mathematics

MDPI

*Article*

# Chaotic Search-Based Salp Swarm Algorithm for Dealing with System of Nonlinear Equations and Power System Applications

**Mohammed A. El-Shorbagy [1,2,*], Islam M. Eldesoky [2,3], Mohamady M. Basyouni [2], Islam Nassar [2] and Adel M. El-Refaey [4]**

[1] Department of Mathematics, College of Science and Humanities in Al-Kharj, Prince Sattam Bin Abdulaziz University, Al-Kharj 11942, Saudi Arabia
[2] Department of Basic Engineering Science, Faculty of Engineering, Menoufia University, Shebin El-Kom 32511, Egypt; islam_eldesoky@sh-eng.menofia.edu.eg (I.M.E.); mohamedi.basyouni@sh-eng.menofia.edu.eg (M.M.B.); islam-nassar@sh-eng.menofia.edu.eg (I.N.)
[3] Basic Sciences Department, Elmenofia Higher Institute of Engineering and Technology, El Bagour 32821, Egypt
[4] Basic and Applied Science Department, College of Engineering and Technology, Arab Academy for Science, Technology and Maritime Transport, Smart Village Campus, Giza 12577, Egypt; adel_elrefaey@aast.edu
* Correspondence: ma.hassan@psau.edu.sa

**Abstract:** The system of nonlinear equations (SNLEs) is one of the eminent problems in science and engineering, and it is still open to research. A new hybrid intelligent algorithm is presented in this research to solve SNLEs. It is a composite of the salp swarm algorithm (SSA) and chaotic search technique (CST). The proposed methodology is named chaotic salp swarm algorithm (CSSA). CSSA is designed as an optimization process, whereby feasible and infeasible solutions are updated to move closer to the optimum value. The use of this hybrid intelligent methodology aims to improve performance, increase solution versatility, avoid the local optima trap, speed up convergence and optimize the search process. Firstly, SNLEs are transformed into an optimization problem. Secondly, CSSA is used to solve this optimization problem: SSA is used to update the feasible solutions, whereas the infeasible solutions are updated by CST. One of the most significant advantages of the suggested technique is that it does not ignore infeasible solutions that are updated, because these solutions are often extremely near to the optimal solution, resulting in increased search effectiveness and effective exploration and exploitation. The algorithm's mathematical model is presented in detail. Finally, the proposed approach is assessed with several benchmark problems and real-world applications. Simulation results show that the proposed CSSA is competitive and better in comparison to others, which illustrates the effectiveness of the proposed algorithm. In addition, a statistical analysis by the Wilcoxon rankings test between CSSA and the other comparison methods shows that all *p*-values are less than 0.05, and CSSA achieves negative ranks' sum values ($R^-$) much better than the positive ranks' sum values ($R^+$) in all benchmark problems. In addition, the results have high precision and show good agreement in comparison with similar methods, and they further proved the ability of CSSA to solve real-world applications.

**Keywords:** system of nonlinear equations; swarm intelligence; salp swarm algorithm; chaotic search technique; hybrid intelligent algorithm; optimization

**MSC:** 68UXX; 68WXX; 90BXX; 90CXX

**Citation:** El-Shorbagy, M.A.; Eldesoky, I.M.; Basyouni, M.M.; Nassar, I.; El-Refaey, A.M. Chaotic Search-Based Salp Swarm Algorithm for Dealing with System of Nonlinear Equations and Power System Applications. *Mathematics* **2022**, *10*, 1368. https://doi.org/10.3390/math10091368

Academic Editor: Dragos Sburlan

Received: 10 March 2022
Accepted: 16 April 2022
Published: 19 April 2022

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



## 1. Introduction

In nature, most systems are nonlinear, from the equation of motion of a simple pendulum to more complex systems such as fluid flow. The system of nonlinear equations appears widely in many applications in the real world. In mathematical biology, there are

many models, such as population growth and molecular evolution [1]. There are also many fields in medicine in which the study of nonlinearity is required, such as understanding the working of neurons and the study of plasma [2]. Further, in geometric computations, the SNLE arises in many applications, such as minimum distance, intersections, the creation of centenary curves, and when solving boundary or initial value problems in partial or ordinary differential equations [3]. In addition, this system is widely applicable in engineering, computing, physics, finance, astronomy, robotics, etc. Therefore, solving this system is essential, but it is very difficult. Furthermore, as the number of equations grows, complications increase. Many books and researchers study these problems, such as the book *Nonlinear Optimization with Engineering Applications* [4].

