.0000×10^{00}	0.0000×10^{00}
F12	Avg.	1.2064×10^{00}	1.4563×10^{00}	1.9961×10^{-02}	1.0540×10^{-08}	3.7369×10^{-04}	7.9748×10^{-01}	7.4001×10^{-03}	1.3909×10^{-20}	1.5705×10^{-32}
	STD.	2.5890×10^{00}	2.4947×10^{00}	7.7080×10^{-02}	3.8589×10^{-08}	1.1984×10^{-03}	1.7667×10^{00}	1.2526×10^{-02}	6.6394×10^{-20}	5.5674×10^{-48}
	Min	1.6660×10^{-05}	2.1551×10^{-04}	4.8395×10^{-07}	6.3387×10^{-21}	2.1843×10^{-09}	3.6269×10^{-13}	1.3758×10^{-06}	1.0149×10^{-25}	1.5705×10^{-32}
F13	Avg.	2.3921×10^{-01}	1.8630×10^{00}	4.0527×10^{-01}	7.6694×10^{-07}	2.4754×10^{-03}	2.3928×10^{-01}	1.5123×10^{-01}	2.2315×10^{-02}	1.3498×10^{-32}
	STD.	2.3561×10^{-01}	9.1497×10^{-01}	1.5566×10^{00}	3.5053×10^{-06}	6.0994×10^{-03}	1.2677×10^{00}	1.5863×10^{-01}	5.2790×10^{-02}	5.5674×10^{-48}
	Min	5.3035×10^{-04}	3.4038×10^{-02}	7.9766×10^{-06}	1.3498×10^{-32}	2.8468×10^{-09}	1.1833×10^{-10}	6.0890×10^{-05}	8.1780×10^{-24}	1.3498×10^{-32}
F14	Avg.	3.2156×10^{00}	1.9345×10^{00}	1.0643×10^{00}	9.9800×10^{-01}	9.9800×10^{-01}	1.0311×10^{00}	9.9800×10^{-01}	9.9800×10^{-01}	9.9800×10^{-01}
	STD.	3.8889×10^{00}	9.9707×10^{-01}	2.5219×10^{-01}	1.5699×10^{-10}	1.3046×10^{-08}	1.8148×10^{-01}	1.7835×10^{-07}	0.0000×10^{00}	1.7494×10^{-16}
	Min	9.9800×10^{-01}	9.9800×10^{-01}	9.9800×10^{-01}	9.9800×10^{-01}	9.9800×10^{-01}	9.9800×10^{-01}	9.9800×10^{-01}	9.9800×10^{-01}	9.9800×10^{-01}
F15	Avg.	4.0459×10^{-04}	5.7433×10^{-04}	6.4506×10^{-04}	3.2725×10^{-04}	3.2225×10^{-04}	4.6375×10^{-04}	3.7770×10^{-04}	3.3801×10^{-04}	3.0806×10^{-04}
	STD.	1.1637×10^{-04}	2.5944×10^{-04}	3.3345×10^{-04}	2.2421×10^{-05}	2.5146×10^{-05}	2.4872×10^{-04}	1.7279×10^{-04}	1.6718×10^{-04}	7.8742×10^{-07}
	Min	3.0749×10^{-04}	3.3338×10^{-04}	3.0769×10^{-04}	3.0751×10^{-04}	3.0784×10^{-04}	3.0749×10^{-04}	3.0758×10^{-04}	3.0749×10^{-04}	3.0749×10^{-04}
F16	Avg.	-1.0316×10^{00}	-1.0315×10^{00}	-1.0316×10^{00}	-1.0316×10^{00}	-1.0316×10^{00}	-1.0316×10^{00}	-1.0316×10^{00}	-1.0316×10^{00}	-1.0316×10^{00}
	STD.	7.0549×10^{-08}	8.7695×10^{-05}	4.0464×10^{-14}	3.0122×10^{-09}	3.6944×10^{-09}	6.6486×10^{-16}	1.5990×10^{-05}	6.7752×10^{-16}	4.6137×10^{-09}
	Min	-1.0316×10^{00}	-1.0316×10^{00}	-1.0316×10^{00}	-1.0316×10^{00}	-1.0316×10^{00}	-1.0316×10^{00}	-1.0316×10^{00}	-1.0316×10^{00}	-1.0316×10^{00}
F17	Avg.	3.9789×10^{-01}	4.0100×10^{-01}	3.9789×10^{-01}	3.9789×10^{-01}	3.9790×10^{-01}	3.9789×10^{-01}	3.9791×10^{-01}	3.9789×10^{-01}	3.9789×10^{-01}
	STD.	5.3714×10^{-06}	3.2491×10^{-03}	7.3008×10^{-15}	5.2004×10^{-06}	1.7830×10^{-05}	0.0000×10^{00}	2.2182×10^{-05}	0.0000×10^{00}	5.2398×10^{-08}
	Min	3.9789×10^{-01}	3.9811×10^{-01}	3.9789×10^{-01}	3.9789×10^{-01}	3.9789×10^{-01}	3.9789×10^{-01}	3.9789×10^{-01}	3.9789×10^{-01}	3.9789×10^{-01}
F18	Avg.	3.0000×10^{00}	3.0000×10^{00}	3.0000×10^{00}	3.0000×10^{00}	3.0001×10^{00}	3.0000×10^{00}	2.9715×10^{01}	3.0000×10^{00}	3.0000×10^{00}
	STD.	4.1823×10^{-05}	3.5195×10^{-05}	1.8243×10^{-13}	6.0229×10^{-07}	4.6026×10^{-04}	9.9301×10^{-16}	5.0596×10^{00}	9.3663×10^{-16}	4.9599×10^{-06}
	Min	3.0000×10^{00}	3.0000×10^{00}	3.0000×10^{00}	3.0000×10^{00}	3.0000×10^{00}	3.0000×10^{00}	3.0032×10^{00}	3.0000×10^{00}	3.0000×10^{00}
F19	Avg.	-3.8616×10^{00}	-3.8507×10^{00}	-3.8628×10^{00}	-3.8583×10^{00}	-3.8518×10^{00}	-3.8628×10^{00}	-3.8040×10^{00}	-3.8628×10^{00}	-3.8625×10^{00}

Table 6. *Cont.*

		GWO	SCA	SSA	HHO	WOA	PSO	TSO	CGO	CSA
	STD.	2.2933×10^{-05}	3.1389×10^{-01}	5.5503×10^{-02}	1.0806×10^{-01}	3.4871×10^{-02}	7.1334×10^{-02}	9.1886×10^{-03}	5.8273×10^{-02}	4.1813×10^{-02}
	Min	-3.3220×10^{00}	-3.0564×10^{00}	-3.3220×10^{00}	-3.1993×10^{00}	-3.3217×10^{00}	-3.3220×10^{00}	-3.3211×10^{00}	-3.3220×10^{00}	-3.3220×10^{00}
F21	Avg.	-8.3916×10^{00}	-6.8959×10^{00}	-1.0153×10^{01}	-9.8130×10^{00}	-9.8109×10^{00}	-8.9697×10^{00}	-1.0128×10^{01}	-1.0153×10^{01}	-1.0153×10^{01}
	STD.	2.1174×10^{00}	2.2821×10^{00}	5.5169×10^{-11}	1.2933×10^{00}	1.2928×10^{00}	2.1819×10^{00}	2.7575×10^{-02}	6.7923×10^{-15}	1.4067×10^{-07}
	Min	-1.0152×10^{01}	-1.0152×10^{01}	-1.0153×10^{01}	-1.0153×10^{01}	-1.0153×10^{01}	-1.0153×10^{01}	-1.0153×10^{01}	-1.0153×10^{01}	-1.0153×10^{01}
F22	Avg.	-8.9990×10^{00}	-7.2308×10^{00}	-1.0227×10^{01}	-9.8713×10^{00}	-9.8701×10^{00}	-8.3816×10^{00}	-1.0391×10^{01}	-1.0403×10^{01}	-1.0403×10^{01}
	STD.	2.2109×10^{00}	2.4640×10^{00}	9.6292×10^{-01}	1.6218×10^{00}	1.6214×10^{00}	2.7339×10^{00}	1.0467×10^{-02}	1.3601×10^{-15}	3.3477×10^{-05}
	Min	-1.0402×10^{01}	-1.0394×10^{01}	-1.0403×10^{01}	-1.0403×10^{01}	-1.0403×10^{01}	-1.0403×10^{01}	-1.0403×10^{01}	-1.0403×10^{01}	-1.0403×10^{01}
F23	Avg.	-8.6580×10^{00}	-7.2350×10^{00}	-9.8216×10^{00}	-1.0175×10^{01}	-1.0354×10^{01}	-8.7432×10^{00}	-1.0518×10^{01}	-1.0358×10^{01}	-1.0536×10^{01}
	STD.	2.5993×10^{00}	2.5826×10^{00}	1.8535×10^{00}	1.3719×10^{00}	9.8705×10^{-01}	2.5794×10^{00}	2.0234×10^{-02}	9.7874×10^{-01}	2.7028×10^{-05}
	Min	-1.0536×10^{01}	-1.0536×10^{01}	-1.0536×10^{01}	-1.0536×10^{01}	-1.0536×10^{01}	-1.0536×10^{01}	-1.0536×10^{01}	-1.0536×10^{01}	-1.0536×10^{01}

Table 7. The *p*-values of the null hypothesis *t*-test of CSA versus other algorithms.

