applied sciences

Article

Boosted Reptile Search Algorithm for Engineering and Optimization Problems

Mohamed Abd Elaziz [1,2,3,4,5,*], Samia Chelloug [6,*], Mai Alduailij [7,*] and Mohammed A. A. Al-qaness [8]

1 Department of Mathematics, Faculty of Science, Zagazig University, Zagazig 44519, Egypt
2 Artificial Intelligence Research Center (AIRC), Ajman University, Ajman 346, United Arab Emirates
3 Department of Artificial Intelligence Science and Engineering, Galala University, Suze 435611, Egypt
4 Department of Electrical and Computer Engineering, Lebanese American University, Byblos 13-5053, Lebanon
5 Faculty of Information Technology, Middle East University, Amman 11831, Jordan
6 Department of Information Technology, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University, Riyadh 11671, Saudi Arabia
7 Department of Computer Science, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University, Riyadh 11671, Saudi Arabia
8 College of Physics and Electronic Information Engineering, Zhejiang Normal University, Jinhua 321004, China
* Correspondence: abd_el_aziz_m@yahoo.com (M.A.E.); sachelloug@pnu.edu.sa (S.C.); maalduailij@pnu.edu.sa (M.A.)

Abstract: Recently, various metaheuristic (MH) optimization algorithms have been presented and applied to solve complex engineering and optimization problems. One main category of MH algorithms is the naturally inspired swarm intelligence (SI) algorithms. SI methods have shown great performance on different problems. However, individual MH and SI methods face some shortcomings, such as trapping at local optima. To solve this issue, hybrid SI methods can perform better than individual ones. In this study, we developed a boosted version of the reptile search algorithm (RSA) to be employed for different complex problems, such as intrusion detection systems (IDSs) in cloud–IoT environments, as well as different optimization and engineering problems. This modification was performed by employing the operators of the red fox algorithm (RFO) and triangular mutation operator (TMO). The aim of using the RFO was to boost the exploration of the RSA, whereas the TMO was used for enhancing the exploitation stage of the RSA. To assess the developed approach, called RSRFT, a set of six constrained engineering benchmarks was used. The experimental results illustrated the ability of RSRFT to find the solution to those tested engineering problems. In addition, it outperformed the other well-known optimization techniques that have been used to handle these problems.

Keywords: intrusion detection systems; reptile search algorithm; red fox algorithm; triangular mutation operator; engineering problems

check for updates

Citation: Abd Elaziz, M.; Chelloug, S.; Alduailij, M.; Al-qaness, M.A.A. Boosted Reptile Search Algorithm for Engineering and Optimization Problems. *Appl. Sci.* **2023**, *13*, 3206. https://doi.org/10.3390/app13053206

Academic Editors: Dimitris Mourtzis and Jose Machado

Received: 30 January 2023
Revised: 21 February 2023
Accepted: 26 February 2023
Published: 2 March 2023

1. Introduction

In daily life, optimization is everywhere. Optimization is used widely in many fields, including system management, technical design, economics, and various engineering problems [1,2]. To optimize something means to make sure one or more goals of a particular situation are as ideal as possible. The goal of the optimization process is to investigate potential solutions and find the optimal one. Occasionally, many constraints are applied to solutions, making optimization more difficult [3].

In recent decades, the effectiveness of metaheuristic (MH) techniques in solving complicated problems with high dimensionality, multimodality, and non-differentiability has been proven by different studies in different applications [4–6]. As a result, the use of these algorithms has increased significantly, and there is a growing trend to suggest efficient upgrades and new metaheuristic algorithms. These algorithms primarily gain their inspiration from natural phenomena, physics laws, animals' and birds' behaviors, and so on. Earlier, some efficient optimization algorithms were developed, such as the genetic algorithm (GA) [7], artificial bee colony (ABC) [8], particle swarm optimization (PSO) [9], and the firefly algorithm [10]. Recently, there have been many newly developed metaheuristic optimization algorithms based on different inspirations, for example the sine–cosine algorithm (SCA) [11], the multi-verse optimizer (MVO) [12], Harris hawks optimization (HHO) [13], the marine predator algorithm [14], the Aquila optimizer [15], the arithmetic optimization algorithm [16], the Reptile search algorithm (RSA) [17], and many other optimization techniques.

Although those algorithms have shown good performance in different optimization and engineering algorithms, in some cases, they face critical limitations and shortcomings. According to the well-known no free lunch theorem, no one method has the ability of solving all problems. To this end, in recent years, researchers have used the hybridization concept to develop hybrid metaheuristic optimization algorithms to address different engineering and optimization issues.

For example, Houssein et al. [3] applied an improved equilibrium optimizer (EO) using a technique called the self-adaptive EO algorithm. They used four search techniques to enhance the search process of the EO optimization method. This method was employed to address different optimization engineering problems, and it showed better results compared to the original EO. Bo et al. [18] applied several search techniques, namely opposition-based learning and greedy search, to boost the chimp optimization algorithm (ChOA) search mechanism. The developed ChOA was employed to solve different constrained engineering problems and recorded significant outcomes. Shen et al. [19] introduced an enhanced whale optimization algorithm (WOA) to handle engineering design problems and global optimization issues. They used a technique called multi-population evolution to improve the original WOA. A modified SCA was developed by Yang et al. [20]. They suggested a multi-mechanism acting SCA method to overcome the drawbacks of the original SCA. This modified version of the SCA was utilized to address constrained engineering optimization problems. An improved SSA optimization technique was developed by Hong et al. [21] to solve engineering problems, using a new strategy called producer centralization. Furthermore, in [22], a modified improved moth–flame optimizer was suggested using two search mechanisms, namely Gaussian mutation and chaotic grouping. This modified method was evaluated with complex engineering and optimization problems. An enhanced FA was suggested by Peng et al. [23] to solve complex engineering optimization problems. They applied three search techniques to enhance the traditional FA. Zhang et al. [24] presented a modified AOA to be employed in various numerical and engineering optimization applications based on two techniques, called random high-speed jumping and multi-leader wandering around search strategy. Hybrid optimization algorithms have also been adopted in other research areas, such as a hybrid of AO and the seagull optimization algorithm for wind power forecasting [25] and a hybrid of AO with the SMA and SSO to predict CO_2 trapping in deep saline [26,27]. Another hybrid optimization method was developed by [28] using DE and the Cuckoo Search algorithm to optimize the ship pipe route. Moreover, a hybrid method of HHO and DE was suggested by Zhong et al. to solve the flight trajectory prediction problem [29].

In recent years, MH optimization techniques also have been investigated to address the problems of intrusion detection systems. Many optimization algorithms have been suggested as feature selection methods to improve the quality solutions of the IDS, such as PSO [30], the GA [31], AO [32], HHO [33], and many other approaches [1,2].

Paper Contribution

Inspired by the successful applications of the hybrid concept to overcome the drawbacks of the individual optimization algorithms, in this study, we present a new version of the reptile search algorithm using two techniques, namely the red fox optimization algorithm and the mutation operators. The RSA [17] is a recently developed optimization method emulating crocodiles' hunting behaviors in nature. It has been adopted in various applications since it has high performance, such as in [34–36]. The red fox optimization algorithm [37] is also a nature-inspired technique based on the hunting behavior of the red fox. It has also been adopted in solving some optimization problems [38–40].

The main modification was performed using RFO to enhance the ability of the RSA to explore the search domain. Additionally, the triangular mutation operator (TMO) was applied to improve the exploitation of the RSA. This competition between the operators of the RSA, TMO, and RFO led to enhancing the rate of convergence toward the optimal solution.

In short, we can represent the contributions of this study as the following points:

1. We propose a boosted version of the reptile search algorithm (RSA), called RSRFT, to address IDS problems in IoT and cloud environments, as well as complex and multidimensional engineering problems
2. We employed the operators of the red fox optimization and triangular mutation operator to boost the performance of the RSA.
3. We applied the RSRFT technique to solve different and complex engineering problems. We conducted a set of comparisons with other efficient techniques to verify the quality of RSRFT.

The rest of this study is organizes as follows: In Section 2, detailed background descriptions of the applied algorithms are presented. Section 3 introduces the descriptions of the RSRFT approach. In Section 4, we assess the quality of the modified RSA method using different datasets and with extensive comparisons to other methods, and in Section 5, we present the conclusion of this paper.

2. Background

2.1. The Reptile Search Algorithm

The conventional RSA, which simulates actual crocodile behavior, is presented in this section. This was accomplished using two steps: the local and global search stages.

2.1.1. Exploration Search

Following [17], the process of splitting the maximum number of generations into four parts will trigger the RSA to switch from the exploration to the exploitation stage. Moreover, the RSA exploration phase investigates the search domains using two primary search techniques in pursuit of a more reliable agent. There are two requirements for this search component. If the long walking plan is not adjusted by $t \leq \frac{T}{4}$, the agents are modified by $t \leq 2\frac{T}{4}$ and $t > \frac{T}{4}$. Equation (1) can be utilized to update the positions for the exploration stage:

$$x_{ij}(t+1) = \begin{cases} Best_j(t) \times -\eta_{ij}(t) \times \beta - R_{ij}(t) \times rand, & t \leq \frac{T}{4} \\ Best_j(t) \times x_{(r_1,j)} \times ES(t) \times rand, & t \leq 2\frac{T}{4} \; and \; t > \frac{T}{4} \end{cases} \tag{1}$$

where $Best_j$ stands for the best solution at dimension j. $x_{(r_1,j)}$ denotes the value of a randomly selected solution at dimension j. $rand$ represents a random value, and $\beta = 0.1$ as in [17]. T represents the total number of generations. N and η_{ij} are the number of solutions and the hunting operator. The definition of η_{ij} is given as

$$\eta_{ij} = Best_j(t) \times P_{ij}, \tag{2}$$

$$R_{ij} = \frac{Best_j(t) - x_{(r_2,j)}}{Best_j(t) + \epsilon}, \tag{3}$$

$$ES(t) = 2 \times r_3 \times \left(1 - \frac{1}{T}\right), \tag{4}$$

In Equation (4), ϵ refers to a small value. $r_3 \in [-1,1]$ is a random integer value, while P_{ij} is defined as

$$P_{ij} = \alpha + \frac{x_{ij} - M(x_i)}{Best_j(t) \times (UB_{(j)} - LB_{(j)}) + \epsilon}, \tag{5}$$

In Equation (5), $UB_{(j)}$ and $LB_{(j)}$ represent the boundaries of the search domain, and $\alpha = 0.1$ [17]. $M(x_i)$ denotes the mean value of X, and it is given as

$$M(x_i) = \frac{1}{n}\sum_{j=1}^{n} x_{ij}, \tag{6}$$

2.1.2. Exploitation Search

Using the hunting coordination plan, $t \le 3\frac{T}{4}$ and $t > 2\frac{T}{4}$ adopt the agents in this stage. In contrast, if $t \le T$ and $t > 3\frac{T}{4}$, the hunting cooperation plan is carried out. Equation (7) provides the updated value using the exploitation process.

$$x_{ij}(t+1) = \begin{cases} Best_j(t) \times P_{ij}(t) \times rand, & t \le 3\frac{T}{4} \; and \; t > 2\frac{T}{4} \\ Best_j(t) - \eta_{ij}(t) \times \epsilon - R_{ij}(t) \times rand, & t \le T \; and \; t > 3\frac{T}{4} \end{cases} \tag{7}$$