There are two techniques to solving the SNLE: classic and advanced. Classic approaches are those such as bisection, Newton's approach and secant methods, etc. [5,6]. The majority of these methods rely on Newton's approach. Newton's approach would be a good choice if the system of nonlinear equations has analytical derivatives for all variables. Classic methods have many disadvantages, such as the need for good starting guesses, function continuity and differentiability and many computations. Moreover, advanced algorithms take their methodology from animal behaviors such as fish schools, bird flocks and bug swarms. They are also regarded as computational models that simulate natural swarm systems. These algorithms became very popular because of their gradient-free mechanism and their flexibility, and they avoid local solutions. They are better and easier than classic methods and provide us with optimal multiple solutions. These techniques made improvements in results. For examples of advanced algorithms, there are the firefly algorithm (FA) (this approach is an optimization approach inspired by the flashing patterns and intelligent behavior of fireflies and their characteristics [7,8]); ant colony optimization (ACO) (which follows the behavior of real ants who communicate through pheromones [9]); bacterial foraging (BF) [10]; the krill herd algorithm (KHA) [11]; cat swarm optimization (CSO) [12]; the artificial bee colony (ABC) [13]; the sine cosine algorithm (SCA) [14]; particle swarm optimization (PSO) (which is a swarm-based intelligence algorithm for optimization that is dependent on the social behavior of birds, insects and fish [15–17]); the genetic algorithm (GA) (which is a methodology for solving optimization issues that are dependent on genetics and natural selection, the processes that lead to biological evolution [18,19]); the grasshopper optimization algorithm (GOA) [20], etc.

Advanced approaches suffer from some limitations. Therefore, many researchers devoted themselves to developing and improving them by combining two approaches to improve the quality of solutions and benefit from the advantages of some algorithms and avoid any deficiencies such as hybrid ACO [21], hybrid PSO [22], hybrid GA [23,24], hybrid SCA [25], hybrid CSO [26], the hybrid whale optimization algorithm (WOA) [27], etc. Recently, advanced algorithms have become the most widely used method for solving SNLEs [28–32].

The chaotic search technique (CST) is one of the common mathematical methods that proved its efficacy in enhancing the performance of many algorithms. This technique is well known as the simulation of nonlinear systems' dynamic behavior [33]. CST has gained great interest, and it has been used in a variety of fields such as chaos control [34] and optimization research [35]. In [36], the authors used a GA and a logistic chaotic map to encrypt pictures, and the results show that their strategy has a high level of resistance against brute force and statistical invasions. In [37], the authors used a combination between PSO and the chaos optimization algorithm (COA), and the results reveal that embedding a logistic chaotic map improves PSO efficiency in terms of time oscillation and convergence rate. In [38], the authors' methodology depends on the combination of chaotic maps, in which employed biogeography-based optimization (BBO) with ten maps of chaos techniques were used, and the findings show that the improved chaotic BBO can increase BBO's performance in terms of exploitation and exploration. It has been found that hybridization with chaotic maps reduces the time of computations and improves convergence to the global solution. It has also been demonstrated that these chaotic maps

(particularly logistic maps) have enhanced solution quality, which has improved the global searching capability by avoiding local solutions in all circumstances. Therefore, in our proposed technique, we used the logistic map because it was found to be the best chaotic map.