	GWO	SCA	SSA	HHO	WOA	PSO	TSO	CGO
F1	1.7344×10^{-06}	1.7344×10^{-06}	1.7344×10^{-06}	6.0350×10^{-03}	1.7344×10^{-06}	1.7344×10^{-06}	1.7344×10^{-06}	1.9209×10^{-06}
F2	1.7344×10^{-06}	1.7344×10^{-06}	1.7344×10^{-06}	5.0383×10^{-01}	1.7344×10^{-06}	1.7344×10^{-06}	1.7344×10^{-06}	2.3534×10^{-06}
F3	1.7344×10^{-06}	1.7344×10^{-06}	1.7344×10^{-06}	1.7344×10^{-06}	1.7344×10^{-06}	1.7344×10^{-06}	1.7344×10^{-06}	3.1817×10^{-06}
F4	1.7344×10^{-06}	1.7344×10^{-06}	1.7344×10^{-06}	3.7243×10^{-05}	1.7344×10^{-06}	1.7344×10^{-06}	1.7344×10^{-06}	2.3534×10^{-06}
F5	1.7344×10^{-06}	1.7344×10^{-06}	1.7344×10^{-06}	1.5625×10^{-02}	1.7344×10^{-06}	1.7344×10^{-06}	1.7344×10^{-06}	1.7344×10^{-06}
F6	1.7344×10^{-06}	1.7344×10^{-06}	1.7344×10^{-06}	1.3183×10^{-04}	1.7344×10^{-06}	1.7344×10^{-06}	1.7344×10^{-06}	1.7344×10^{-06}
F7	5.3070×10^{-05}	6.5641×10^{-02}	1.7344×10^{-06}	1.7088×10^{-03}	3.3269×10^{-02}	1.7344×10^{-06}	5.7924×10^{-05}	1.3194×10^{-02}
F8	1.7344×10^{-06}	1.7344×10^{-06}	1.7344×10^{-06}	6.2500×10^{-02}	2.5631×10^{-06}	1.7344×10^{-06}	1.7344×10^{-06}	1.2383×10^{-06}
F9	2.5631×10^{-06}	2.5596×10^{-06}	1.7344×10^{-06}	1.0000×10^{00}	1.0000×10^{00}	1.7344×10^{-06}	1.7344×10^{-06}	1.0000×10^{00}
F10	1.4383×10^{-06}	1.7344×10^{-06}	1.7344×10^{-06}	1.0000×10^{00}	2.1912×10^{-05}	1.7344×10^{-06}	1.7344×10^{-06}	6.3342×10^{-05}

Table 7. *Cont.*

	GWO	SCA	SSA	HHO	WOA	PSO	TSO	CGO
F11	1.2500×10^{-01}	1.7344×10^{-06}	1.7344×10^{-06}	1.0000×10^{00}	1.0000×10^{00}	1.7344×10^{-06}	1.7344×10^{-06}	1.0000×10^{00}
F12	1.7344×10^{-06}	1.7344×10^{-06}	1.7344×10^{-06}	1.7344×10^{-06}	1.7344×10^{-06}	1.7344×10^{-06}	1.7344×10^{-06}	1.7344×10^{-06}
F13	1.7344×10^{-06}	1.7344×10^{-06}	1.7344×10^{-06}	2.7016×10^{-05}	1.7344×10^{-06}	1.7344×10^{-06}	1.7344×10^{-06}	1.7300×10^{-06}
F14	1.7344×10^{-06}	1.7344×10^{-06}	1.0231×10^{-05}	1.8435×10^{-04}	1.7344×10^{-06}	1.0000×10^{00}	1.7344×10^{-06}	5.0000×10^{-01}
F15	2.8434×10^{-05}	1.7344×10^{-06}	1.9209×10^{-06}	6.3391×10^{-06}	2.8786×10^{-06}	3.3173×10^{-04}	3.5152×10^{-06}	3.1123×10^{-05}
F16	3.8822×10^{-06}	1.7344×10^{-06}	1.7344×10^{-06}	8.9443×10^{-04}	4.1140×10^{-03}	1.7344×10^{-06}	2.1266×10^{-06}	1.7344×10^{-06}
F17	1.7344×10^{-06}	1.7344×10^{-06}	1.7344×10^{-06}	5.2165×10^{-06}	1.9209×10^{-06}	1.7344×10^{-06}	1.9209×10^{-06}	1.7344×10^{-06}
F18	3.0650×10^{-04}	1.2453×10^{-02}	1.7344×10^{-06}	2.1630×10^{-05}	1.1499×10^{-04}	1.7344×10^{-06}	1.7344×10^{-06}	1.7344×10^{-06}
F19	1.4936×10^{-05}	1.7344×10^{-06}	1.7344×10^{-06}	9.3157×10^{-06}	1.1265×10^{-05}	1.7344×10^{-06}	2.3534×10^{-06}	1.7344×10^{-06}
F20	1.5658×10^{-02}	1.7344×10^{-06}	5.7924×10^{-05}	1.7344×10^{-06}	1.1748×10^{-02}	3.0861×10^{-01}	1.4795×10^{-02}	7.3433×10^{-01}
F21	1.7344×10^{-06}	1.7344×10^{-06}	1.7344×10^{-06}	1.3591×10^{-01}	2.6033×10^{-06}	6.6858×10^{-01}	1.7344×10^{-06}	6.0496×10^{-07}
F22	1.7344×10^{-06}	1.7344×10^{-06}	3.1123×10^{-05}	1.7988×10^{-05}	9.3157×10^{-06}	3.8202×10^{-01}	2.1266×10^{-06}	1.7300×10^{-06}
F23	1.7344×10^{-06}	1.7344×10^{-06}	1.4795×10^{-02}	1.9729×10^{-05}	3.1123×10^{-05}	6.4352×10^{-01}	1.7344×10^{-06}	3.1123×10^{-05}

Table 8. Statistical results of the optimized standard functions with $d = 100$.

Function	Test	CSA	PSO	SSA	SCA	GWO
F1	Best	4.05207×10^{-66}	1.40909×10^{-02}	8.41001×10^{-02}	7.65220×10^{-05}	1.15671×10^{-16}
	Mean	9.49793×10^{-37}	3.23392×10^{01}	1.98479×10^{01}	4.22900×10^{-01}	2.72480×10^{-14}
	Std	5.20184×10^{-36}	4.83756×10^{01}	2.80619×10^{01}	9.12128×10^{-01}	3.45607×10^{-14}
F2	Best	1.79982×10^{-33}	1.51649×10^{-01}	3.28652×10^{-01}	9.66793×10^{-05}	4.82345×10^{-10}
	Mean	2.92106×10^{-22}	3.89166×10^{00}	4.75249×10^{00}	3.31250×10^{-03}	6.43809×10^{-09}
	Std	1.53426×10^{-21}	4.88194×10^{00}	3.71882×10^{00}	3.35433×10^{-03}	4.85863×10^{-09}
F3	Best	6.77086×10^{-66}	2.43120×10^{01}	2.58477×10^{02}	3.09196×10^{-04}	5.83050×10^{-01}
	Mean	1.18525×10^{-40}	8.05289×10^{04}	2.33034×10^{04}	4.24039×10^{00}	1.82227×10^{02}
	Std	6.39505×10^{-40}	3.84529×10^{04}	3.16829×10^{04}	6.62382×10^{00}	2.88485×10^{02}
F4	Best	5.64112×10^{-36}	1.97453×10^{-01}	1.96642×10^{-01}	1.97453×10^{-01}	1.97453×10^{-01}
	Mean	3.81723×10^{-22}	2.62001×10^{00}	9.88436×10^{-01}	2.62001×10^{00}	2.59439×10^{00}
	Std	1.83964×10^{-21}	2.72100×10^{00}	7.22522×10^{-01}	2.72100×10^{00}	2.71903×10^{00}
F5	Best	0.00000×10^{00}	8.09021×10^{-01}	1.15830×10^{00}	1.80961×10^{00}	1.80961×10^{00}
	Mean	0.00000×10^{00}	3.38351×10^{03}	4.75416×10^{02}	4.89317×10^{03}	1.26269×10^{02}
	Std	0.00000×10^{00}	1.14975×10^{04}	9.22158×10^{02}	1.39892×10^{04}	1.65257×10^{02}
F6	Best	0.00000×10^{00}	9.17634×10^{-05}	3.55921×10^{-04}	1.76154×10^{-03}	8.62047×10^{-04}
	Mean	0.00000×10^{00}	3.30019×10^{01}	1.32821×10^{01}	2.93302×10^{01}	7.17494×10^{00}
	Std	0.00000×10^{00}	5.13931×10^{01}	1.86246×10^{01}	2.88694×10^{01}	3.74740×10^{00}
F7	Best	4.23203×10^{-06}	1.54553×10^{-02}	4.83797×10^{-03}	3.57837×10^{-04}	3.66995×10^{-03}
	Mean	4.00859×10^{-04}	1.86225×10^{-01}	1.41949×10^{-01}	1.01286×10^{-01}	8.42239×10^{-03}
	Std	4.78167×10^{-04}	1.75616×10^{-01}	1.17454×10^{-01}	2.46964×10^{-01}	4.68636×10^{-03}
F8	Best	-4.18983×10^{04}	-4.18952×10^{04}	-4.18914×10^{04}	-4.18464×10^{04}	-4.18464×10^{04}
	Mean	-4.18983×10^{04}	-3.96372×10^{04}	-4.03440×10^{04}	-3.75470×10^{04}	-3.82785×10^{04}
	Std	2.96014×10^{-11}	4.49162×10^{03}	3.63037×10^{03}	5.82693×10^{03}	5.27419×10^{03}
F9	Best	0.00000×10^{00}	3.27575×10^{-03}	8.23333×10^{-03}	3.73030×10^{-05}	3.25244×10^{-06}
	Mean	0.00000×10^{00}	8.27521×10^{01}	4.28287×10^{01}	1.24652×10^{02}	1.51656×10^{02}
	Std	0.00000×10^{00}	7.32871×10^{01}	4.87230×10^{01}	1.19664×10^{02}	1.11244×10^{02}
F10	Best	8.88178×10^{-16}	7.84248×10^{-02}	2.24259×10^{-02}	8.35089×10^{-04}	1.59905×10^{-09}
	Mean	8.88178×10^{-16}	2.88503×10^{00}	1.76283×10^{00}	1.31387×10^{00}	9.57690×10^{-01}
	Std	0.00000×10^{00}	2.62189×10^{00}	1.12465×10^{00}	2.18567×10^{00}	2.49990×10^{00}
F11	Best	0.00000×10^{00}	4.25764×10^{-03}	5.71780×10^{-02}	3.41024×10^{-06}	0.00000×10^{00}
	Mean	0.00000×10^{00}	1.52847×10^{00}	9.20741×10^{-01}	2.27770×10^{-01}	1.55201×10^{-03}
	Std	0.00000×10^{00}	1.66071×10^{00}	5.63135×10^{-01}	2.79365×10^{-01}	5.96126×10^{-03}
F12	Best	4.71163×10^{-33}	1.97180×10^{-04}	1.12509×10^{-03}	5.24202×10^{-03}	1.23122×10^{-04}
	Mean	4.71163×10^{-33}	1.01808×10^{00}	4.60484×10^{-01}	1.64094×10^{00}	1.44908×10^{00}
	Std	1.39185×10^{-48}	1.49538×10^{00}	1.11625×10^{00}	1.78266×10^{00}	2.82336×10^{00}
F13	Best	1.34978×10^{-32}	3.52700×10^{-05}	2.58636×10^{-04}	7.25952×10^{-04}	4.19563×10^{-04}
	Mean	1.34978×10^{-32}	1.45863×10^{01}	2.90656×10^{00}	2.47516×10^{01}	2.72461×10^{00}
	Std	5.56740×10^{-48}	2.21891×10^{01}	5.27444×10^{00}	4.78364×10^{01}	2.82642×10^{00}