2.2. Red Fox Algorithm

This optimizer begins by figuring out the parameters' values and then creating the population of N foxes according to Equation (8):

$$X_{ij} = LB_j + r \times \left(UB_j - LB_j\right),\ i = 1,\ldots,N,\ j = 1,\ldots,D, \quad r \in [0,1] \tag{8}$$

where LB_j and UB_j indicate the boundaries of the search space, while D represents the dimension of X_{ij} foxes. Thereafter, the objective value of X_i is evaluated then allocating the best solution X_b. The next process is to update X_{ir} using the following formula:

$$X_{ir} = X_i + asign(X_b - X_i) \tag{9}$$

In Equation (9), $a \in [0, d_{ib}]$ stands for a scaling hyperparameter chosen randomly, whereas d_{ib} is defined as

$$d_{ib} = \sqrt{||X_i - X_b||} \tag{10}$$

In case X_{ir} has a fitness value better than its previous value, then we replace X_i with X_{ir}. Otherwise, we preserved the X_i value. The observation radius r is then adjusted by applying the formula Equation (11) in the case of the fox not being seen:

$$r = \begin{cases} a\frac{\sin(\phi_0)}{\phi_0} & if\ \phi_0 \ne 0 \\ \theta & otherwise \end{cases} \tag{11}$$

In Equation (11), $\theta \in [0,1]$ is a random value that is applied to control adverse weather conditions such as fog, rain, etc. After that, X is updated based on the following formula:

$$\begin{cases} x_0^{new} = ar.\cos(\phi_1) + x_0^{act} \\ x_1^{new} = ar.\sin(\phi_1) + ar.\cos(\phi_2) + x_1^{act} \\ x_2^{new} = ar.\sin(\phi_1) + ar.\sin(\phi_2) + ar.\cos(\phi_3) + x_2^{act} \\ \vdots \\ x_{n-1}^{new} = ar.\sin(\phi_1) + ar.\sin(\phi_2) + \cdots + ar.\sin(\phi_{n-1}) + x_{n-1}^{act} \end{cases} \tag{12}$$

where $\phi_i \in [0, 2\pi], i = 1, 2, \ldots, n-1$ and each angular value represents a randomized one for each point.

This set of equations simulates a fox's actions after it spots a target and tries to attack it. Then, the fitness value of X_i is assessed and then sorted based on the values. Hunters may kill the worst solution, while Equation (13) is employed to update the X solution.

$$X=\begin{cases} \text{nomadic agent} & \text{if} > 0.5 \\ \text{Reproduction of the alpha couple} & \text{otherwise} \end{cases} \tag{13}$$

where in the first branch of Equation (13); the new solutions wander outside the environment in search of a new space to repopulate their herd as nomadic agents. Outside of the habitat and within the search zone, the solution is chosen randomly. The center of the habitat (C_H) is calculated as

$$C_H=\frac{X_b+X_\beta}{2} \tag{14}$$

In Equation (14), X_b and X_β denote the first- and second-best solutions, respectively.

In addition, in the second case of Equation (13), the following formula is used to update X.

$$X=\frac{X_b+X_\beta}{2} \tag{15}$$

2.3. Triangular Mutation Operator

The mutation technique utilizes a integration vector, which is constructed using three randomly selected vectors along with three distinct vectors that have been selected during the competition (the best, worst, and better). This combination vector is then utilized to create the new mutant vector.

$$V_i(t+1) = M_1 \times (X_b(t) - \overline{X}_{br}(t)) + M_2 \times (X_b(t) - X_w(t)) + M_3 \times (X_{br}(t) - X_w(t)) + \overline{X}_c(t) \tag{16}$$

In Equation (16), M_1, M_2, and M_3 refer to the mutation factors that are related to x_i and they are produced according to the uniform distribution. In addition, three tournaments are given as $X_w(t)$, $X_{br}(t)$, $X_b(t)$. In addition, $\overline{X}_c(t)$ is the combination vector triangle at iteration t and formulated as:

$$\overline{X}_c = w_1 \times X_b + w_2 \times X_{br} + w_3 \times X_w,\ w_i \geqslant 0,\ \sum_{i=1}^{3} w_i = 1 \tag{17}$$

where w_i is a real weight and w_i is formulated as $w_i = p_i / \sum_{i=1}^{3} p_i$.

Moreover, the mutation is used to balance the exploitation propensity with the exploration capacity, with the primary mutation favoring the exploration phase. As a result, utilizing the presented mutation is twice as likely to succeed as using the fundamental principles. The differential evolution (DE) method is the foundation of the advanced mutation technique. Thus, using the fundamental mutation approach DE/rand/1/bin, the following new combination is created:

If $rand \leq (2/3))$, then

$$V_i(t+1) = M_1 \times (X_b(t) - X_{br}(t)) + M_2 \times (X_b(t) - X_w(t)) + M_3 \times (X_{br}(t) - X_w(t)) + \overline{X}_c(t) \tag{18}$$

Else:

$$V_i(t+1) = X_{r_1}(t) + F \times (X_{r_1}(t) - X_{r_3}(t)) \tag{19}$$

In Equation (18), F is a random value at $[(-1,0) \cup (0,1)]$ and $rand \in [0,1]$ a random value.

3. Proposed RSRFT Method

Within this section, we introduce the stages of the developed technique, which depends on modifying the performance of the RSA using TMO and RFO as given in Figure 1. The main objective of RFO is to facilitate the identification of a feasible solution within the feasible area, while TMO is applied to promote harmony between the exploitation and exploration. The initial population of RSRFT is constructed inside the search space's confines. The best solution is then selected from the current population after the fitness function for each solution has been evaluated. The solutions are updated using the RSA, RFO, and TMO operators. The RFO technique will then be used to update a collection of individuals that have reached a local standstill. Until the terminal condition is satisfied, the updating of the population is conducted again. The next subsections offer a detailed explanation of the developed RSRFT approach.

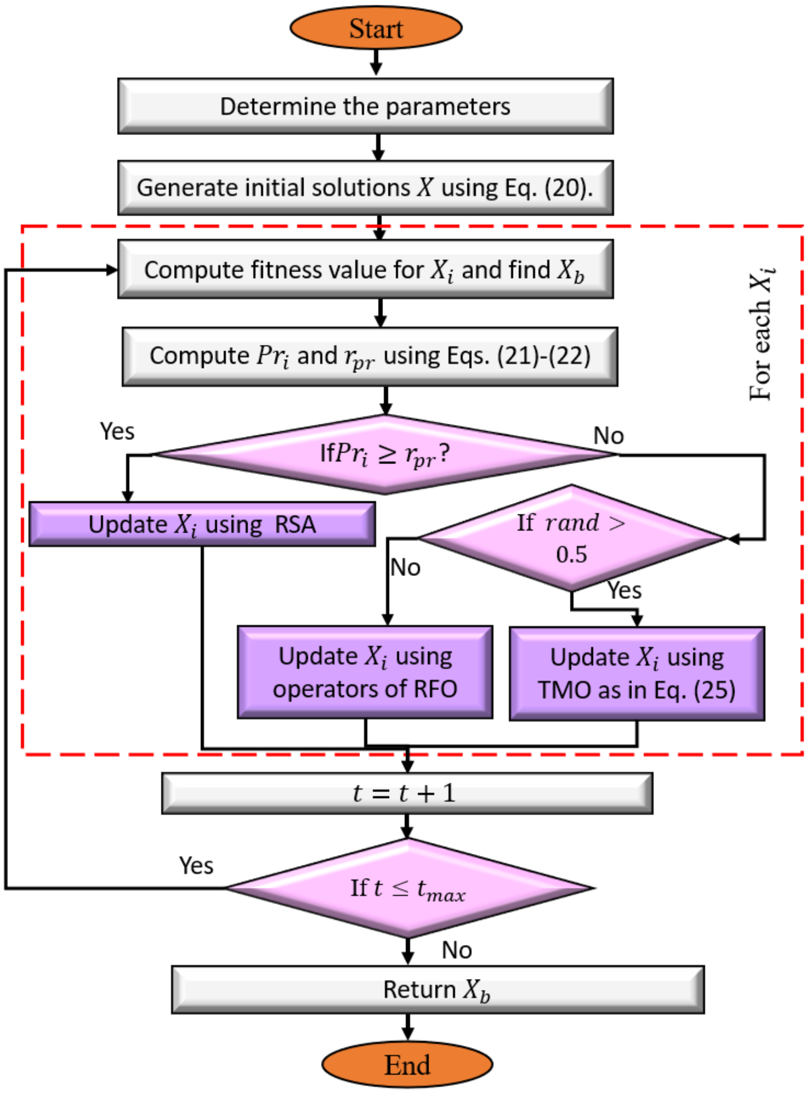
Start
Determine the parameters
Generate initial solutions X using Eq. (20).
Compute fitness value for X_i and find X_b
Compute Pr_i and r_{pr} using Eqs. (21)-(22)
For each X_i
Yes
No
If$Pr_i \geq r_{pr}$?
Update X_i using RSA
If $rand >$ 0.5
No
Yes
Update X_i using operators of RFO
Update X_i using TMO as in Eq. (25)
$t = t + 1$
Yes
If $t \leq t_{max}$
No
Return X_b
End

Figure 1. Developed RSRFT approach.

3.1. Initial Phase

The RSRFT creates the N agent starting populations (X_i) as

$$X_i = rand(1, D) \times (UB - LB) + LB, i = 1, \ldots, N \tag{20}$$

In Equation (20), $rand \in [0, 1]$ stands for a random vector with dimension D. N is the number of solutions.

3.2. Updating Phase

The current X is updated by using either the operators of the RSA, RFO, or TMO. This can be conducted through a set of steps. The first step is to calculate the fitness value for X_i.

The smallest fitness value and its related solution are then identified. Then, according to the probability of the fitness value, either the operators of the RSA or RFO and TMO are used to update X. This probability is computed as

$$Pr_i = \frac{Fit_i}{\sum_{i=1}^{N} Fit_i} \tag{21}$$

Then, the updating is performed using the following formula:

$$X_i(t+1) = \begin{cases} X_i^{RSA} & if\ Pr_i > r_{pr} \\ X_i^{RT} & Otherwise \end{cases}, \tag{22}$$

where r_{pr} is computed as

$$r_{pr} = min(Pr) + (max(Pr) - min(Pr)) \times rand \tag{23}$$

In addition, the value of X_i^{RT} is computed as

$$X_i(t+1) = \begin{cases} X_i^{TMO} & if\ rand > 0.5 \\ X_i^{RFO} & Otherwise \end{cases}, \tag{24}$$

$$X_i^{TMO} = \begin{cases} Use\ Equation\ (18)\ to\ update\ X_i & if\ rand > 0.5 \\ Use\ Equation\ (19)\ to\ update\ X_i & Otherwise \end{cases}, \tag{25}$$

while the values of X_i^{RFO} and X_i^{RSA} refer to the updated value of X_i using the operators of RFO and the RSA, respectively.

3.3. Terminal Phase

The stopping conditionsare examined during this phase, and in case they are satisfied, the steps of updating are halted and X_b is returned. Otherwise, a new updating step is carried out.

The main steps of the proposed RSRFT are given in Algorithm 1.