The majority of real-world optimization problems have constraints, and constraint handling has long been a focus of research. Constraint handling refers to all strategies used that deal with infeasible solutions, either during problem formulation or during execution. Prior techniques favored viable over non-feasible solutions; they assumed that selecting just feasible solutions led to a better search path within the feasible region. Current research, on the other hand, has revealed certain flaws in such strategies; for example, if certain selected infeasible solutions that are always near the search space boundary (i.e., good infeasible solutions) were allowed to exist in the population, the search trajectory could be significantly shorter, resulting in significant savings in the number of function evaluations [39]. In our suggested technique, infeasible solutions are not ignored, but rather updated, resulting in improved search effectiveness, as well as balancing between exploration and exploitation.

Salp swarm algorithm (SSA) is a recent technique of optimization that has been utilized to handle a variety of optimization issues [40]. It is a mathematical model approach to finding the optimal solution that imitates the swarming behavior of salps in the sea. In SSA, a set of solutions is generated at random, which is modified by salp swarm equations by shifting the solutions outside or inside the solution space to generate new solutions. SSA is suitable for many kinds of optimization problems in different fields, and it has good convergence acceleration. It has important characteristics such as adaptability and robustness [40]. Although SSA is a new optimization method, it has some limitations that degrade its performance characteristics. For example, the leading strategy in SSA depends on a set of random parameters that could obstruct the variety of solutions and cause local optima to become stuck. Additionally, SSA lacks an effective strategy to improve the quality of solutions during each generation, as it can generate poor-quality solutions as generations evolve. Furthermore, according to our information, no endeavors to insert SSA to solve SNLEs and their implementations in power systems have been reported in the literature.

In this paper, an intelligent approach called chaotic salp swarm algorithm (CSSA) is presented by combining the SSA and CST to solve SNLEs. CSSA hybridizes the salp swarm algorithm's features and the chaotic search technique. The essential objective of the new approach is to fine-tune the solution area for the destination solution; the solution area is decreased progressively with an augmented number of iterations. SSA is used to update the feasible solutions, and the infeasible solutions are updated by CST. The CST is incorporated with concentrations on using the infeasible solution to enhance the efficacy of the solutions, increasing the variety and intensification of the solution, reaching the optimum solution and avoiding snaring inside the local optimum. Our proposed approach has been assessed with several benchmarks and two distinctive electrical applications. The results have high precision and show good agreement in comparison with similar methods.

The major contributions of this paper include:

(1) Presenting a new hybrid algorithm, CSSA, to solve SNLEs.
(2) Introducing a variety of solutions effectively, and preventing CSSA from dropping into the local optima by using CST.
(3) Providing a fine balance between exploration and exploitation trends in CSSA by combining SSA's exploration and CST's exploitation abilities.
(4) Applying CST, with a focus on using the infeasible solution, has helped improve performance, avoid the local optima trap, speed up convergence and optimize the search process.
(5) Enhancing the quality of solutions and accelerating convergence to the best results with the hybridization between SSA with CST.

(6) Testing CSSA by using several benchmark problems, as well as two real-world power system applications.
(7) Using the Wilcoxon test to evaluate the significance of the CSSA results.
(8) Displaying that CSSA is competitive and better than other optimization algorithms through statistical analysis and computational results.

The structure of this research paper is as follows: in Section 2, the system of nonlinear equations is shown. Section 3 presents the preliminary information about the main characteristics of SSA and CST. Section 4 explains the proposed approach. Numerical studies are provided in Section 5. Section 6 demonstrates how CSSA may be used to tackle power system applications. Finally, in Section 7, a conclusion is given.

## 2. System of Nonlinear Equations (SNLEs)

The definition of SNLEs is:

$$\begin{gathered} f_1(y_1,\ y_2, \ldots, y_n) = 0 \\ f_2(y_1,\ y_2, \ldots, y_n) = 0 \\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots \\ f_m(y_1,\ y_2, \ldots, y_n) = 0\ ; \end{gathered} \tag{1}$$

where, $f_i\ , i = 1, 2, \ldots, m$ is a nonlinear equation system, and $y = (y_1,\ y_2, \ldots, y_n)$ are the unknown variables. For SNLEs, the solution requires finding a solution that makes each of the functions above $f_i,\ i = 1, 2, \ldots, m$ equal to zero.