In addition, the null hypothesis *t*-test with 5% significance level was used to validate the significance level of the results of the CSA versus other algorithms. The significance level revealed that there was a significant comparison between the achieved results by the CSA and the results of the comparative algorithms. This can be measured by comparing the results of the CSA for 30 independent runs with the results of other algorithms, e.g., CSA vs. GWO. If the *p*-value of the *t*-test was less than 0.05, then there was a significant

comparison between the CSA and the other algorithms. However, if the *p*-value was higher than 0.05, then there was no significant comparison between the CSA and the other algorithms. Table 7 shows the obtained *p*-values of the CSA versus other algorithms for the 30 independent runs. Most of the *p*-values were less than 0.05, except 8 values out of 92 values.

4.4. High-Dimensional Functions

To investigate the CSA's behavior with high-dimensional problems in further detail, the CSA was tested for benchmark functions (F1–F13) with 100 and 1000 dimensions. The statistical results of all algorithms are shown in Table 8, including the best minimum, mean, and standard deviation. It is obvious that the CSA was resilient and consistently produced the best results for all high-dimensional functions. Additionally, the standard deviation demonstrated the CSA's durability and stability in 30 independent experiments. On the other hand, the compared algorithms were not resilient against high-dimensional algorithms with a large standard deviation. Additionally, the 1000 dimensions ensured the suggested CSA's robustness. The results of all algorithms and functions are summarized in Table 9. It is obvious that the CSA is resilient and produces the best minimal results, with a very low standard deviation in comparison with other algorithms.

Table 9. Statistical results of the optimized standard functions with $d = 1000$.

Function	Test	CSA	PSO	SSA	SCA	GWO
F1	Best	1.74114×10^{-68}	7.86330×10^{-01}	8.69625×10^{-01}	9.85563×10^{-01}	1.28824×10^{-03}
	Mean	6.75805×10^{-43}	2.57658×10^{04}	1.87823×10^{03}	3.13124×10^{04}	6.86880×10^{01}
	Std	3.69123×10^{-42}	5.49528×10^{04}	4.45580×10^{03}	6.67314×10^{04}	1.47024×10^{02}
F2	Best	4.94485×10^{-35}	1.70351×10^{00}	1.86340×10^{00}	2.11466×10^{00}	7.00593×10^{-04}
	Mean	2.44060×10^{-21}	2.85661×10^{02}	9.02285×10^{01}	4.69947×10^{01}	4.62960×10^{-02}
	Std	1.33304×10^{-20}	2.42764×10^{02}	7.96943×10^{01}	2.46857×10^{01}	3.27695×10^{-02}
F3	Best	7.28940×10^{-62}	1.44335×10^{06}	7.08473×10^{03}	3.52592×10^{04}	3.55999×10^{05}
	Mean	4.45675×10^{-39}	9.83821×10^{06}	5.17443×10^{06}	8.89991×10^{05}	1.07487×10^{06}
	Std	2.13852×10^{-38}	3.95888×10^{06}	4.61038×10^{06}	6.68057×10^{05}	3.87543×10^{05}
F4	Best	9.39244×10^{-41}	2.82903×10^{-02}	2.82903×10^{-02}	2.82903×10^{-02}	2.82903×10^{-02}
	Mean	4.97072×10^{-23}	3.13296×10^{00}	1.47575×10^{00}	3.13296×10^{00}	3.01621×10^{00}
	Std	2.35097×10^{-22}	2.36133×10^{00}	1.34765×10^{00}	2.36133×10^{00}	2.28900×10^{00}
F5	Best	0.00000×10^{00}	1.11144×10^{02}	1.10805×10^{02}	1.13580×10^{02}	1.13580×10^{02}
	Mean	0.00000×10^{00}	3.96596×10^{06}	1.47830×10^{05}	4.84064×10^{06}	1.08327×10^{06}
	Std	0.00000×10^{00}	1.44863×10^{07}	7.61203×10^{05}	1.75361×10^{07}	4.23453×10^{06}
F6	Best	0.00000×10^{00}	1.11552×10^{00}	1.18127×10^{00}	1.37640×10^{00}	1.16202×10^{00}
	Mean	0.00000×10^{00}	9.06200×10^{03}	1.26919×10^{03}	1.13358×10^{04}	1.43531×10^{02}
	Std	0.00000×10^{00}	1.17632×10^{04}	2.22354×10^{03}	1.46871×10^{04}	1.19839×10^{02}
F7	Best	1.34248×10^{-05}	2.95314×10^{-02}	3.36464×10^{-03}	8.28122×10^{-03}	5.86957×10^{-03}
	Mean	2.40186×10^{-04}	3.51020×10^{01}	7.03426×10^{-01}	3.98208×10^{01}	8.03855×10^{00}
	Std	2.09618×10^{-04}	1.65495×10^{02}	1.69178×10^{00}	1.89489×10^{02}	2.83190×10^{01}
F8	Best	-4.18983×10^{05}	-4.18978×10^{05}	-4.18978×10^{05}	-4.18977×10^{05}	-4.18977×10^{05}
	Mean	-4.18983×10^{05}	-3.87824×10^{05}	-3.88408×10^{05}	-3.84621×10^{05}	-3.84898×10^{05}
	Std	1.18405×10^{-10}	4.42563×10^{04}	4.34900×10^{04}	4.69810×10^{04}	4.68726×10^{04}
F9	Best	0.00000×10^{00}	4.15359×10^{-01}	4.65994×10^{-01}	5.27444×10^{-01}	5.27444×10^{-01}
	Mean	0.00000×10^{00}	1.34989×10^{03}	4.44170×10^{02}	1.47652×10^{03}	1.47580×10^{03}
	Std	0.00000×10^{00}	1.02977×10^{03}	4.18444×10^{02}	1.10934×10^{03}	1.10745×10^{03}

Table 9. *Cont.*

Function	Test	CSA	PSO	SSA	SCA	GWO
F10	Best	8.88178×10^{-16}	1.17690×10^{-01}	1.21822×10^{-01}	1.30832×10^{-01}	9.76292×10^{-04}
	Mean	8.88178×10^{-16}	4.07750×10^{00}	2.10393×10^{00}	4.24648×10^{00}	2.82129×10^{00}
	Std	0.00000×10^{00}	2.78347×10^{00}	1.56515×10^{00}	2.84110×10^{00}	3.32683×10^{00}
F11	Best	0.00000×10^{00}	1.54060×10^{00}	5.76886×10^{-01}	1.65931×10^{00}	4.45692×10^{-03}
	Mean	0.00000×10^{00}	1.40518×10^{02}	1.45067×10^{01}	1.74547×10^{02}	7.56728×10^{-01}
	Std	0.00000×10^{00}	1.93782×10^{02}	3.49974×10^{01}	2.45555×10^{02}	6.70865×10^{-01}
F12	Best	4.71163×10^{-34}	3.49707×10^{-05}	2.95159×10^{-05}	3.53593×10^{-05}	3.22334×10^{-05}
	Mean	4.71163×10^{-34}	4.61458×10^{00}	1.15674×10^{00}	4.71509×10^{00}	2.37655×10^{00}
	Std	8.69906×10^{-50}	7.79023×10^{00}	2.17639×10^{00}	8.25036×10^{00}	3.71613×10^{00}
F13	Best	1.34978×10^{-32}	1.24017×10^{00}	1.07560×10^{00}	1.37886×10^{00}	1.05795×10^{00}
	Mean	1.34978×10^{-32}	2.66755×10^{02}	2.86286×10^{01}	2.83343×10^{02}	2.02468×10^{02}
	Std	5.56740×10^{-48}	4.21627×10^{02}	2.54245×10^{01}	4.39341×10^{02}	3.96828×10^{02}

4.5. Computational Time

In some applications, the computational speed and simplicity of the optimization algorithm are critical. As a result, the CPU time for the proposed CSA and other algorithms is measured during the optimization of high-dimensional functions. As shown in Table 10, the CSA outperformed the PSO and required less CPU time than all other algorithms for all functions. However, GWO and SSA required higher CPU time. Additionally, Table 11 compares the CPU times of each algorithm, revealing that the CSA ran very fast and almost had half the CPU time of the GWO. The results have proven that the CSA is more efficient and faster than the compared algorithms.

Table 10. CPU time for high-dimensional functions at $d = 100$.

	CSA	PSO	SSA	SCA	GWO
F1	0.178432	0.190542	0.302927	0.20064	0.282661
F2	0.12315	0.159538	0.228658	0.195575	0.239618
F3	0.568033	0.611411	0.721536	0.730838	0.698834
F4	0.12295	0.188939	0.239374	0.196382	0.251754
F5	0.156947	0.170969	0.235264	0.209225	0.246743
F6	0.115594	0.169406	0.215389	0.193195	0.227015
F7	0.275001	0.305304	0.393652	0.364141	0.403618
F8	0.14028	0.223961	0.268633	0.246461	0.294057
F9	0.122207	0.160593	0.262984	0.209103	0.27208
F10	0.133938	0.175045	0.255505	0.214336	0.264772
F11	0.13927	0.173938	0.255895	0.230421	0.256377
F12	0.526553	0.582185	0.667134	0.6323	0.678474
F13	0.512411	0.559926	0.652665	0.616762	0.688092
Sum	3.114766	3.671757	4.699616	4.239379	4.804095
Rank	(1)	(2)	(4)	(3)	(5)

Table 11. CPU time for high-dimensional functions at $d = 1000$.