Algorithm 1 The RSRFT method.

```
1:  Initialize the parameters as $N$ solutions.
2:  Construct initial set of $X$ solutions using Equation (20).
3:  repeat
4:      Evaluate the fitness value of $X_i$, and find $X_b$.
5:      Compute $Pr_i$ and $r_{pr}$ using Equations (21) and (23), respectively.
6:      for $i = 1 : N$ do
7:          if $Pr_i > r_{pr}$ then
8:              Use the RSA to update $X_i$ as in Equations (1)–(7).
9:          else
10:             if $rand > 0.5$ then Update $X_i$ using the operators of RFO as in Equations (8)–(15).
11:             else
12:                 Use TMO to update $X_i$ as in Equation (25).
13:             end if
14:         end if
15:     end for
16:     $t = t + 1$.
17: until Stopping conditions met
18: Return best solution $X_b$.
```

4. Experimental Results and Discussion

In this study, the performance of RSRFT was evaluated through two experimental series, which included engineering problems and improve the security in the IoT environment. The experiments illustrated in this section were conducted using MATLAB R2020b installed on machine configured with a 2.40 GHz Intel Core i5 CPU, 4.00 GB RAM, and a Windows 10 operating system.

4.1. Series of Analysis 1: Engineering Problems

The evaluation of the developed RSRFT to solve the constraint optimization engineering problem is presented in this section. Six engineering optimization problems—the design of welded beams, the design of tension/compression springs, and the design optimization challenge for pressure vessels—were the focus of this experiment. The next subsections address the problem definition and results.

4.1.1. Welded Beam Design Problem

Determining the welding beam's settings that lower the fabrication costs are the major goal of solving welded beam design (WBD) problem (Figure 2). These parameters are the bar's height (t), the length of an attached portion (l), the weld's thickness (h), and the bar's thickness (b).

The definition of WBD is formulated as

$$
\begin{aligned}
\textit{Consider}\quad & \vec{u} = [u_1\ u_2\ u_3\ u_4] = [h\ l\ t\ b],\\
\textit{Minimize}\quad & f(\vec{u}) = 1.10471u_1^2u_2 + 0.04811u_3u_4(14.0+u_2),\\
\textit{Subject to}\quad & g_1(\vec{u}) = \tau(\vec{u}) - \tau_{max} \leqslant 0,\\
& g_2(\vec{u}) = \delta(\vec{u}) - \delta_{max} \leqslant 0,\\
& g_3(\vec{u}) = \sigma(\vec{u}) - \sigma_{max} \leqslant 0,\\
& g_4(\vec{u}) = P - P_c(\vec{u}) \leqslant 0,\\
& g_5(\vec{u}) = u_1 - u_4 \leqslant 0,\\
& g_6(\vec{u}) = 1.10471u_1^2 + 0.04811u_3u_4(14.0+u_2) - 5.0 \leqslant 0,\\
& g_7(\vec{u}) = 0.125 - u_1 \leqslant 0\\
\textit{Variables' range}\quad & 0.1 \leqslant u_1 \leqslant 2,\\
& 0.1 \leqslant u_2 \leqslant 10,\\
& 0.1 \leqslant u_3 \leqslant 10,\\
& 0.1 \leqslant u_4 \leqslant 2\\
\textit{where}\quad & \tau(\vec{u}) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{u_2}{2R} + (\tau'')^2},\\
& \tau' = \frac{p}{\sqrt{2u_1u_2}},\ \ \tau'' = \frac{MR}{J},\\
& M = P(L + \frac{u_2}{2}),\\
& R = \sqrt{\frac{u_2^2}{4} + (\frac{u_1+u_3}{2})^2},\\
& J = 2\Big\{\sqrt{2u_1u_2}\Big[\frac{u_2^2}{4} + (\frac{u_1+u_3}{2})^2\Big]\Big\},\\
& \sigma(\vec{u}) = \frac{6PL}{u_4u_3^2},\ \ \delta(\vec{u}) = \frac{6PL^3}{Eu_3^2u_4}\\
& P_c(\vec{u}) = \frac{\sqrt[4.013E]{\frac{u_3^2u_4^6}{36}}}{L^2}(1 - \frac{u_3}{2L}\sqrt{\frac{E}{4G}}),\\
& P = 6000\,lb, L = 14\,in.,\ \ \delta_{max} = 0.25\,in.,\\
& E = 30 \times 1^6\ \text{psi},\ \ G = 12 \times 10^6\ \text{psi},\\
& \tau_{max} = 13600\ \text{psi},\ \ \sigma_{max} = 30000\ \text{psi}
\end{aligned}
\tag{26}
$$

RSRFT was compared to several optimization techniques, such as the whale optimization algorithm (WOA) [41], MVO [12], evolutionary particle swarm optimization (CPSO) [42], LSHcEpS [43], co-evolutionary differential evolution (CSCA) [44], the simplex method (SIMPLEX) [45], Davidon–Fletcher–Powell (DAVID) [45], the gravitational search algorithm (GSA) [46], the GA [47], harmony search (HS) [48], and LSHADE_SPACMA (we renamed it as LSHSPCM) [49]. The developed RSRFT was conducted for 25 independent runs with $t_{max} = 500$ and $N = 25$.

Table 1 contains the results of RSRFT and the compared optimization techniques. The RSRFT technique had the lowest cost (as given in the column Optimal Objective), followed by LSHSPCM, while SIMPLEX had the highest cost. As a result, the design parameters obtained by RSRFT were more suitable for WBD.

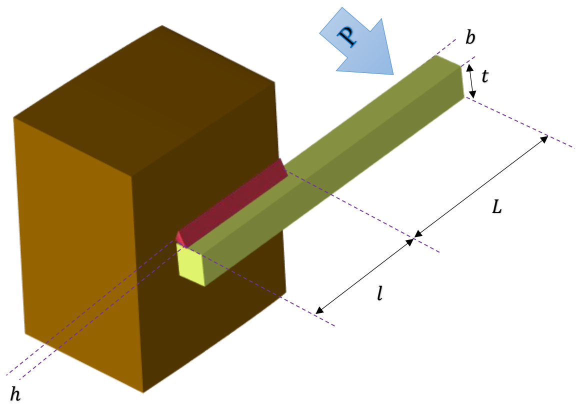

Figure 2. Structure of the WBD problem.

Table 1. The value of the estimated parameters using RSRFT and other methods for solving the WBD problem.

Algorithm	*H*	*L*	*t*	*b*	**Optimal Objective**
RSRFT	0.20572	3.4704	9.0370	0.2057	1.72489
RFO	0.21846	3.51024	8.87254	0.22491	1.86612
LSHSPCM	0.2057	3.4705	9.0366	0.2057	1.7249
RSA	0.14468	3.514	8.9251	0.21162	1.6726
LSHcEpS	0.2038	3.5148	9.0486	0.2057	1.7294
OBLGOA [50]	0.205769	3.471135	9.032728	0.2059072	1.7257
RO [51]	0.203687	3.528467	9.004233	0.207241	1.735344
HS [48]	0.2442	6.2231	8.2915	0.2443	2.3807
DAVID [45]	0.2434	6.2552	8.2915	0.2444	2.3841
SIMPLEX [45]	0.2792	5.6256	7.7512	0.2796	2.5307
CPSO [42]	0.202369	3.544214	9.04821	0.205723	1.72802
MVO [12]	0.205463	3.473193	9.044502	0.205695	1.72645
GA [47]	0.205986	3.471328	9.020224	0.20648	1.728226
GSA [46]	0.182129	3.856979	10	0.202376	1.87995
CSCA [44]	0.203137	3.542998	9.033498	0.206179	1.733461
WOA [41]	0.205396	3.484293	9.037426	0.206276	1.730499

4.1.2. Tension/Compression Spring Design Problem

Within this section, we evaluated the performance of RSRFT to allocate the parameters that are used to minimize spring weight during tension/compression spring design (TCSD), which is defined in Figure 3. Those parameters include the wire diameter (d), the mean coil diameter (D), and the number of active coils (N) of a spring. The formulation of TCSD is illustrated as

$$
\begin{aligned}
\textit{Consider} \quad & \vec{u} = [u_1\ u_2\ u_3] = [d\ D\ N], \\
\textit{Minimize} \quad & f(\vec{u}) = (u_3 + 2)u_2 u_1^2, \\
\textit{Subject to} \quad & g_1(\vec{u}) = 1 - \frac{u_2^3 u_3}{71785 u_1^4} \leqslant 0, \\
& g_2(\vec{u}) = \frac{4u_2^2 - u_1 u_2}{12566(u_2 u_1^3 - u_1^4)} + \frac{1}{5108 u_1^2} \leqslant 0, \\
& g_3(\vec{u}) = 1 - \frac{140.45 u_1}{u_2^2 u_3} \leqslant 0, \\
& g_4(\vec{u}) = \frac{u_1 + u_2}{1.5} - 1 \leqslant 0, \\
\textit{Variables' range} \quad & 0.05 \leqslant u_1 \leqslant 2 \\
& 0.25 \leqslant u_2 \leqslant 1.30 \\
& 2.00 \leqslant u_3 \leqslant 15
\end{aligned}
\tag{27}
$$

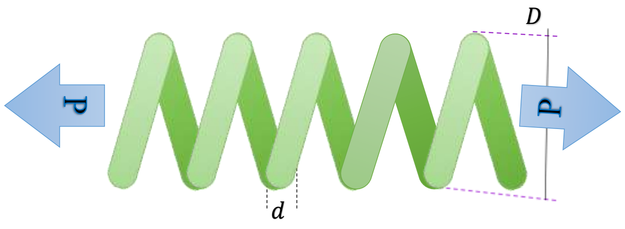

Figure 3. Design of the TCSD problem.

In this experiment, RSRFT was compared with other approaches that have been used to solve the TCSD problem, for example CPSO [42], MVO [12], the GA [52], the GSA [41], the evolution strategy (ES) [53], the Ray–Saini method [54], the WOA [41], the CSCA [44], the method proposed by Belegundu and Arora [55], LSHSPCM, and LSHcEpS. The results of the comparison between RSRFT and the others are given in Table 2. The RSRFT offered superior outcomes over the other methods.

Table 2. The value of the estimated parameters using RSRFT and others to solve the TCSD problem.