*Converting the SNLEs into an Optimization Problem*

Many methods exist for converting SNLEs $f_i,\ i = 1, 2, \ldots, m$ into a problem of optimization. In Nie [41,42], the first method is used, in which the SNLEs $f_i,\ i = 1, 2, \ldots, m$ are recast as a restricted optimization problem. The original equations have been divided into two sets, S1 and S2; S1 refers to the equations that make up the objective function, and S2 refers to the equations that serve as equality constraints. Furthermore, at each stage of the optimization process, the two sets are changed. The constrained problem of optimization can then be expressed as follows:

$$\begin{gathered} F(y) = min \sum_{i \in s_1} f_i^2(y) \\ \text{Subject to: } f_i(y) = 0,\ \ j \in s_2 \end{gathered} \tag{2}$$

In the second method, the SNLE is recast as a multi-objective problem of optimization [43,44] as follows:

$$\begin{cases} f_1(y) = 0 \\ f_2(y) = 0 \\ \vdots \\ f_m(y) = 0 \end{cases} \longrightarrow \begin{cases} F_1 = min(|f_1(y)|) = 0 \\ F_2 = min(|f_2(y)|) = 0 \\ \vdots \\ F_m = min(|f_m(y)|) = 0 \end{cases} \tag{3}$$

The purpose of these functions is to reduce the absolute value difference between the equations' right and left sides. This technique does not need any additional constraints and is capable of finding solutions even for a wide range of SNLEs.

Finally, the SNLEs are turned into a restricted optimization problem by adding the left hand side of all equations and applying the absolute value function [45]:

$$F(y) = abs\big(f_1(y) + f_2(y) + \cdots + f_m(y)\big) \tag{4}$$

$$subject\ to: \begin{cases} f_1(y) = 0 \\ f_2(y) = 0 \\ \vdots \\ f_m(y) = 0 \end{cases}$$

This objective function's description differs greatly from Abraham's and Grosan's [43]. If whole equations of the system are equal to zero ($f_i = 0 \ \ \forall i = 1, \dots, m$), Equation (4)'s objective function has a global minimum.

## 3. Preliminaries

In this section, we provide a brief overview of both the salp swarm algorithm (SSA) and the chaotic search technique (CST).

### *3.1. The Main Characteristics of SSA*

SSA is one of the most recent meta-heuristic algorithms, presented by Mirjalili [40] in 2017. The swarming behavior of salps in the ocean inspired this algorithm. Salps are members of the Salpidae family. Their body is diaphanous and barrel-shaped. They are very similar to jellyfish in terms of their tissues and movement. They move by contracting and changing postures by pushing water through their jellied bodies. Salps usually live in groups, and they often make a swarm, known as a chain of salps. The major cause for their actions has remained unknown till now. It is thought that this behavior reaches optimal motion through speedy regulated shifts and foraging. The salps' population consists of two groups. These groups are leaders and followers. The chain's leader is in front, and followers follow the leader to find a food source.

The position of the salps is defined in $m$-dimensions of a problem's solution area, where $m$ denotes the number of variables in the problem. Therefore, the position of all salps is stored in a two-dimensional matrix called $x$. The swarm's target is a food source in the search space, which is called $F$. The leader should update his location, frequently referring to the later equation:

$$x_i^j = \begin{cases} F_i + k_1\big((ub_i - lb_i)k_2 + lb_i\big), & k_3 \geq 0 \\ F_i - k_1\big((ub_i - lb_i)k_2 + lb_i\big), & k_3 < 0 \end{cases}; \quad (5)$$