	CSA	PSO	SSA	SCA	GWO
F1	0.864606	0.920696	1.411428	1.526312	1.954178
F2	0.8211	0.927257	1.691217	1.690701	2.304578
F3	9.746041	9.914311	9.757791	9.416562	9.929068
F4	0.712832	0.857485	1.391238	1.644103	2.219137

Table 11. *Cont.*

	CSA	PSO	SSA	SCA	GWO
F5	0.784051	0.858395	1.423493	1.550448	2.002662
F6	0.764518	0.883734	1.460421	1.610766	1.972805
F7	1.450708	1.453512	2.155236	2.387385	2.742038
F8	0.970483	1.17946	1.778078	2.079184	2.718625
F9	0.904283	1.135896	1.676209	1.895193	2.421654
F10	0.918339	1.132918	1.680022	1.861007	2.290501
F11	1.024792	1.171338	1.809926	2.044052	2.410413
F12	2.495781	2.802061	3.292436	3.421013	4.016006
F13	2.471004	2.700387	3.175498	3.381912	4.266181
Sum	23.92854	25.93745	32.70299	34.50864	41.24785

4.6. Convergence Speed

For finding a fair comparison between the convergence speed of the CSA and the convergence speed of other algorithms, the same initial populations were used for all algorithms. Figure 4 shows that all convergence curves started from the same point, where the convergence curve (red line) of the proposed CSA converged to the minimum value faster than other algorithms. This fast convergence speed of the CSA was due to the behavior of the Tan function with the variations of the angle.