Algorithm	d	D	N	Optimal Objective
RSRFT	0.05147146	0.3515050	11.6013141	0.01266617
RFO	0.052667011	0.3806680	10.0213925	0.0126934
RSA [17]	0.057814	0.58478	4.0167	0.01176
LSHSPCM	0.0535	0.4010	9.0962	0.012721349
LSHcEpS	0.0517	0.3567	11.2876	0.01266523
OBLGOA [50]	0.0530178	0.38953229	9.6001616	0.01270136
Belegundu-Arora method [55]	0.0500	0.3177	14.026	0.012730
GA [52]	0.05148	0.351661	11.632201	0.01270478
WOA [41]	0.051207	0.345215	12.004032	0.0126763
CPSO [42]	0.051728	0.357644	11.244543	0.0126747
ES [53]	0.051643	0.35536	11.397926	0.012698
MVO [12]	0.05251	0.37602	10.33513	0.012790
GSA[41]	0.050276	0.323680	13.525410	0.0127022
Ray–Saini method [54]	0.321532	0.050417	13.979915	0.013060

4.1.3. Pressure Vessel Design Problem

This section presents the performance of developed RSRFT to solve another engineering problem, named the pressure vessel design (PVD) problem (see Figure 4), by determining the parameters that lead to minimizing the cost of the pressure cylinder. Those parameters are the thickness of the head Th, the length of the cylindrical section of the vessel L, the thickness Ts, and the inner radius R. The mathematical representation of PVD is given as

$$\begin{aligned}
&\textit{Consider} && \vec{u} = [u_1\ u_2\ u_3\ u_4] = [T_s\ T_h\ R\ L],\\
&\textit{Minimize} && 0.6224u_1u_3u_4 + 1.7781u_2u_3^2\\
& && + 3.1661u_1^2u_4 + 19.84u_1^2u_3,\\
&\textit{Subject to} && g_1(\vec{u}) = -u_1 + 0.0193u_3 \leqslant 0,\\
& && g_2(\vec{u}) = -u_2 + 0.00954u_3 \leqslant 0,\\
& && g_3(\vec{u}) = -\pi u_3^2u_4 - \frac{4}{3}\pi u_3^3 + 1296000 \leqslant 0,\\
& && g_4(\vec{u}) = u_4 - 240 \leqslant 0,\\
&\textit{Variables' range} && 0 \leqslant u_1 \leqslant 99,\\
& && 0 \leqslant u_2 \leqslant 99,\\
& && 10 \leqslant u_3 \leqslant 200,\\
& && 10 \leqslant u_4 \leqslant 200
\end{aligned} \tag{28}$$

The outcomes of the comparisons are listed in Table 3. Those approaches are HPSO [56], ACO [57], the ES [53], the GA [52], PSO-DE [58], the GSA [41], CDE [44], and LSHSPCM [49]. From these results, we noticed that RSRFT was able to reach the minimum objective value (as given in the column Optimal Objective), which outperformed the other compared techniques.

Table 3. The value of the estimated parameters using RSRFT and other approaches to solve the PVD problem.

Estimated Values for Parameters					
Method	***Th***	***Ts***	***R***	***L***	**Optimal Objective**
RSRFT	0.81612257	0.403409949	42.2861349	174.325078	5953.4364
RFO	0.81425	0.44521	42.20231	176.62145	6113.3195
RSA	0.8400693	0.4189594	43.38117	161.5556	6034.7591
LSHSPCM	0.9808	0.5251	50.1435	99.5532	6654.8902747
OBLGOA [50]	0.81622	0.40350	42.291138	174.811191	5966.67160
PSO-DE [58]	0.8125	0.4375	42.098446	176.6366	6059.71433
HPSO [56]	0.8125	0.4375	42.0984	176.6366	6059.7143
ACO [57]	0.8125	0.4375	42.098353	176.637751	6059.7258
CDE [44]	0.8125	0.4375	42.098411	176.63769	6059.734
ES [53]	0.8125	0.4375	42.098087	176.640518	6059.7456
GA [52]	0.8125	0.4375	42.097398	176.65405	6059.94634
GSA [41]	1.125	0.625	55.9886598	84.4542025	8538.8359

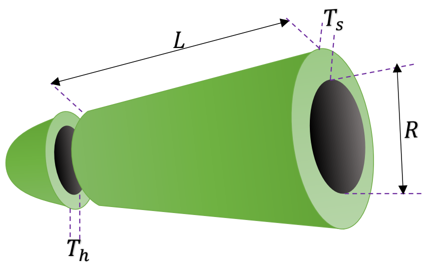

Figure 4. Schematic of the PVD problem.

4.1.4. Three-Bar Truss Design Problem Design

The major aim of Three-Bar Truss Design (TBTD) is to minimize the weight of the structure [59,60]. Figure 5 illustrates the geometry structure of TBTD, in which cross-sectional areas stand in for the design variables. The cross-section with $A_1(=x_1)$ and $A_2(=x_2)$ must be found according to [59] because of the system's symmetry. Below is a diagram that represents the TBTD problem mathematically.

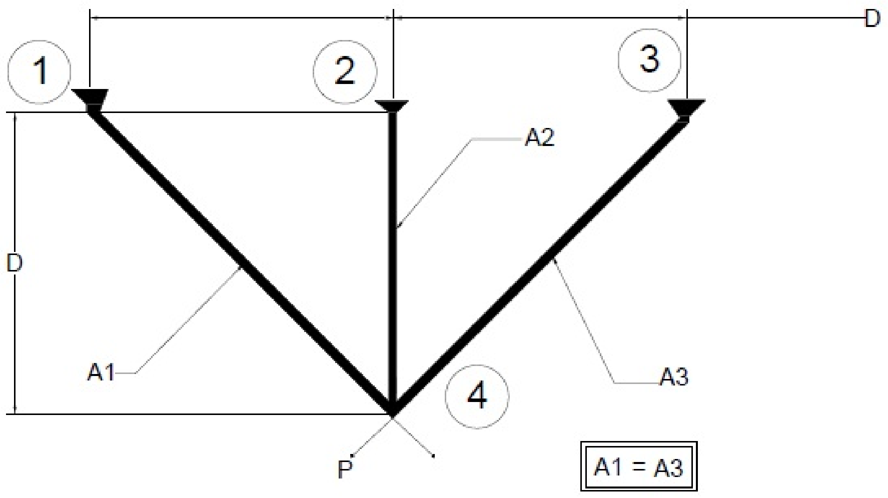

Figure 5. The three-bar truss design problem.

$$
\begin{aligned}
&\textit{Consider} && f(x) = (2\sqrt{2x_1} + x_2) \times l \\
&\textit{Minimize} && f(\vec{u}) = (u_3 + 2)u_2u_1^2, \\
&\textit{Subject to} && g_1(x) = \frac{\sqrt{x1}x_1 + x_2}{\sqrt{2}x_1^2 + 2x_1x_2}P - \sigma \leq 0, \\
& && g_2(x) = \frac{x_2}{\sqrt{2}x_1^2 + 2x_1x_2}P - \sigma \leq 0, \\
& && g_3(x) = \frac{1}{\sqrt{2}x_2 + x_1}P - \sigma \leq 0 \\
& && g_4(\vec{u}) = \frac{u_1 + u_2}{1.5} - 1 \leqslant 0, \\
&\textit{Variables' range} && l = 100\text{ cm}, P = 2\text{ kN/cm}^2, \sigma = 2\text{ kN/cm}^2, \\
& && (0 \leq x_1, x_2 \leq 1,)
\end{aligned}
\tag{29}
$$

RSRFT was used to allocate the design variables. Furthermore, we compared RSRFT to other methods used to address the TBTD problem, which are shown in Table 4. Since RSRFT obtained the smallest weight (as given under Optimal Weight), it is clear from the findings in this table that it offered the best solution. However, the outcomes of the traditional RSA were better than the competitive algorithms, including RSRFT.

Table 4. The value of the estimated parameters using RSRFT to solve the 3-bar truss design problem.

Algorithm	Estimated Values for Parameters		Optimal Weight
	x_1	x_2	
RSRFT	0.78875052	0.4080351	263.89584
RFO	0.75356	0.55373	268.51195
RSA [17]	0.78873	0.40805	263.8928
DEDS [61]	0.78867513	0.40824828	263.89584
SSA [62]	0.78866541	0.408275784	263.89584
MBA [63]	0.7885650	0.4085597	263.89585
PSO-DE [58]	0.7886751	0.4082482	263.89584
Ray and Saini [54]	0.795	0.395	264.3
CS [64]	0.78867	0.40902	263.9716
AAA [65]	0.7887354	0.408078	263.895880
GOA [66]	0.78889755557	0.40761957011	263.89588149

4.1.5. Speed Reducer Problem

In order for an airplane's propeller and engine to rotate at an efficient speed, a gearbox between them is needed (see Figure 6 [67]). The optimization strategy employed to address the speed reducer problem is intended for minimizing the design's weight while still meeting the requirements for the bending stress of the gear teeth, surface stress, transverse shaft deflections, and stresses in the shafts. The best value for each of the seven design factors must be established to solve this problem.

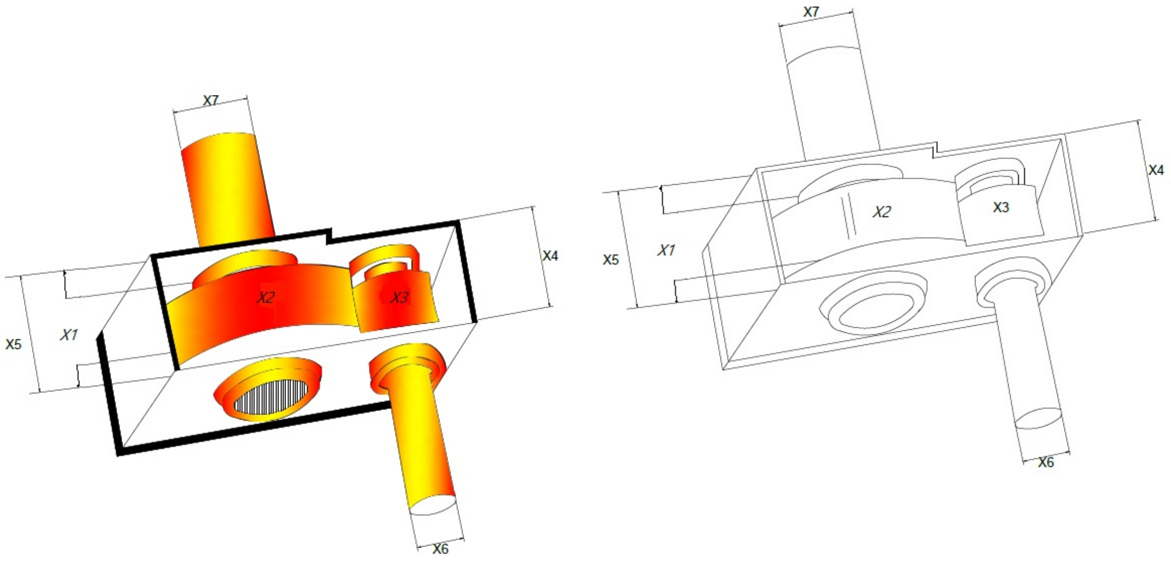

Figure 6. The speed reducer problem.

These variables are the face width (x_1), the module of the teeth (x_2), the number of teeth on the pinion (x_3), the lengths of the first and second shafts between the bearings (x_4 and x_5, respectively), the first shaft's diameter (x_6 and x_7, respectively), and the module of the teeth (x_2). Given that it contains more limitations than other issues, the speed reducer problem is defined as having a high degree of complexity. Below is a diagram that represents this issue mathematically.

$$
\begin{aligned}
\textit{Assume} \quad & f(x) = 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) - \\
& 1.508x_1(x_6^2 + x_7^2) + 7.4777x_6^3 + x_7^3 + 0.7854x_4x_6^2 + x_5x_7^2 \\
\textit{Minimize} \quad & f(\vec{u}) = (u_3 + 2)u_2u_1^2, \\
\textit{Subject to} \quad & g(1) = \frac{27}{x_1x_2^2x_3} - 1 \leq 0, \\
& g(2) = \frac{397.5}{x_1x_2^2x_3^2} - 1 \leq 0, \\
& g(3) = \frac{1.93x_4^3}{x_2x_3x_6^4} - 1 \leq 0, \\
& g(4) = \frac{1.93x_5^3}{x_2x_3x_7^4} - 1 \leq 0, \\
& g(5) = \frac{1}{110x_6^3}\sqrt{(\frac{745x_4}{x_2x_3})^2 + 16.9 \times 10^6} - 1 \leq 0, \\
& g(6) = \frac{1}{85x_7^3}\sqrt{(\frac{745x_5}{x_2x_3})^2 + 157.5 \times 10^6} - 1 \leq 0, \\
& g(7) = \frac{x_2x_3}{40} - 1 \leq 0, \\
& g(8) = \frac{5x_2}{x_1} - 1 \leq 0, \\
& g(9) = (x_1/12x_2) - 1 \leq 0, \\
& g(10) = \frac{1.5x_6 + 1.9}{x_4} - 1 \leq 0, \\
& g(11) = \frac{1.1x_7 + 1.9}{x_5} - 1 \leq 0 \\
\textit{Variables' range} \quad & (0.7 \leq x_2 \leq 0.8), (2.6 \leq x_1 \leq 3.6), (17 \leq x_3 \leq 28), (7.3 \leq x_4 \leq 8.3), \\
& (2.9 \leq x_6 \leq 3.9), (7.3 \leq x_5 \leq 8.3), (5 \leq x_7 \leq 5.5)
\end{aligned} \tag{30}
$$

Table 5 provides the comparative findings between RSRFT and the other MH approaches that have been given in the literature. It is clear that RSRFT surpassed the majority of the techniques that were examined (as given under Optimal Weight), and it was generally assigned the third rank. CS and SBSM were assigned the first and second ranks, respectively [64,68]. Since all three algorithms almost had the same ideal weight, there was little difference between them.