where $j = 1$, $x_i^j \ \forall\, i = 1, 2, \dots, m$ refers to the position of the first salp (leader) in $i^{\text{th}}$ dimension, $lb_i$ is the lower limit at $i^{\text{th}}$ dimension, $ub_i$ is the upper limit at $i^{\text{th}}$ dimension, $F_i$ is the position of the food source in $i^{\text{th}}$ dimension and $k_1$, $k_2$ and $k_3$ are values in range [0, 1], which generate randomly. In SSA, coefficient $k_1$ is the most important factor, as it decreases as the number of iterations increases, resulting in high exploration in the early phases of the optimization and high exploitation in the later stages, resulting in a balanced optimization. It is defined as follows:

$$k_1 = 2e^{-\left(\frac{t}{T_{max}}\right)}; \quad (6)$$

where $t$ is the present iteration, $T_{max}$ is the maximum number of iterations and $k_3$ is in charge of determining whether the following position should be toward $-\infty$ or $+\infty$, as well as the step size. Newton's law of motion is used to adjust the location of the followers, as seen in the equation below:

$$x_i^j = 0.5\, at^2 + v_o t; \quad (7)$$

where $j \geq 2$, $x_i^j \ \forall\, i = 1, 2, \dots, m$ indicates the position of the $j^{\text{th}}$ follower salp in the $i^{\text{th}}$dimension, $v_o$ is the start speed; $a = \frac{v_{final}}{t}$, where $v_{final} = \frac{x - x_0}{t}$, $v_{final}$ is the final speed motion of the salp, and $t$ is time. The time in the optimization process is the current iteration; the iteration discrepancy is equivalent to one. Assuming that $v_o = 0$, we can define this equation as follows:

$$x_i^{j,t+1} = 0.5\left(x_i^{j-1,t+1} + x_i^{j,t}\right); \quad (8)$$

The salp chains can be modeled using Equations (5) and (8). Figure 1 shows the main steps of SSA.

**Initialize** $lb, ub, T_{max}, j$
**Initialize** $j$ salps ($x_i \forall i = 1,2, \ldots, m$) randomly.
**while** $t \leq T_{max}$
  **Calculate** the fitness of each search agent (salp)
  **Determine** the best search agent $F$
  **Update** $k_1$ using Equation (6)
    **For** each salp ($x_i$)
      **If** ($j == 1$) then
        **Update** the position of salp using Equation (5)
      **Else**
        **Update** the position of salp using Equation (8)
      **End If**
    **End For**
  $\boldsymbol{t = t + 1}$
**End**
**Return** $F$.

**Figure 1.** The pseudo-code of the general SSA.

*3.2. Chaotic Search Technique (CST)*

CST is a section of mathematics that transacts with nonlinear dynamical systems, which follows deterministic laws but appears unpredictable and random. CST is more appealing to many systems in different fields, such as physics, robotics, microbiology and computer science [46]. There are three main characteristics of chaos theory: sensitivity to initial conditions, quasi-stochastic and ergodicity. Sensitivity to initial conditions characteristics means that any little alterations in the initial starting points make a difference in behavior. Quasi-stochastic means the capacity to substitute random variables with values from chaotic maps. An ergodic characteristic is defined as chaotic variables' capacity to search non-frequently for all states in a given range. When all of these qualities are combined, meta-heuristic optimization techniques can considerably improve their performance. CST minimizes local optima and increases convergence [47]. In this section, we present several well-known chaotic maps from the literature [48–56].

– Chebyshev map: the Chebyshev map [48] is formulated as:

$$x_{j+1} = cos\left(j\ cos^{-1}\left(x_j\right)\right). \quad (9)$$

– Singer map: Singer's chaotic one-dimensional map [49] is defined as the following equation:

$$x_{j+1} = \mu\left(7.86\, x_j - 23.31\, x_j^2 + 28.75\, x_j^3 - 13.302875\, x_j^4\right); \quad (10)$$

where $\mu \in (0.9, 1.08)$.