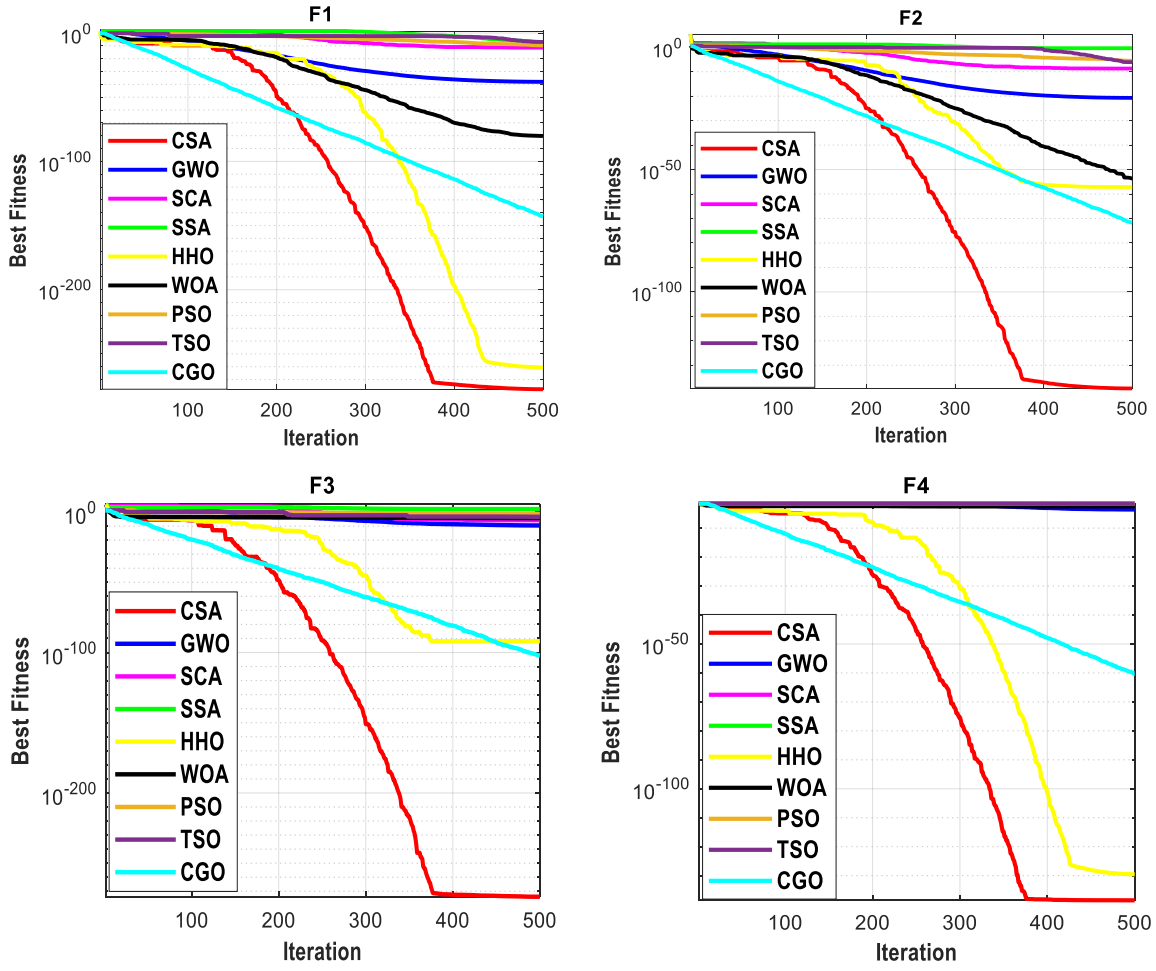


Figure 4. *Cont.*

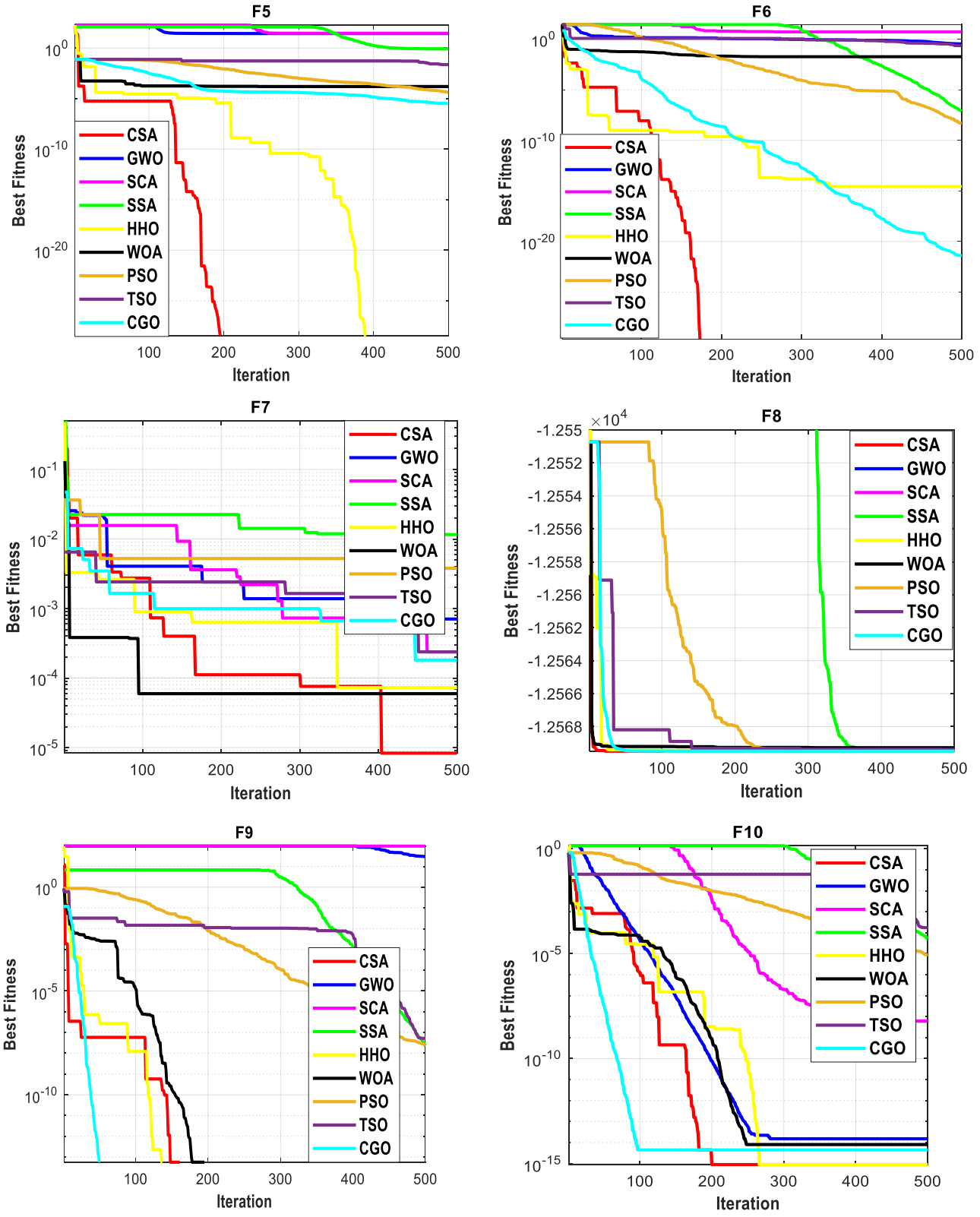


Figure 4. *Cont.*

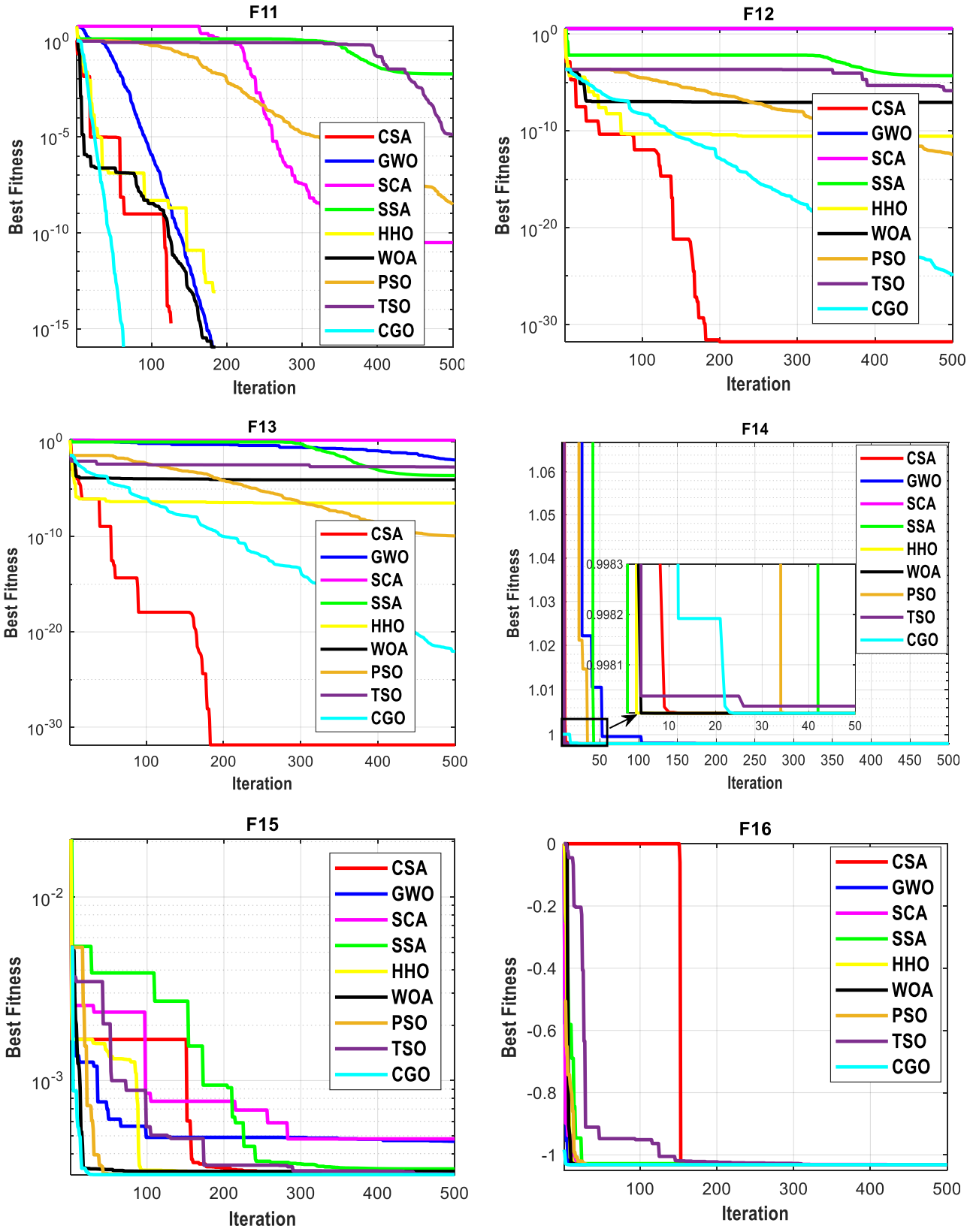


Figure 4. *Cont.*

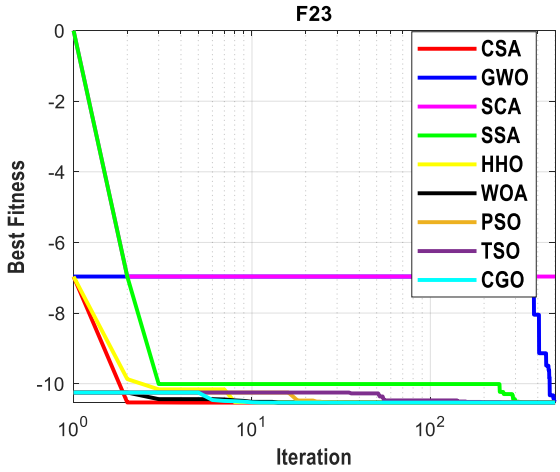


Figure 4. Convergence curves of the applied algorithms.

Further statistical analysis was conducted using a box plot to show the minimum, maximum, and median of the results of 30 independent experiments. Figure 5 exhibits the statistical analysis for four selected functions (F5, F12, F16, and F22), where the deviation between the minimum and maximum for the proposed CSA was very small for all functions. However, other algorithms exhibited wide deviation between the minimum and maximum values for 30 independent runs.

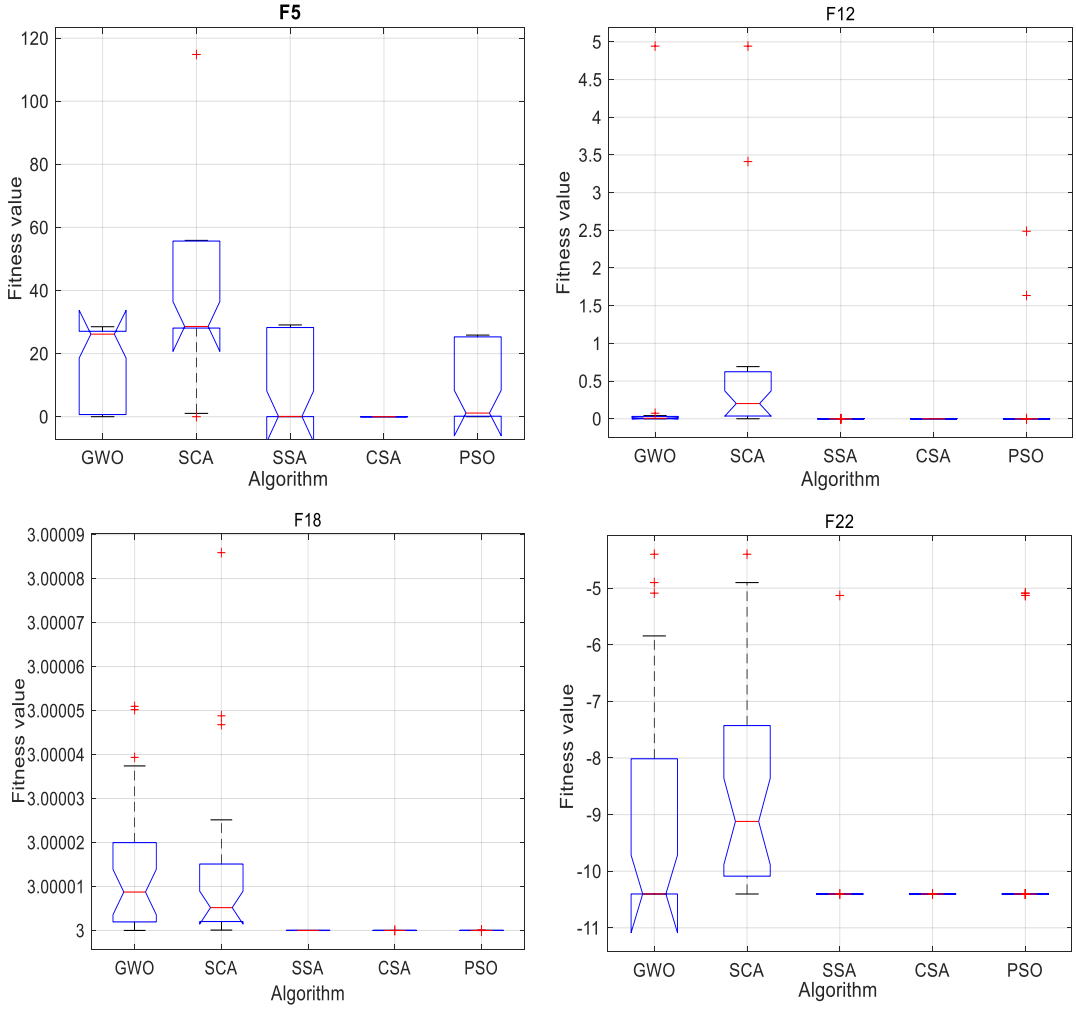


Figure 5. Statistical results using box-plot for 30 independent runs.

5. Real-World Engineering Problems

The principal goal of this section is to evaluate the performance of the proposed CSA using constrained engineering problems, including welded beam, pressure vessel, and tension spring designs. The CSA, PSO, SSA, SCA, and GWO algorithms were applied to obtain an optimal design of these problems. The number of the population of all algorithms was 100, and the iteration number was 10,000. The statistical results were obtained for 30 independent runs.

5.1. Welded Beam

The SCA was used to optimally design the famous welded beam design, which is shown in Figure 6. The objective is to reduce the manufacturing cost of the welded beam by finding its optimum design factors. The optimal design is subjected to many restrictions, such as shear stress (τ), bending stress (σ), buckling load (P_c), deflection (δ), and other restrictions. The formulations of objective function and its constraints are stated in Equation (9). The design factors $x = [x_1, x_2, x_3, x_4] = [h, l, t, b]$ refer to thickness of weld, length, height, and thickness of the bar, respectively. Table 12 lists the optimal design factors that were achieved by the CSA and other algorithms. It is obvious that the CSA obtained the lowest cost compared with PSO, SSA, SCA, and GWO algorithms. Moreover, the standard deviation of 30 runs' results of the CSA was low, and thus the CSA was stable at all 30 runs. Therefore, the CSA was considered robust enough to be used in real-world engineering problems.

$$\begin{aligned} &x = [x_1\ x_2\ x_3\ x_4] \Rightarrow \min f(x) = 1.10471{x_1}^2 x_2 + 0.04811 x_3 x_4 (14.0 + x_2) \\ &g_1(x) = \tau - \tau_{\max} \le 0 \text{ and } g_2(x) = \sigma - \sigma_{\max} \le 0 \\ &g_3(x) = \delta - \delta_{\max} \le 0 \text{ and } g_4(x) = x_1 - x_4 \le 0 \\ &g_5(x) = P - P_c \le 0 \text{ and } g_6(x) = 0.125 - x_1 \le 0 \\ &g_7(x) = 0.10471{x_1}^2 + 0.04811 x_3 x_4 (14.0 + x_2) - 5.0 \le 0 \end{aligned} \tag{10}$$

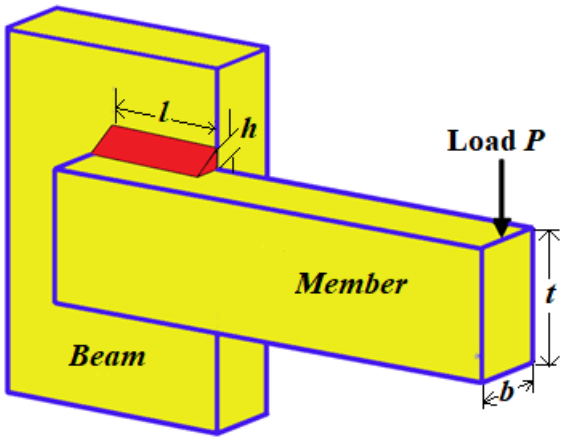


Figure 6. Welded beam design.