Table 5. The value of the estimated parameters using RSRFT to solve the speed reducer design problem.

Method	Estimated Values for Parameters							Optimal Weight
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	
RSRFT	3.5000055	0.7	17	7.305888	8.004689	3.3502353	5.2868060	3000.97899
RFO	3.500001	0.7	17.00002	7.314497	8.0294718	3.350253	5.2867662	3001.5811
RSA [17]	3.50279	0.7	17	7.30812	7.74715	3.35067	5.28675	2996.5157
GA [69]	3.510253	0.7	17	8.35	7.8	3.362201	5.287723	3067.561
GSA [46]	3.600000	0.7	17	8.3	7.8	3.369658	5.289224	3051.120
HS [70]	3.520124	0.7	17	8.37	7.8	3.366970	5.288719	3029.002

Table 5. *Cont.*

Method	Estimated Values for Parameters							Optimal Weight
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	
SES [71]	3.506163	0.700831	17	7.460181	7.962143	3.362900	5.308949	3025.005127
MDA [72]	3.5	0.7	17	7.3	7.670396	3.542421	5.245814	3019.583365
SBSM [68]	3.506122	0.700006	17	7.549126	7.859330	3.365576	5.289773	3008.08
SCA [11]	3.508755	0.7	17	7.3	7.8	3.461020	5.289213	3030.563
CS [64]	3.5015	0.7000	17	7.6050	7.8181	3.3520	5.2875	3000.9810
PSO [73]	3.5001	0.7000	17.0002	7.5177	7.7832	3.3508	5.2867	3145.922
FA [74]	3.507495	0.7001	17	7.719674	8.080854	3.351512	5.287051	3010.137492
hHHO-SCA [75]	3.506119	0.7	17	7.3	7.99141	3.452569	5.286749	3029.873076

4.1.6. Multiple Disc Clutch Brake Problem

Finding the values of five design factors to reduce the mass of an MDCB is the primary goal of research into the multiple disc clutch brake (MDCB) problem, which was cited in [63]. The design parameters are shown in Figure 7 as the inner radius x_1, outer radius x_2, disc thickness x_3, actuation force x_4, and number of friction surfaces x_5. Below is a diagram that represents this issue mathematically.

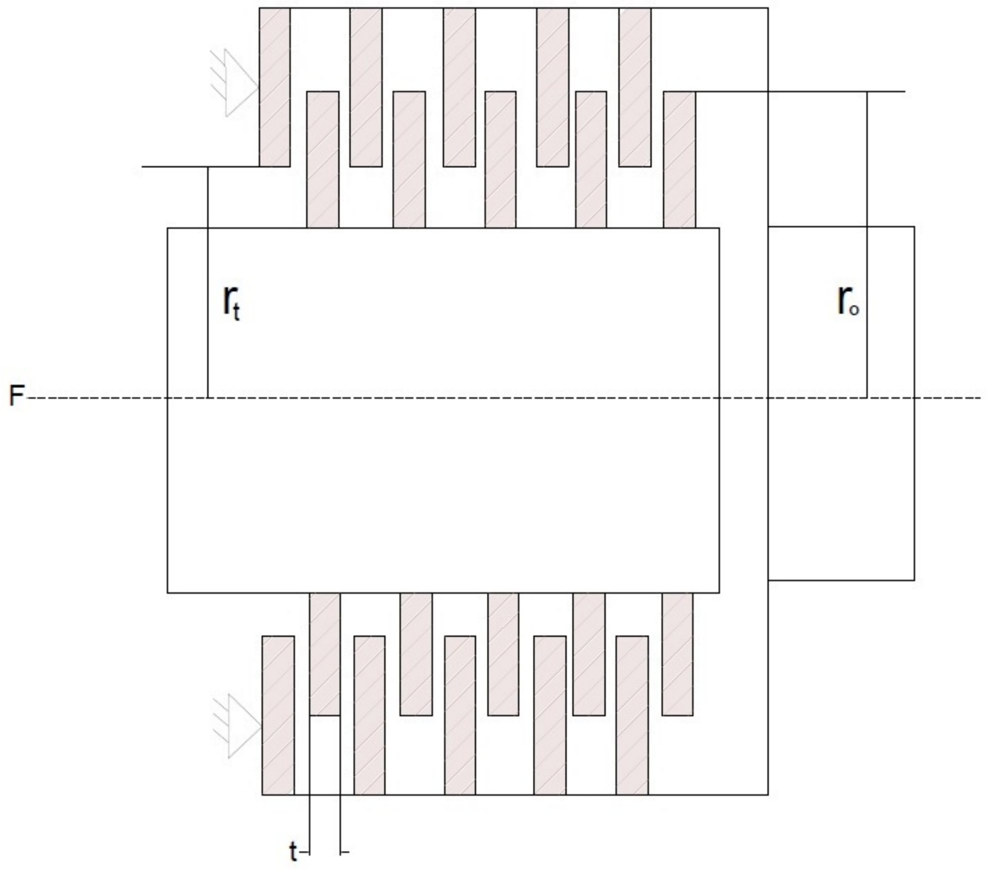

Figure 7. The multiple disc clutch brake problem.

$$\begin{aligned}
\textit{Assume} \quad & f(x) = \Pi(r_o^2 - r_i^2)t(Z+1)\rho \\
& 1.508x_1(x_6^2 + x_7^2) + 7.4777x_6^3 + x_7^3 + 0.7854x_4x_6^2 + x_5x_7^2 \\
\textit{Minimize} \quad & f(\vec{u}) = (u_3+2)u_2u_1^2, \\
\textit{Subject to} \quad & g_1(x) = l_{max} - (Z+1)(t+\delta) \geq 0, g_2(x) = r_o - r_i - \Delta r \geq 0 \\
& g_3(x) = P_{max}\, v_{sr\ max} - P_{rz}\, v_{sr} \geq 0\, g_4(x) = Pmax - Prz \geqslant 0, \\
& g_5(x) = v_{sr\ max} - v_{sr} \geq 0,\ g_6 = T_{max} - T \geq 0 \\
& g_7(x) = M_h - sM_s \geq 0, g_8(x) = T \geq 0 \\
\textit{where} \quad & M_h = \frac{2}{3}\mu FZ\frac{r_o^3 - r_i^2}{r_o^2 - r_i^3},\quad P_{rz} = \frac{F}{\Pi(r_o^2 - r_i^2)}, v_{rz} = \frac{2\Pi(r_o^3 - r_i^3)}{90(r_o^2 - r_i^2)},\quad T = \frac{I_z\,\Pi\, n}{30(M_h + M_f)} \\
& \Delta r = 20\ \text{mm},\ I_z = 55\ \text{kgmm}^2,\ P_{max} = 1\ \text{MPa},\ F_{max} = 1000\ \text{N}, \\
& v_{sr\ max} = 10\ \text{m/s},\ l_{max} = 30\ \text{mm},\ r_{i\ min} = 60, \\
& T_{max} = 15\ \text{s},\ \mu = 0.5,\ s = 1.5,\ M_s = 40\ \text{Nm}, M_f = 3\ \text{Nm},\ n = 250\ \text{rpm}, \\
& F_{max} = 1000,\ Z_{min} = 2,\ Z_{max} = 9, r_{i\ max} = 80, r_{o\ min} = 90, \\
& r_{o\ max} = 110,\ t_{min} = 1.5, t_{max} = 3,\ F_{min} = 600
\end{aligned} \tag{31}$$

Table 6 lists the outcomes of RSRFT and other MH approaches collected from the literature. From this table, it is clear that RSRFT obtained the ideal cost (0.31176) (as given under Optimal Weight), which was then followed by CMVO, MFO, MVO, and the WCA, which had the same efficiency.

Table 6. The value of the estimated parameters using RSRFT to solve the multiple disc clutch brake problem.

Method	Estimated Values for Parameters					Optimal Weight
	x_1	x_2	x_3	x_4	x_5	
RSRFT	69.003908	89.003914	1	789.52330	2.965888	0.307109
RFO	72	93	762	2	1	0.25359
RSA [17]	70.0347	90.0349	1.0000	801.7285	2.9740	0.31176
TLBO [76]	70	90	1	810	3	0.313656611
NSGA-II [77]	70	90	1.5	1000	3	0.470400
WCA [78]	70	90	1	910	3	0.313656
MVO [79]	70	90	1	910	3	0.313656
CMVO [79]	70	90	1	910	3	0.313656
MFO [80]	70	90	1	910	3	0.313656

4.2. Series of Analysis 2: RSRFT for Security in the IoT

Within this section, we evaluated the applicability of RSRFT to improve the detection of IDSs in the IoT environment [81]. In general, the problem definition of IDSs in the Internet of Things (IoT) is the need to detect and prevent unauthorized access, data breaches, and malicious activities in IoT devices and networks. With the rapid growth of the IoT, there has been an increase in the number and complexity of connected devices, which has also led to an increase in the number of potential attack vectors and vulnerabilities. As a result, there is a growing need for effective IDSs that can identify and respond to these threats in real-time [82].

The challenge of developing IDSs for the IoT is the complexity of the networks and the diversity of the devices. The systems need to be designed to detect and respond to a wide range of attack types, including malware, denial-of-service attacks, and insider threats. Furthermore, the systems need to be able to identify anomalous behaviors and patterns that may indicate an attack, while also avoiding false positives [83].

To address this problem, we present an alternative technique to detect the intrusions in the IoT. In this experiment, RSRFT was used as a feature selection technique to enhance the process of selecting the important features of the collected data in the IoT environment. This can be achieved through by the Boolean version of RSRFT by converting the real value of each solution into a binary value.

The proposed RSRFT starts by generating the value of the initial population. This can be defined as

$$X = LB + rand \times (UB - LB) \tag{32}$$

where UB and LB are the upper and lower boundaries, respectively. In this experiment, the values of $UB = 1$ and $LB = 0$, while $rand \in [0, 1]$. This step is followed by obtaining the binary of X_i using the following formula:

$$BX_i = \begin{cases} 1 & if\ X_i > 0.5 \\ 0 & otherwise \end{cases} \tag{33}$$

The next process is to assess the quality of the selected features, which correspond to the ones in X_i, using the following formula.

$$Fit_i = \eta \times \gamma_i + (1 - \eta) \times (\frac{|BX_i|}{D}) \tag{34}$$

where γ_i denotes the classification error of the KNN classifier, whereas the term $(\frac{|BX_i|}{D})$ refers to the ratio of features selected using X_i. The value of $\eta \in [0, 1]$ is applied as the weight of the two terms of Equation (34).