The range of variables is

$$\begin{aligned} &0.1 \le x_1 \le 2 \\ &0.1 \le x_2 \le 10 \\ &0.1 \le x_3 \le 10 \\ &0.1 \le x_4 \le 2 \end{aligned}$$

where

$$\begin{aligned} &P = 6000\ lb;\ L = 14\ in;\ E = 30 \times 10^6\ psi;\ G = 12 \times 10^6\ psi \\ &\tau_{\max} = 13600\ psi;\ \sigma_{\max} = 30000\ psi;\ \delta_{\max} = 0.25\ in \\ &M = P(L + \tfrac{x_2}{2});\ R = \sqrt{\tfrac{{x_2}^2}{4} + (\tfrac{x_1 + x_3}{2})^2};\ \tau' = \tfrac{P}{\sqrt{2} x_1 x_2};\ \tau'' = \tfrac{MR}{J} \end{aligned}$$

$$J = 2\sqrt{2}x_1x_2(\frac{{x_2}^2}{12} + (\frac{x_1+x_3}{2})^2);\ \tau = \sqrt{\tau'^2 + \tau'\tau''\frac{x_2}{R} + \tau''^2}$$
$$P_c = 4.013\frac{E}{L^2}\sqrt{\frac{x_3^2x_4^6}{36}}(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}})$$
$$\sigma = \frac{6PL}{x_4{x_3}^2};\ \delta = \frac{4PL^3}{Ex_4{x_3}^2}$$

Table 12. Optimal design of welded beam.

	CSA	PSO	SSA	SCA	GWO
h	0.205729639786	0.205729639786	0.205723211955	0.205811043402	0.205724311092
l	3.470488665628	7.092414276557	7.092727008708	7.380674589109	7.092527090825
t	9.036623910358	9.036623910358	9.036624222889	8.972002667140	9.036803373437
b	0.205729639786	0.205729639786	0.205729638416	0.208874244972	0.205728938896
Minimum cost	1.724852308597	2.218150861764	2.218172785211	2.273030395508	2.218180086668
Average cost	1.724853828957	2.218150861764	2.244245629001	2.291353653772	2.218198907121
Std.	0.000004807757	0.000000000000	0.052784172626	0.010496471904	0.000013807448

5.2. Pressure Vessel

In this application, the target was to design the cheapest and optimal formation of a pressure vessel, as depicted in Figure 7. The formulation of cost function and its constraints are shown in Equation (10). The design factors of the vessel were the thicknesses of the shell and head (T_s and T_h) and the inner radius and length of the vessel (R and L) without bounce. Table 13 lists the optimal design factors of the pressure vessel that were obtained by the CSA and other algorithms, including PSO, SSA, SCA, and GWO algorithms. It was obvious that the CSA was competing with other algorithms to find the best minimum; however, the GWO algorithm obtained better mean and standard deviations for 30 independent runs.

$$\begin{aligned} &x = [x_1\ x_2\ x_3\ x_4] = [T_s, T_h, R, L] \\ &\min f(x) = 0.6224x_1x_3x_4 + 1.7781x_2{x_3}^2 + 3.1661{x_1}^2x_4 + 19.84{x_1}^2x_3 \\ &g_1(x) = -x_1 + 0.0193x_3 \le 0 \\ &g_2(x) = -x_2 + 0.00954x_3 \le 0 \\ &g_3(x) = -\pi{x_3}^2x_4 - \tfrac{4}{3}\pi{x_3}^3 + 1296000 \le 0 \\ &g_4(x) = x_4 - 240 \le 0 \end{aligned} \tag{11}$$

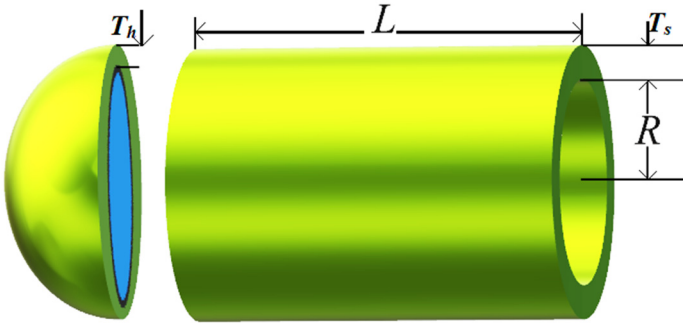


Figure 7. Cylindrical vessel design.

Table 13. Optimal design of pressure vessel.

	CSA	PSO	SSA	SCA	GWO
T_s	0.77816864138	0.7781686414	0.79357920102	0.78922547613	0.007781787
T_h	0.38464916263	0.3846491626	0.39226677976	0.40639993523	0.003846530
R	40.3196187241	40.3196187241	41.1180752663	40.6926147876	40.319922618
L	200.000000000	200.000000000	189.175939539	196.033601579	200.000000000
Minimum cost	5885.33277362	5885.33277362	5912.20652171	6004.52071673	5885.46666087
Average cost	6011.55334154	6013.40373404	6191.42556961	6198.38074830	5974.52840595
Std	175.417988776	179.129462647	307.601967961	116.552624682	79.439547307

The range of variables

$$0 \le x_1 \le 99$$
$$0 \le x_2 \le 99$$
$$10 \le x_3 \le 200$$
$$10 \le x_4 \le 200$$

5.3. Tension Spring

The objective here was to optimally design the tension spring with the lowest weight, as displayed in Figure 8. The mathematical formulations of the weight function and the restriction functions are displayed in Equation (11). The main four restriction functions were deflection, shear stress, surge wave frequency, and outer diameter. The optimal design factors were the mean coil diameter (D), wire diameter (d), and the active coils' number (N). Table 14 lists the optimal design factors of the tension spring coil that were obtained by the CSA and other algorithms, including PSO, SSA, SCA, and GWO algorithms. It was obvious that the proposed CSA was competitive in obtaining the best minimum weight of the tension spring and low average of the results of 30 runs.

$$\begin{gathered} x = [x_1\ x_2\ x_3] = [D, d, N] \Rightarrow \min f(x) = {x_1}^2 x_2 (x_3 + 2) \\ g_1(x) = 1 - \frac{{x_2}^3 x_3}{71785 {x_1}^4} \le 0 \text{ and } g_2(x) = \frac{4{x_2}^2 - x_1 x_2}{12566 {x_1}^3 (x_2 - x_1)} + \frac{1}{5108 {x_1}^2} - 1 \le 0 \\ g_3(x) = 1 - \frac{140.45 x_1}{x_3 {x_2}^2} \le 0 \text{ and } g_4(x) = \frac{x_1 + x_2}{1.5} - 1 \le 0 \end{gathered} \tag{12}$$

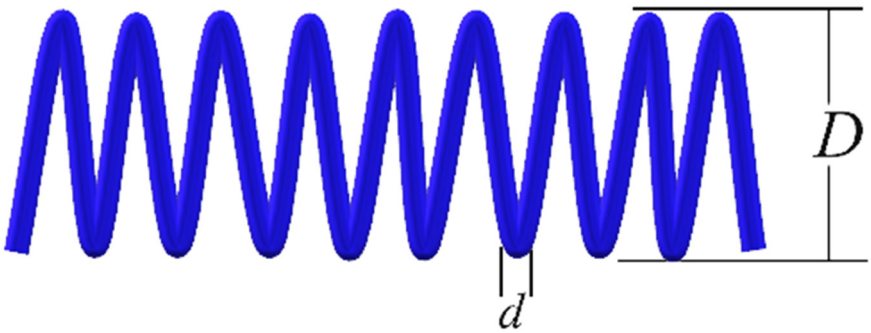


Figure 8. Tension coil spring.

Table 14. Optimal design of tension coil spring.

	CSA	PSO	SSA	SCA	GWO
D	0.0517190259	0.0516975399	0.0500000000	0.0500000000	0.0517410542
d	0.3574390430	0.3569217527	0.3174254133	0.3155229746	0.3579696634
N	11.2468029380	11.2770151263	14.0277750624	14.4243340035	11.2159545387
Min. weight	0.0126652492	0.0126652341	0.0127190578	0.0129556368	0.0126652949
Avg. weight	0.0126789335	0.0133988758	0.0127190585	0.0131845009	0.0126662267
Std	0.0000327544	0.0015508356	0.0000000011	0.0001295549	0.0000011609

The range of variables

$$0.05 \leq x_1 \leq 2 \;\&\; 0.25 \leq x_2 \leq 1.3$$
$$2.00 \leq x_3 \leq 15$$

6. Conclusions and Future Work

This article introduced a novel geometry-based metaheuristic algorithm dubbed the circle search algorithm (CSA). This approach is inspired by the circle's extraordinary features, particularly the relationship between the tangent line and the orthogonal radius, which is represented by the tan function of the opposing angle to the radius. The CSA exploration is represented by numerous random circles that search for the optimal answer in a variety of directions. The CSA is exploited by decreasing the radius and opposing angle of the tangent point until it reaches the center point. The CSA effectiveness was demonstrated using 23 standard renowned functions and compared with the performance of other well-known algorithms, including the PSO, SSA, SCA, and GWO algorithms. The unimodal functions demonstrated that the CSA is more capable of deepening exploitation than other methods. The multimodal functions validated the CSA capacity to explore and escape from local optima. Convergence curves were used to visually inspect the CSA convergence rate. Furthermore, the proposed algorithm was tested with high-dimensional problems, and its results outperformed the other compared algorithms. Moreover, the computational time of the CSA had the shortest value among the algorithms. Numerous statistical tests were conducted to determine the suggested CSA superiority and significance in comparison with other algorithms. The mean, standard deviation, rank test, and t-test for null hypothesis were used to examine the 30 independent runs' results. These statistical tests established the suggested CSA resilience in comparison with other algorithms, where more than 90% of p-values were less than 0.05. To provide additional validation, the CSA method was applied to three well-known, real-world engineering problems, namely, welded beam, pressure vessel, and tension spring. In conclusion, researchers are encouraged by the acquired results to apply the CSA to solve a variety of real-world engineering optimization problems. In future works, the CSA will be applied to renewable energy modeling, power system operation and control, control applications, microgrids, and smart grid applications.

Author Contributions: Conceptualization, M.H.Q.; Data curation, S.A.; Formal analysis, M.H.Q., H.M.H., R.A.T., S.A., M.T.-V. and F.J.; Investigation, H.M.H., R.A.T. and M.T.-V.; Methodology, M.H.Q.; Project administration, S.A.; Resources, F.J.; Software, M.H.Q.; Supervision, S.A. and F.J.; Visualization, H.M.H., R.A.T. and M.T.-V.; Writing—original draft, M.H.Q.; Writing—review & editing, H.M.H., M.T.-V. and F.J. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Researchers Supporting Project number (RSP-2021/307), King Saud University, Riyadh, Saudi Arabia.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: This work was supported by the Researchers Supporting Project number (RSP-2021/307), King Saud University, Riyadh, Saudi Arabia.

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

References

1. Wang, P.C.; Shoup, T.E. Parameter sensitivity study of the Nelder-Mead Simplex Method. *Adv. Eng. Softw.* **2011**, *42*, 529–533. [CrossRef]
2. Altinoz, O.T.; Yilmaz, A.E. Multiobjective Hooke–Jeeves algorithm with a stochastic Newton–Raphson-like step-size method. *Expert Syst. Appl.* **2019**, *117*, 166–175. [CrossRef]
3. Leardi, R. Genetic Algorithms. *Compr. Chemom.* **2009**, *1*, 631–653.
4. Clerc, M. *Particle Swarm Optimization*; ISTE: London, UK, 2006; Volume 4, ISBN 9780470612163.
5. Qais, M.; Abdulwahid, Z. A new method for improving particle swarm optimization algorithm (TriPSO). In Proceedings of the 2013 5th International Conference on Modeling, Simulation and Applied Optimization, Hammamet, Tunisia, 28–30 April 2013.
6. Storn, R.; Price, K. Differential Evolution—A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
7. Karaboga, D.; Basturk, B. Artificial Bee Colony (ABC) optimization algorithm for solving constrained optimization problems. In Proceedings of the Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Cancun, Mexico, 18–21 June 2007; Melin, P., Castillo, O., Aguilar, L.T., Kacprzyk, J., Pedrycz, W., Eds.; Springer: Berlin/Heidelberg, Germany, 2007; Volume 4529, pp. 789–798.
8. Ji, J.; Song, S.; Tang, C.; Gao, S.; Tang, Z.; Todo, Y. An artificial bee colony algorithm search guided by scale-free networks. *Inf. Sci.* **2019**, *473*, 142–165. [CrossRef]
9. Dorigo, M.; Blum, C. Ant colony optimization theory: A survey. *Theor. Comput. Sci.* **2005**, *344*, 243–278. [CrossRef]
10. Yang, X.S. A new metaheuristic Bat-inspired Algorithm. In *Studies in Computational Intelligence*; González, J.R., Pelta, D.A., Cruz, C., Terrazas, G., Krasnogor, N., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; Volume 284, pp. 65–74. ISBN 9783642125379.
11. Cheng, M.Y.; Prayogo, D. Symbiotic Organisms Search: A new metaheuristic optimization algorithm. *Comput. Struct.* **2014**, *139*, 98–112. [CrossRef]
12. Qi, X.; Zhu, Y.; Zhang, H. A new meta-heuristic butterfly-inspired algorithm. *J. Comput. Sci.* **2017**, *23*, 226–239. [CrossRef]
13. Mirjalili, S.; Gandomi, A.H.; Mirjalili, S.Z.; Saremi, S.; Faris, H.; Mirjalili, S.M. Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. *Adv. Eng. Softw.* **2017**, *114*, 163–191. [CrossRef]
14. Abdollahzadeh, B.; Soleimanian Gharehchopogh, F.; Mirjalili, S. Artificial gorilla troops optimizer: A new nature-inspired metaheuristic algorithm for global optimization problems. *Int. J. Intell. Syst.* **2021**, *36*, 5887–5958. [CrossRef]
15. Saremi, S.; Mirjalili, S.; Lewis, A. Grasshopper Optimisation Algorithm: Theory and application. *Adv. Eng. Softw.* **2017**, *105*, 30–47. [CrossRef]
16. Mirjalili, S. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowl. Based Syst.* **2015**, *89*, 228–249. [CrossRef]
17. Oftadeh, R.; Mahjoob, M.J.; Shariatpanahi, M. A novel meta-heuristic optimization algorithm inspired by group hunting of animals: Hunting search. *Comput. Math. Appl.* **2010**, *60*, 2087–2098. [CrossRef]
18. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
19. Mirjalili, S.; Lewis, A. The Whale Optimization Algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
20. Heidari, A.A.; Mirjalili, S.; Faris, H.; Aljarah, I.; Mafarja, M.; Chen, H. Harris hawks optimization: Algorithm and applications. *Futur. Gener. Comput. Syst.* **2019**, *97*, 849–872. [CrossRef]
21. Pierezan, J.; Dos Santos Coelho, L. Coyote Optimization Algorithm: A New Metaheuristic for Global Optimization Problems. In Proceedings of the 2018 IEEE Congress on Evolutionary Computation, Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–8.
22. Faramarzi, A.; Heidarinejad, M.; Mirjalili, S.; Gandomi, A.H. Marine Predators Algorithm: A nature-inspired metaheuristic. *Expert Syst. Appl.* **2020**, *152*, 113377. [CrossRef]
23. Kaveh, A.; Farhoudi, N. A new optimization method: Dolphin echolocation. *Adv. Eng. Softw.* **2013**, *59*, 53–70. [CrossRef]
24. Yu, J.J.Q.; Li, V.O.K. A social spider algorithm for global optimization. *Appl. Soft Comput. J.* **2015**, *30*, 614–627. [CrossRef]
25. Eusuff, M.; Lansey, K.; Pasha, F. Shuffled frog-leaping algorithm: A memetic meta-heuristic for discrete optimization. *Eng. Optim.* **2006**, *38*, 129–154. [CrossRef]
26. Hashim, F.A.; Houssein, E.H.; Hussain, K.; Mabrouk, M.S.; Al-Atabany, W. Honey Badger Algorithm: New metaheuristic algorithm for solving optimization problems. *Math. Comput. Simul.* **2022**, *192*, 84–110. [CrossRef]
27. Gandomi, A.H.; Yang, X.S.; Alavi, A.H. Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems. *Eng. Comput.* **2013**, *29*, 17–35. [CrossRef]
28. Askarzadeh, A. Bird mating optimizer: An optimization algorithm inspired by bird mating strategies. *Commun. Nonlinear Sci. Numer. Simul.* **2014**, *19*, 1213–1228. [CrossRef]
29. Kaur, S.; Awasthi, L.K.; Sangal, A.L.; Dhiman, G. Tunicate Swarm Algorithm: A new bio-inspired based metaheuristic paradigm for global optimization. *Eng. Appl. Artif. Intell.* **2020**, *90*, 103541. [CrossRef]