Thereafter, according to the fitness values, we determined the best of them and their corresponding solution X_b. Then, we used this solution and the operators of RSRFT that are discussed in Sections 3.2 and 3.3 to update X. The next step after reaching the stop conditions is to compute the efficiency of X_b using the testing set of IoT data.

In the following sections, we introduce the datasets used in this experiment to assess RSRFT as an FS method. The comparison was conducted with moth–flame optimization (MFO) [60], the bat (BAT) algorithm [84], transient search optimization (TSO) [83], and the RSA. The parameter of each approach was assigned based on the original implementation.

4.2.1. Dataset Description

The proposed algorithm was assessed using the KDDCup-99, NSL-KDD, BoT-IoT, and CICIDS-2017 datasets. The KDD-99 dataset, created by the Defense Advanced Research Project Agency (DARPA) in 1988, contains network traffic data. Another kind of dataset, called NSL-KDD, was also used, which is a derived dataset with no duplicate network traffic records and grouped into four types: Probe, U2R, R2L, and DoS, in addition to normal data. The CICIDS-2017 dataset [85] was obtained from 25 clients at the Canadian Institute for Cybersecurity (CIC) using the CICFlowMeter tool to simulate realistic network traffic obtained using protocols such as HTTP,SSH, HTTPS, FTP, and email. Additionally, the Bot-IoT dataset [86] includes 3.5 million records produced from different IoT devices and features botnet attacks.The description of all of these datasets is given in Table 7.

4.2.2. Evaluation Criteria and Experimental Setup

In this study, various performance and evaluation criteria were applied to evaluate the effectiveness of RSRFT, including the average accuracy (AV_{Acc}), the average recall (AV_{Sens}), and the average precision (AV_{Prec}):

- AV_{Acc}: This stands for the rate of the correct detection of intrusions in the IoT environment. AV_{Acc} can be formulated as

$$AV_{Acc} = \frac{1}{N_r} \sum_{k=1}^{N_r} Acc^k_{Best}, \ Acc_{Best} = \frac{TP + TN}{TP + FN + FP + TN} \tag{35}$$

where $N_r = 25$ is the total number of runs. FN, TN, TP, and FP refer to false negative, true negative, true positive, and false positive, respectively.

- AV_{Sens}: This can also be referred to as the true positive rate, and it describes the percentage of correctly predicted positive intrusions. AV_{Sens} is computed as

$$AV_{Sens} = \frac{1}{N_r}\sum_{k=1}^{N_r} Sens_{Best}^{k}, \; Sens_{Best} = \frac{TP}{TP + FN} \tag{36}$$

- AV_{Prec}: This describes the ratio of true detections among all correct detection samples, and it is formulated as

$$AV_{Prec} = \frac{1}{N_r}\sum_{k=1}^{N_r} Prec_{Best}^{k}, \; Prec_{Best} = \frac{TP}{FP + TP} \tag{37}$$

Table 7. Description of the datasets.

Target Class	KDDCup-99		NSL-KDD		Target Class	Bot-IoT		Target Class	CICIDS-2017	
	Train	Test	Train	Test		Train	Test		Train	Test
Normal	97,278	60,593	67,343	9710	Normal	370	107	Benign	727,397	163,572
DoS	391,458	229,853	45,927	7458	DoS	1,320,148	385,309	DDoS	112,901	25,388
Probe	4107	4166	11,656	2422	DDoS	1,541,315	330,112	FTP-Patator	6997	1574
								SSH-Patator	5201	1169
R2L	1126	16,189	995	2887	Reconnaissance	72,919	18,163	PortScan	140,043	31,492
								Brute Force	1329	299
U2R	52	228	52	67	Theft	65	14	SQL Injection	19	4
								XSS	575	129

4.2.3. Results and Discussion

The performance of the enhanced RSA based on TMO and RFO as the feature selection technique to improve the detection of IDSs in the IoT environment is discussed in this section. Table 8 shows the average results of the developed RSRFT to improve the detection of IDSs in the IoT environment using the training and testing sets. From these results, it can be noticed that RSRFT had a high ability to improve the detection of the IDSs in terms of accuracy. In addition, it can be observed that the behavior of RFO alone was the worst one among the tested algorithms using the training and testing sets. According to the results of the precision, recall, and F1-measure, it can be noticed that the developed algorithm was the first-ranked algorithm. This indicated that the ability of the developed RSRFT was better than exploring and exploiting the search space.

Figures 8 and 9 show the average accuracy, precision, F1-measure, and recall using the training and testing sets, respectively. From these figures, we can see that the average of each measure (i.e., accuracy, precision, F1-measure, and recall) of RSRFT was better than that of the other methods using the training and testing sets. This was followed by the RSA, which provided better results than the other methods.

Moreover, we used the Friedman test as a non-parametric test to assess if there was a significant difference between the developed RSRFT and the others. Table 9 shows the mean rank obtained using each approach over the training and testing sets and among the performance measures. From these results, it can be noticed that the developed RSRFT had the highest mean rank, and the difference between it and the others was significant since the p-value was less than 0.05, except for the precision value using the training set.

From the previous results of the two experimental series, it can be observed that the developed RSRFT had high efficiency to determine the solution of different engineering problems, whereas we noticed the high ability of RSRFT to enhance the process of improving the detection of IDSs in the IoT environment by reducing the number of selected features. This resulted from boosting the operators of the RSA, RFO, and TMO to improve the rate of convergence towards the optimal solution. However, there are some limitations in RSRFT, for example the initial population can influence the performance of the developed method. In addition, the diversity of the solutions needs more improvement.

Table 8. Results of RSRFT for improving the detection of IDSs in the IoT environment.

		Train				Test			
		Accuracy	**Precision**	**F1-Measure**	**Recall**	**Accuracy**	**Precision**	**F1-Measure**	**Recall**
KDD99	RSRFT	**99.946**	**99.483**	**99.943**	**99.923**	**93.615**	**92.649**	**90.380**	93.495
	RFO	92.275	92.414	97.304	93.126	84.375	82.501	87.351	85.225
	BAT	98.007	94.847	97.337	98.247	90.347	89.134	90.093	90.587
	TSO	95.439	91.027	97.437	94.919	87.536	80.791	87.479	87.016
	MFO	96.073	97.631	98.371	97.123	88.175	87.763	88.420	89.225
	RSA	99.910	99.909	99.906	99.910	92.040	89.684	89.985	92.040
NSL-KDD	RSRFT	**99.382**	**99.545**	**99.548**	**99.301**	**76.407**	**82.371**	**72.731**	**77.107**
	RFO	91.947	92.080	96.968	92.797	67.951	71.131	68.907	68.801
	BAT	97.669	94.501	96.989	97.909	73.671	73.501	68.905	73.911
	TSO	95.078	90.657	97.067	94.558	71.330	71.298	69.697	70.810
	MFO	95.745	97.297	98.035	96.795	71.626	76.122	69.844	72.676
	RSA	99.201	99.158	99.148	99.201	76.107	82.171	71.731	76.107
BIoT	RSRFT	**99.568**	**99.568**	**99.568**	**99.568**	**99.512**	**99.420**	**99.080**	**99.064**
	RFO	99.472	99.472	99.472	99.472	98.956	98.957	99.005	98.964
	BAT	99.475	99.475	99.474	99.475	99.019	98.987	99.012	99.021
	TSO	99.460	99.459	99.459	99.460	98.986	98.941	99.005	98.981
	MFO	99.480	99.480	99.480	99.480	98.998	99.013	99.020	99.009
	RSA	98.829	98.829	98.829	98.829	99.020	99.098	99.070	99.038
CIC2017	RSRFT	**99.941**	**99.920**	**99.918**	**99.931**	**99.931**	**99.947**	**99.983**	**99.931**
	RFO	99.690	99.490	99.450	99.690	99.430	99.240	99.190	99.430
	BAT	99.490	99.630	99.440	99.640	99.230	99.360	99.180	99.380
	TSO	99.680	99.750	99.680	99.710	99.420	99.480	99.420	99.450
	MFO	99.360	99.370	99.480	99.430	99.100	99.120	99.220	99.170
	RSA	99.911	99.910	99.889	99.911	99.911	99.907	99.888	99.911

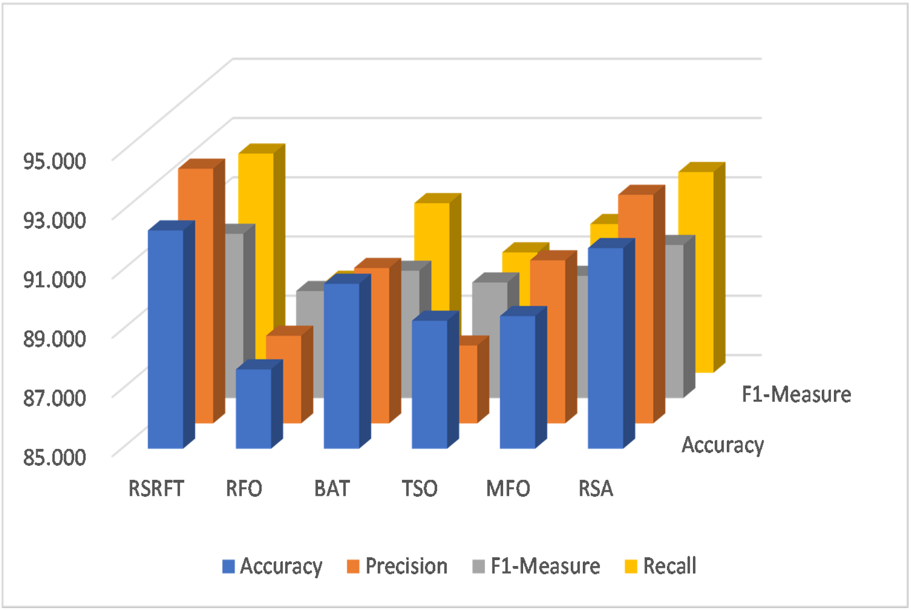

Figure 8. Performance measures of RSRFT using the training set.

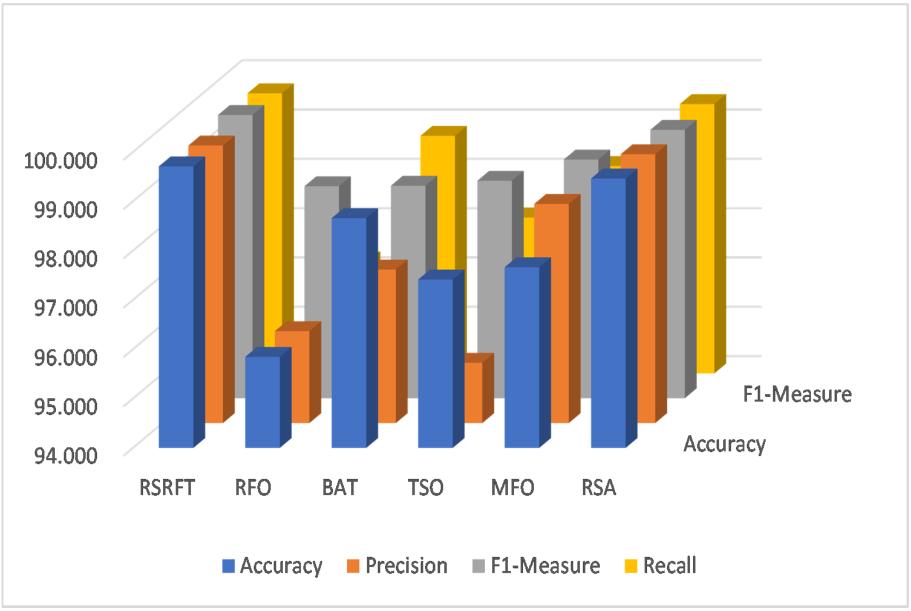

Figure 9. Performance measures of RSRFT using the testing set.