30. Qais, M.H.; Hasanien, H.M.; Alghuwainem, S.; Elgendy, M.A. Output Power Smoothing of Grid-Tied PMSG-Based Variable Speed Wind Turbine Using Optimal Controlled SMES. In Proceedings of the 2019 54th International Universities Power Engineering Conference, Bucharest, Romania, 3–6 September 2019; pp. 1–6.
31. Kallioras, N.A.; Lagaros, N.D.; Avtzis, D.N. Pity beetle algorithm—A new metaheuristic inspired by the behavior of bark beetles. *Adv. Eng. Softw.* **2018**, *121*, 147–166. [CrossRef]
32. Dhiman, G.; Kumar, V. Spotted hyena optimizer: A novel bio-inspired based metaheuristic technique for engineering applications. *Adv. Eng. Softw.* **2017**, *114*, 48–70. [CrossRef]
33. Mohammadi-Balani, A.; Dehghan Nayeri, M.; Azar, A.; Taghizadeh-Yazdi, M. Golden eagle optimizer: A nature-inspired metaheuristic algorithm. *Comput. Ind. Eng.* **2021**, *152*, 107050. [CrossRef]
34. Gomes, G.F.; da Cunha, S.S.; Ancelotti, A.C. A sunflower optimization (SFO) algorithm applied to damage identification on laminated composite plates. *Eng. Comput.* **2019**, *35*, 619–626. [CrossRef]
35. Kiran, M.S. TSA: Tree-seed algorithm for continuous optimization. *Expert Syst. Appl.* **2015**, *42*, 6686–6698. [CrossRef]
36. Yang, X.S. Flower pollination algorithm for global optimization. In Proceedings of the Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Orléan, France, 3–7 September 2012; Springer: Berlin/Heidelberg, Germany; Volume 7445, pp. 240–249.
37. Eskandar, H.; Sadollah, A.; Bahreininejad, A.; Hamdi, M. Water cycle algorithm—A novel metaheuristic optimization method for solving constrained engineering optimization problems. *Comput. Struct.* **2012**, *110–111*, 151–166. [CrossRef]
38. Kaveh, A.; Bakhshpoori, T. Water Evaporation Optimization: A novel physically inspired optimization algorithm. *Comput. Struct.* **2016**, *167*, 69–85. [CrossRef]
39. Patel, V.K.; Savsani, V.J. Heat transfer search (HTS): A novel optimization algorithm. *Inf. Sci.* **2015**, *324*, 217–246. [CrossRef]
40. Shayanfar, H.; Gharehchopogh, F.S. Farmland fertility: A new metaheuristic algorithm for solving continuous optimization problems. *Appl. Soft Comput. J.* **2018**, *71*, 728–746. [CrossRef]
41. Kaveh, A.; Dadras Eslamlou, A. Water strider algorithm: A new metaheuristic and applications. *Structures* **2020**, *25*, 520–541. [CrossRef]
42. Moazzeni, A.R.; Khamehchi, E. Rain optimization algorithm (ROA): A new metaheuristic method for drilling optimization solutions. *J. Pet. Sci. Eng.* **2020**, *195*, 107512. [CrossRef]
43. Qais, M.H.; Hasanien, H.M.; Alghuwainem, S. Output power smoothing of grid-connected permanent-magnet synchronous generator driven directly by variable speed wind turbine: A review. *J. Eng.* **2017**, *2017*, 1755–1759. [CrossRef]
44. Rashedi, E.; Nezamabadi-pour, H.; Saryazdi, S. GSA: A Gravitational Search Algorithm. *Inf. Sci.* **2009**, *179*, 2232–2248. [CrossRef]
45. Wang, Y.; Yu, Y.; Gao, S.; Pan, H.; Yang, G. A hierarchical gravitational search algorithm with an effective gravitational constant. *Swarm Evol. Comput.* **2019**, *46*, 118–139. [CrossRef]
46. Pereira, J.L.J.; Francisco, M.B.; Diniz, C.A.; Antônio Oliver, G.; Cunha, S.S.; Gomes, G.F. Lichtenberg algorithm: A novel hybrid physics-based meta-heuristic for global optimization. *Expert Syst. Appl.* **2021**, *170*, 114522. [CrossRef]
47. Hashim, F.A.; Houssein, E.H.; Mabrouk, M.S.; Al-Atabany, W.; Mirjalili, S. Henry gas solubility optimization: A novel physics-based algorithm. *Futur. Gener. Comput. Syst.* **2019**, *101*, 646–667. [CrossRef]
48. Qais, M.; Khaled, U.; Alghuwainem, S. Improved differential relay for bus bar protection scheme with saturated current transformers based on second order harmonics. *J. King Saud Univ. Eng. Sci.* **2018**, *30*, 320–329. [CrossRef]
49. Qais, M.; Khaled, U. Evaluation of V–t characteristics caused by lightning strokes at different locations along transmission lines. *J. King Saud Univ. Eng. Sci.* **2018**, *30*, 150–160. [CrossRef]
50. Kaveh, A.; Dadras, A. A novel meta-heuristic optimization algorithm: Thermal exchange optimization. *Adv. Eng. Softw.* **2017**, *110*, 69–84. [CrossRef]
51. Kaveh, A.; Mahdavi, V.R. Colliding bodies optimization: A novel meta-heuristic method. *Comput. Struct.* **2014**, *139*, 18–27. [CrossRef]
52. Zhao, W.; Wang, L.; Zhang, Z. Atom search optimization and its application to solve a hydrogeologic parameter estimation problem. *Knowl. Based Syst.* **2019**, *163*, 283–304. [CrossRef]
53. Kaveh, A.; Talatahari, S. A novel heuristic optimization method: Charged system search. *Acta Mech.* **2010**, *213*, 267–289. [CrossRef]
54. Kaveh, A.; Khayatazad, M. A new meta-heuristic method: Ray Optimization. *Comput. Struct.* **2012**, *112–113*, 283–294. [CrossRef]
55. Qais, M.H.; Hasanien, H.M.; Alghuwainem, S. Transient search optimization: A new meta-heuristic optimization algorithm. *Appl. Intell.* **2020**, *50*, 3926–3941. [CrossRef]
56. Muthiah-Nakarajan, V.; Noel, M.M. Galactic Swarm Optimization: A new global optimization metaheuristic inspired by galactic motion. *Appl. Soft Comput. J.* **2016**, *38*, 771–787. [CrossRef]
57. Formato, R.A. Central force optimization: A new nature inspired computational framework for multidimensional search and optimization. *Stud. Comput. Intell.* **2008**, *129*, 221–238.
58. Javidy, B.; Hatamlou, A.; Mirjalili, S. Ions motion algorithm for solving optimization problems. *Appl. Soft Comput. J.* **2015**, *32*, 72–79. [CrossRef]
59. Gholizadeh, S.; Danesh, M.; Gheyratmand, C. A new Newton metaheuristic algorithm for discrete performance-based design optimization of steel moment frames. *Comput. Struct.* **2020**, *234*, 106250. [CrossRef]
60. Ahmadianfar, I.; Bozorg-Haddad, O.; Chu, X. Gradient-based optimizer: A new metaheuristic optimization algorithm. *Inf. Sci.* **2020**, *540*, 131–159. [CrossRef]

61. Azizi, M. Atomic orbital search: A novel metaheuristic algorithm. *Appl. Math. Model.* **2021**, *93*, 657–683. [CrossRef]
62. Talatahari, S.; Azizi, M. Chaos Game Optimization: A novel metaheuristic algorithm. *Artif. Intell. Rev.* **2021**, *54*, 917–1004. [CrossRef]
63. van Laarhoven, P.J.M.; Aarts, E.H.L. (Eds.) *Simulated Annealing: Theory and Applications*; Springer: Dordrecht, The Netherlands, 1987; pp. 7–15 ISBN 978-94-015-7744-1.
64. Hasanien, H.M.; Muyeen, S.M. Particle swarm optimization-based superconducting magnetic energy storage for low-voltage ride-through capability enhancement in wind energy conversion system. *Electr. Power Compon. Syst.* **2015**, *43*, 1278–1288. [CrossRef]
65. Qais, M.H.; Hasanien, H.M.; Alghuwainem, S. Enhanced whale optimization algorithm for maximum power point tracking of variable-speed wind generators. *Appl. Soft Comput. J.* **2020**, *86*, 105937. [CrossRef]
66. Qais, M.H.; Hasanien, H.M.; Alghuwainem, S. Augmented grey wolf optimizer for grid-connected PMSG-based wind energy conversion systems. *Appl. Soft Comput. J.* **2018**, *69*, 504–515. [CrossRef]
67. Qais, M.; Hasanien, H.M.; Alghuwainem, S. Salp swarm algorithm-based TS-FLCs for MPPT and fault ride-through capability enhancement of wind generators. *ISA Trans.* **2020**, *101*, 211–224. [CrossRef]
68. Qais, M.H.; Hasanien, H.M.; Alghuwainem, S. A Grey Wolf Optimizer for Optimum Parameters of Multiple PI Controllers of a Grid-Connected PMSG Driven by Variable Speed Wind Turbine. *IEEE Access* **2018**, *6*, 44120–44128. [CrossRef]
69. Qais, M.H.; Hasanien, H.M.; Alghuwainem, S. Whale optimization algorithm-based Sugeno fuzzy logic controller for fault ride-through improvement of grid-connected variable speed wind generators. *Eng. Appl. Artif. Intell.* **2020**, *87*, 103328. [CrossRef]
70. Qais, M.H.; Hasanien, H.M.; Alghuwainem, S. Identification of electrical parameters for three-diode photovoltaic model using analytical and sunflower optimization algorithm. *Appl. Energy* **2019**, *250*, 109–117. [CrossRef]
71. Qais, M.H.; Hasanien, H.M.; Alghuwainem, S.; Nouh, A.S. Coyote optimization algorithm for parameters extraction of three-diode photovoltaic models of photovoltaic modules. *Energy* **2019**, *187*, 116001. [CrossRef]
72. Qais, M.H.; Hasanien, H.M.; Alghuwainem, S. Parameters extraction of three-diode photovoltaic model using computation and Harris Hawks optimization. *Energy* **2020**, *195*, 117040. [CrossRef]
73. Qais, M.H.; Hasanien, H.M.; Alghuwainem, S. Transient search optimization for electrical parameters estimation of photovoltaic module based on datasheet values. *Energy Convers. Manag.* **2020**, *214*, 112904. [CrossRef]
74. Qais, M.H.; Hasanien, H.M.; Alghuwainem, S. Optimal transient search algorithm-based PI controllers for enhancing low voltage ride-through ability of grid-linked PMSG-based wind turbine. *Electronics* **2020**, *9*, 1807. [CrossRef]
75. Wolpert, D.H.; Macready, W.G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [CrossRef]
76. Jiang, Y.; Wu, Q.; Zhu, S.; Zhang, L. Orca predation algorithm: A novel bio-inspired algorithm for global optimization problems. *Expert Syst. Appl.* **2022**, *188*, 116026. [CrossRef]
77. Suyanto, S.; Ariyanto, A.A.; Ariyanto, A.F. Komodo Mlipir Algorithm. *Appl. Soft Comput.* **2022**, *114*, 108043. [CrossRef]
78. Li, C.; Chen, G.; Liang, G.; Luo, F.; Zhao, J.; Dong, Z.Y. Integrated optimization algorithm: A metaheuristic approach for complicated optimization. *Inf. Sci.* **2022**, *586*, 424–449. [CrossRef]
79. Abualigah, L.; Elaziz, M.A.; Sumari, P.; Geem, Z.W.; Gandomi, A.H. Reptile Search Algorithm (RSA): A nature-inspired meta-heuristic optimizer. *Expert Syst. Appl.* **2022**, *191*, 116158. [CrossRef]
80. 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]
81. Jafari, M.; Salajegheh, E.; Salajegheh, J. Elephant clan optimization: A nature-inspired metaheuristic algorithm for the optimal design of structures. *Appl. Soft Comput.* **2021**, *113*, 107892. [CrossRef]
82. Feng, Z.; Niu, W.; Liu, S. Cooperation search algorithm: A novel metaheuristic evolutionary intelligence algorithm for numerical optimization and engineering optimization problems. *Appl. Soft Comput.* **2021**, *98*, 106734. [CrossRef]
83. Zhang, Y.; Jin, Z. Group teaching optimization algorithm: A novel metaheuristic method for solving global optimization problems. *Expert Syst. Appl.* **2020**, *148*, 113246. [CrossRef]
84. Zervoudakis, K.; Tsafarakis, S. A mayfly optimization algorithm. *Comput. Ind. Eng.* **2020**, *145*, 106559. [CrossRef]
85. Shabani, A.; Asgarian, B.; Salido, M.; Asil Gharebaghi, S. Search and rescue optimization algorithm: A new optimization method for solving constrained engineering optimization problems. *Expert Syst. Appl.* **2020**, *161*, 113698. [CrossRef]
86. Mirjalili, S. SCA: A Sine Cosine Algorithm for solving optimization problems. *Knowl. Based Syst.* **2016**, *96*, 120–133. [CrossRef]