Table 9. The mean rank using the Friedman test.

		p-Value	RSRFT	RFO	BAT	TSO	MFO	RSA
Training	Accuracy	0.046	6	2.25	3.5	2.25	3	4
	Precision	0.0543	5.75	2.25	3.25	2	3.5	4.25
	F1-Measure	0.0208	6	1.75	2.25	3	4	4
	Recall	0.0435	6	2	3.5	2.5	3	4
Testing	Accuracy	0.0064	6	1.75	3.5	2.25	2.5	5
	Precision	0.0064	6	1.75	3.25	2	3	5
	F1-Measure	0.0101	6	1.625	2.5	2.625	3.5	4.75
	Recall	0.0054	6	1.5	3.5	2.5	2.5	5

5. Conclusions

In this paper, the use of an enhanced version of the reptile search algorithm (RSA) to solve engineering problems was shown to be a highly effective approach. By combining the strengths of the RSA with those of the red fox algorithm (RFO) and triangular mutation operator (TMO), this modification was able to effectively explore a wider range of possible solutions and find improved solutions more quickly. The results of this study demonstrated the effectiveness of the developed approach, named RSRFT, in solving a variety of engineering problems and showed that this algorithm can provide reliable and accurate solutions in a variety of contexts. Overall, the use of RSRFT represents a promising advancement in the field of engineering and has the potential to greatly improve the design and optimization of various systems and technologies. In the second experiment series, we evaluated RSRFT as a feature selection method to improve the classification of intrusion detection systems (IDSs) in the IoT and cloud environments. Four IDS datasets were utilized to test the performance of RSRFT, namely, KDD99, NSL-KDD, BIoT, and CIC2017. The results also confirmed the superiority of RSRFT.

In future works, the presented RSRFT can be extended to other areas, including task scheduling in cloud and fog environments, medical applications, agriculture, and others. Additionally, it may be tested on multi-objective optimization problems and solar cell parameter estimation.

Author Contributions: All authors contributed equally to this work. All authors have read and agreed to the published version of the manuscript.

Funding: This research project was funded by the Deanship of Scientific Research, Princess Nourah bint Abdulrahman University, through the Program of Research Project Funding After Publication, grant No (43- PRFA-P-24).

Data Availability Statement: All of the datasets are public, as we described in the main texts.

Acknowledgments: This research project was funded by the Deanship of Scientific Research, Princess Nourah bint Abdulrahman University, through the Program of Research Project Funding After Publication, grant No (43- PRFA-P-24).

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

Nomenclature

Acronyms	
IDS	Intrusion detection system
IoT	Internet of Things
MH	Metaheuristic
RFO	Red fox algorithm
RSA	Reptile search algorithm
TMO	Triangular mutation operator
Variables	
$\overline{X}_c$	Combination vector triangle
η_{ij}	Hunting operator
$Best_j$	Best solution
$M(x_i)$	Mean value of X
N	Number of solutions
t	Iteration
X	Population
$x_{(r_1,j)}$	Value of randomly selected solution

References

1. Sarkar, A.; Guchhait, R.; Sarkar, B. Application of the artificial neural network with multithreading within an inventory model under uncertainty and inflation. *Int. J. Fuzzy Syst.* **2022**, *24*, 2318–2332. [CrossRef]
2. Sarkar, B.; Takeyeva, D.; Guchhait, R.; Sarkar, M. Optimized radio-frequency identification system for different warehouse shapes. *Knowl.-Based Syst.* **2022**, *258*, 109811. [CrossRef]
3. Houssein, E.H.; Çelik, E.; Mahdy, M.A.; Ghoniem, R.M. Self-adaptive Equilibrium Optimizer for solving global, combinatorial, engineering, and Multi-Objective problems. *Expert Syst. Appl.* **2022**, *195*, 116552. [CrossRef]
4. Abualigah, L.; Ewees, A.A.; Al-qaness, M.A.; Elaziz, M.A.; Yousri, D.; Ibrahim, R.A.; Altalhi, M. Boosting arithmetic optimization algorithm by sine cosine algorithm and levy flight distribution for solving engineering optimization problems. *Neural Comput. Appl.* **2022**, *34*, 8823–8852. [CrossRef]
5. Al-Qaness, M.A.; Helmi, A.M.; Dahou, A.; Elaziz, M.A. The applications of metaheuristics for human activity recognition and fall detection using wearable sensors: A comprehensive analysis. *Biosensors* **2022**, *12*, 821. [CrossRef] [PubMed]
6. Al-Qaness, M.A.; Ewees, A.A.; Abualigah, L.; AlRassas, A.M.; Thanh, H.V.; Abd Elaziz, M. Evaluating the Applications of Dendritic Neuron Model with Metaheuristic Optimization Algorithms for Crude-Oil-Production Forecasting. *Entropy* **2022**, *24*, 1674. [CrossRef]
7. Whitley, D. A genetic algorithm tutorial. *Stat. Comput.* **1994**, *4*, 65–85. [CrossRef]
8. Dorigo, M.; Di Caro, G. Ant colony optimization: A new meta-heuristic. In Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), Washington, DC, USA, 6–9 July 1999; Volume 2, pp. 1470–1477.
9. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95-International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948.
10. Yang, X.S.; Slowik, A. Firefly algorithm. In *Swarm Intelligence Algorithms*; CRC Press: Boca Raton, FL, USA, 2020; pp. 163–174.
11. Mirjalili, S. SCA: A sine cosine algorithm for solving optimization problems. *Knowl.-Based Syst.* **2016**, *96*, 120–133. [CrossRef]
12. Mirjalili, S.; Mirjalili, S.M.; Hatamlou, A. Multi-verse optimizer: A nature-inspired algorithm for global optimization. *Neural Comput. Appl.* **2016**, *27*, 495–513. [CrossRef]
13. 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]

14. Faramarzi, A.; Heidarinejad, M.; Mirjalili, S.; Gandomi, A.H. Marine Predators Algorithm: A nature-inspired metaheuristic. *Expert Syst. Appl.* **2020**, *152*, 113377. [CrossRef]
15. Abualigah, L.; Yousri, D.; Abd Elaziz, M.; Ewees, A.A.; Al-Qaness, M.A.; Gandomi, A.H. Aquila optimizer: A novel meta-heuristic optimization algorithm. *Comput. Ind. Eng.* **2021**, *157*, 107250. [CrossRef]
16. Abualigah, L.; Diabat, A.; Mirjalili, S.; Abd Elaziz, M.; Gandomi, A.H. The arithmetic optimization algorithm. *Comput. Methods Appl. Mech. Eng.* **2021**, *376*, 113609. [CrossRef]
17. Abualigah, L.; Abd Elaziz, M.; 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]
18. Bo, Q.; Cheng, W.; Khishe, M. Evolving chimp optimization algorithm by weighted opposition-based technique and greedy search for multimodal engineering problems. *Appl. Soft Comput.* **2023**, *132*, 109869. [CrossRef]
19. Shen, Y.; Zhang, C.; Gharehchopogh, F.S.; Mirjalili, S. An Improved Whale Optimization Algorithm based on Multi-Population Evolution for Global Optimization and Engineering Design problems. *Expert Syst. Appl.* **2023**, *215*, 119269. [CrossRef]
20. Yang, X.; Wang, R.; Zhao, D.; Yu, F.; Huang, C.; Heidari, A.A.; Cai, Z.; Bourouis, S.; Algarni, A.D.; Chen, H. An adaptive quadratic interpolation and rounding mechanism sine cosine algorithm with application to constrained engineering optimization problems. *Expert Syst. Appl.* **2023**, *213*, 119041. [CrossRef]
21. Hong, J.; Shen, B.; Xue, J.; Pan, A. A vector-encirclement-model-based sparrow search algorithm for engineering optimization and numerical optimization problems. *Appl. Soft Comput.* **2022**, *131*, 109777. [CrossRef]
22. Zhao, X.; Fang, Y.; Ma, S.; Liu, Z. Multi-swarm improved moth–flame optimization algorithm with chaotic grouping and Gaussian mutation for solving engineering optimization problems. *Expert Syst. Appl.* **2022**, *204*, 117562. [CrossRef]
23. Peng, H.; Xiao, W.; Han, Y.; Jiang, A.; Xu, Z.; Li, M.; Wu, Z. Multi-strategy firefly algorithm with selective ensemble for complex engineering optimization problems. *Appl. Soft Comput.* **2022**, *120*, 108634. [CrossRef]
24. Zhang, Y.J.; Wang, Y.F.; Yan, Y.X.; Zhao, J.; Gao, Z.M. LMRAOA: An improved arithmetic optimization algorithm with multi-leader and high-speed jumping based on opposition-based learning solving engineering and numerical problems. *Alex. Eng. J.* **2022**, *61*, 12367–12403. [CrossRef]
25. Al-qaness, M.A.; Ewees, A.A.; Elaziz, M.A.; Samak, A.H. Wind Power Forecasting Using Optimized Dendritic Neural Model Based on Seagull Optimization Algorithm and Aquila Optimizer. *Energies* **2022**, *15*, 9261. [CrossRef]
26. Al-qaness, M.A.; Ewees, A.A.; Thanh, H.V.; AlRassas, A.M.; Dahou, A.; Elaziz, M.A. Predicting CO_2 trapping in deep saline aquifers using optimized long short-term memory. *Environ. Sci. Pollut. Res.* **2022**, 1–15. [CrossRef] [PubMed]
27. Al-qaness, M.A.; Ewees, A.A.; Thanh, H.V.; AlRassas, A.M.; Abd Elaziz, M. An optimized neuro-fuzzy system using advance nature-inspired Aquila and Salp swarm algorithms for smart predictive residual and solubility carbon trapping efficiency in underground storage formations. *J. Energy Storage* **2022**, *56*, 106150. [CrossRef]
28. Lin, Y.; Bian, X.Y.; Dong, Z.R. A discrete hybrid algorithm based on Differential Evolution and Cuckoo Search for optimizing the layout of ship pipe route. *Ocean. Eng.* **2022**, *261*, 112164. [CrossRef]
29. Zhong, X.; You, Z.; Cheng, P. A hybrid optimization algorithm and its application in flight trajectory prediction. *Expert Syst. Appl.* **2023**, *213*, 119082. [CrossRef]
30. Balyan, A.K.; Ahuja, S.; Lilhore, U.K.; Sharma, S.K.; Manoharan, P.; Algarni, A.D.; Elmannai, H.; Raahemifar, K. A hybrid intrusion detection model using ega-pso and improved random forest method. *Sensors* **2022**, *22*, 5986. [CrossRef]
31. Bu, S.J.; Kang, H.B.; Cho, S.B. Ensemble of Deep Convolutional Learning Classifier System Based on Genetic Algorithm for Database Intrusion Detection. *Electronics* **2022**, *11*, 745. [CrossRef]
32. Fatani, A.; Dahou, A.; Al-Qaness, M.A.; Lu, S.; Abd Elaziz, M. Advanced feature extraction and selection approach using deep learning and Aquila optimizer for IoT intrusion detection system. *Sensors* **2022**, *22*, 140. [CrossRef]
33. Mansour, R.F. Blockchain assisted clustering with Intrusion Detection System for Industrial Internet of Things environment. *Expert Syst. Appl.* **2022**, *207*, 117995. [CrossRef]
34. Dahou, A.; Abd Elaziz, M.; Chelloug, S.A.; Awadallah, M.A.; Al-Betar, M.A.; Al-qaness, M.A.; Forestiero, A. Intrusion Detection System for IoT Based on Deep Learning and Modified Reptile Search Algorithm. *Comput. Intell. Neurosci.* **2022**, *2022*, 6473507 . [CrossRef]
35. Elgamal, Z.; Sabri, A.Q.M.; Tubishat, M.; Tbaishat, D.; Makhadmeh, S.N.; Alomari, O.A. Improved Reptile Search Optimization Algorithm using Chaotic map and Simulated Annealing for Feature Selection in Medical Filed. *IEEE Access* **2022**, *10*, 51428–51446. [CrossRef]
36. Chauhan, S.; Vashishtha, G.; Kumar, A. Approximating parameters of photovoltaic models using an amended reptile search algorithm. *J. Ambient. Intell. Humaniz. Comput.* **2022**, 1–16. [CrossRef]
37. Połap, D.; Woźniak, M. Red fox optimization algorithm. *Expert Syst. Appl.* **2021**, *166*, 114107. [CrossRef]
38. Khorami, E.; Mahdi Babaei, F.; Azadeh, A. Optimal diagnosis of COVID-19 based on convolutional neural network and red Fox optimization algorithm. *Comput. Intell. Neurosci.* **2021**, *2021*, 4454507. [CrossRef] [PubMed]
39. Natarajan, R.; Megharaj, G.; Marchewka, A.; Divakarachari, P.B.; Hans, M.R. Energy and Distance Based Multi-Objective Red Fox Optimization Algorithm in Wireless Sensor Network. *Sensors* **2022**, *22*, 3761. [CrossRef] [PubMed]
40. Zaborski, M.; Woźniak, M.; Mańdziuk, J. Multidimensional Red Fox meta-heuristic for complex optimization. *Appl. Soft Comput.* **2022**, *131*, 109774. [CrossRef]
41. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]

42. Krohling, R.A.; dos Santos Coelho, L. Coevolutionary particle swarm optimization using Gaussian distribution for solving constrained optimization problems. *IEEE Trans. Syst. Man Cybern. Part B (Cybern.)* **2006**, *36*, 1407–1416. [CrossRef]
43. Awad, N.H.; Ali, M.Z.; Suganthan, P.N. Ensemble sinusoidal differential covariance matrix adaptation with Euclidean neighborhood for solving CEC2017 benchmark problems. In Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC), San Sebastián, Spain, 5–8 June 2017; pp. 372–379.
44. Huang, F.z.; Wang, L.; He, Q. An effective co-evolutionary differential evolution for constrained optimization. *Appl. Math. Comput.* **2007**, *186*, 340–356. [CrossRef]
45. Ragsdell, K.; Phillips, D. Optimal design of a class of welded structures using geometric programming. *J. Manuf. Sci. Eng.* **1976**, *38*, 1021–1025. [CrossRef]
46. Rashedi, E.; Nezamabadi-Pour, H.; Saryazdi, S. GSA: A gravitational search algorithm. *Inf. Sci.* **2009**, *179*, 2232–2248. [CrossRef]
47. Deb, K. Optimal design of a welded beam via genetic algorithms. *AIAA J.* **1991**, *29*, 2013–2015. [CrossRef]
48. Lee, K.S.; Geem, Z.W. A new meta-heuristic algorithm for continuous engineering optimization: Harmony search theory and practice. *Comput. Methods Appl. Mech. Eng.* **2005**, *194*, 3902–3933. [CrossRef]
49. Mohamed, A.W.; Hadi, A.A.; Fattouh, A.M.; Jambi, K.M. LSHADE with semi-parameter adaptation hybrid with CMA-ES for solving CEC 2017 benchmark problems. In Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC), San Sebastián, Spain, 5–8 June 2017; pp. 145–152.
50. Ewees, A.A.; Abd Elaziz, M.; Houssein, E.H. Improved grasshopper optimization algorithm using opposition-based learning. *Expert Syst. Appl.* **2018**, *112*, 156–172. [CrossRef]
51. Kaveh, A.; Khayatazad, M. A new meta-heuristic method: Ray optimization. *Comput. Struct.* **2012**, *112*, 283–294. [CrossRef]
52. Coello, C.A.C. Use of a self-adaptive penalty approach for engineering optimization problems. *Comput. Ind.* **2000**, *41*, 113–127. [CrossRef]
53. Mezura-Montes, E.; Coello, C.A.C. An empirical study about the usefulness of evolution strategies to solve constrained optimization problems. *Int. J. Gen. Syst.* **2008**, *37*, 443–473. [CrossRef]
54. Ray, T.; Saini, P. Engineering design optimization using a swarm with an intelligent information sharing among individuals. *Eng. Optim.* **2001**, *33*, 735–748. [CrossRef]
55. Belegundu, A.D.; Arora, J.S. A study of mathematical programmingmethods for structural optimization. Part II: Numerical results. *Int. J. Numer. Methods Eng.* **1985**, *21*, 1601–1623. [CrossRef]
56. He, Q.; Wang, L. A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization. *Appl. Math. Comput.* **2007**, *186*, 1407–1422. [CrossRef]
57. Kaveh, A.; Talatahari, S. An improved ant colony optimization for constrained engineering design problems. *Eng. Comput.* **2010**, *27*, 155–182. [CrossRef]
58. Liu, H.; Cai, Z.; Wang, Y. Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. *Appl. Soft Comput.* **2010**, *10*, 629–640. [CrossRef]
59. Ray, T.; Liew, K.M. Society and civilization: An optimization algorithm based on the simulation of social behavior. *IEEE Trans. Evol. Comput.* **2003**, *7*, 386–396. [CrossRef]
60. Mirjalili, S. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowl.-Based Syst.* **2015**, *89*, 228–249. [CrossRef]
61. Zhang, M.; Luo, W.; Wang, X. Differential evolution with dynamic stochastic selection for constrained optimization. *Inf. Sci.* **2008**, *178*, 3043–3074. [CrossRef]
62. 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]
63. Sadollah, A.; Bahreininejad, A.; Eskandar, H.; Hamdi, M. Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems. *Appl. Soft Comput.* **2013**, *13*, 2592–2612. [CrossRef]
64. 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]
65. YILDIRIM, A.E.; KARCI, A. Application of Three Bar Truss Problem among Engineering Design Optimization Problems using Artificial Atom Algorithm. In Proceedings of the 2018 International Conference on Artificial Intelligence and Data Processing (IDAP), Malatya, Turkey, 28–30 September 2018; pp. 1–5.
66. Saremi, S.; Mirjalili, S.; Lewis, A. Grasshopper optimisation algorithm: Theory and application. *Adv. Eng. Softw.* **2017**, *105*, 30–47. [CrossRef]
67. Siddall, J.N. *Analytical Decision-Making in Engineering Design*; Prentice Hall: Upper Saddle River, NJ, USA, 1972.
68. Akhtar, S.; Tai, K.; Ray, T. A socio-behavioral simulation model for engineering design optimization. *Eng. Optim.* **2002**, *34*, 341–354. [CrossRef]
69. SARUHAN, H.; UYGUR, İ. Design optimization of mechanical systems using genetic algorithms. *Sak. Üniv. Fen Bilim. Enst. Derg.* **2003**, *7*, 77–84.
70. Geem, Z.W.; Kim, J.H.; Loganathan, G.V. A new heuristic optimization algorithm: Harmony search. *Simulation* **2001**, *76*, 60–68. [CrossRef]

71. Mezura-Montes, E.; Coello, C.C.; Landa-Becerra, R. Engineering optimization using simple evolutionary algorithm. In Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence, Sacramento, CA, USA, 5 November 2003; pp. 149–156.
72. Lu, S.; Kim, H.M. A regularized inexact penalty decomposition algorithm for multidisciplinary design optimization problems with complementarity constraints. *J. Mech. Des.* **2010**, *132*, 410051–4100512. [CrossRef]
73. Stephen, S.; Christu, D.; David, D.C.N. Design Optimization of Weight of Speed Reducer Problem Through Matlab and Simulation Using Ansys. *Int. J. Mech. Eng. Technol. (IJMET)* **2018**, *9*, 339–349.
74. Baykasoğlu, A.; Ozsoydan, F.B. Adaptive firefly algorithm with chaos for mechanical design optimization problems. *Appl. Soft Comput.* **2015**, *36*, 152–164. [CrossRef]
75. Kamboj, V.K.; Nandi, A.; Bhadoria, A.; Sehgal, S. An intensify Harris Hawks optimizer for numerical and engineering optimization problems. *Appl. Soft Comput.* **2020**, *89*, 106018. [CrossRef]
76. Rao, R.V.; Savsani, V.J.; Vakharia, D. Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems. *Comput.-Aided Des.* **2011**, *43*, 303–315. [CrossRef]
77. Deb, K.; Srinivasan, A. Innovization: Discovery of innovative design principles through multiobjective evolutionary optimization. In *Multiobjective Problem Solving from Nature*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 243–262.
78. 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*, 151–166. [CrossRef]
79. Sayed, G.I.; Darwish, A.; Hassanien, A.E. A new chaotic multi-verse optimization algorithm for solving engineering optimization problems. *J. Exp. Theor. Artif. Intell.* **2018**, *30*, 293–317. [CrossRef]
80. Bhesdadiya, R.; Trivedi, I.N.; Jangir, P.; Jangir, N. Moth-flame optimizer method for solving constrained engineering optimization problems. In *Advances in Computer and Computational Sciences*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 61–68.
81. Zarpelão, B.B.; Miani, R.S.; Kawakani, C.T.; de Alvarenga, S.C. A survey of intrusion detection in Internet of Things. *J. Netw. Comput. Appl.* **2017**, *84*, 25–37. [CrossRef]
82. Abomhara, M.; Køien, G.M. Security and privacy in the Internet of Things: Current status and open issues. In Proceedings of the 2014 international conference on privacy and security in mobile systems (PRISMS), Aalborg, Denmark, 11–14 May 2014; pp. 1–8.
83. Fatani, A.; Abd Elaziz, M.; Dahou, A.; Al-Qaness, M.A.; Lu, S. IoT intrusion detection system using deep learning and enhanced transient search optimization. *IEEE Access* **2021**, *9*, 123448–123464. [CrossRef]
84. Yang, X.S. A new metaheuristic bat-inspired algorithm. In *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 65–74.
85. Koroniotis, N.; Moustafa, N.; Sitnikova, E.; Turnbull, B. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. *Future Gener. Comput. Syst.* **2019**, *100*, 779–796. [CrossRef]
86. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp* **2018**, *1*, 108–116.