– Sinusoidal map: the following equation is used to create a sinusoidal map [50]:

$$x_{j+1} = a\, x_j^2\ sin\left(\pi\, x_j\right); \quad (11)$$

where $a = 2.3$.

– Piecewise map: a piecewise map [51] can be defined as follows:

$$x_{j+1} = \begin{cases} \dfrac{x_j}{p}, & 0 < x_j < p \\ \dfrac{(x_j - p)}{(0.5 - p)}, & p \le x_j < 0.5 \\ \dfrac{(1 - p - x_j)}{(0.5 - p)}, & 0.5 \le x_j < 1 - p \\ \dfrac{1 - x_j}{p}, & 1 - p < x_j < 1 \end{cases}; \quad (12)$$

where $p \in (0, 0.5)$ and $x \in (0, 1)$.

- Sine map: the equation for a sine map [52] is as follows:

$$x_{j+1} = 4a\ sin(\pi\, x_j); \quad (13)$$

where $0 < a \le 4$.

- Circle map: a circle map [53] is described as:

$$x_{j+1} = x_j + h - (k - 2\pi)\, sin(2\pi\, x_j)\, mod(1) \quad (14)$$

where $k = 0.5$ and $h = 0.2$.

- Logistic map: a logistic map [54] shows how a simple deterministic system can produce complex behavior without the use of a random sequence. It depends on a simple polynomial equation that explains biological population dynamics [50].

$$x_{j+1} = a\, x_j(1 - x_j), \quad a = 4. \quad (15)$$

- Intermittency map: the intermittency map [55] can be formulated as:

$$x_{j+1} = \begin{cases} \varepsilon + x_j + k\, x_j^n & if \quad 0 < x_j \le p \\ \dfrac{(x_j - p)}{1 - p}, & else\ if \quad p < x_j < 1 \end{cases}; \quad (16)$$

where $k = \frac{1-\varepsilon-p}{p^2}$, $n = 2$ and $\varepsilon$ is very close to zero.

- Gauss map/Mouse map: the Gauss map [56] is given by the Gaussian function:

$$x_{j+1} = e^{-A\, x_j^2} + B; \quad (17)$$

where $A$ and $B$ are real parameters.

- Iterative map: we can formulate the iterative chaotic map [51] as follows:

$$x_{j+1} = sin\left(\frac{h\pi}{x_j}\right) \quad (18)$$

where $h \in (0,1)$.

- Liebovitch map: we can formulate the Liebovitch chaotic map [48] as follows:

$$x_{j+1} = \begin{cases} A\, x_j & 0 < x_j \le h_1 \\ \frac{h_2 - x_j}{h_2 - h_1} & h_1 < x_j \le h_2 \\ 1 - B(1 - x_j) & h_2 < x_j \le 1 \end{cases}; \quad (19)$$

where $h_1, h_2 \in (0, 1), h_1 < h_2$, $A = \frac{h_2(1-(h_2-h_1))}{h_1}$ and $B = \frac{((h_2-1)-h_1(h_2-h_1))}{h_2-1}$.

- Tent map: the tent map [57] is represented as:

$$x_{j+1} = \begin{cases} \frac{x_j}{0.07}, & x_j < 0.7 \\ \frac{10(1-x_j)}{3}, & x_j \ge 0.7 \end{cases}. \quad (20)$$

## 4. Chaotic Salp Swarm Algorithm (CSSA)

This section explains our proposed algorithm, CSSA, which is a composite of SSA and CST to solve nonlinear equations system. SSA updates its agents about the potential solution according to Equations (5) and (8). Although it is a highly effective algorithm, in some states, the SSA remains susceptible to convergence in local optima. As a result, SSA's difficulties with unripe convergence and poor results may persist. In some cases, the basic SSA is not able to move seamlessly from the phases of exploration to the phases of exploitation. In this respect, the SSA needs enhanced agents to overcome issues and more exploration/exploitation ability. To alleviate the aforementioned problems, CST is presented as a local search phase. The combination of SSA with CST and using infeasible solutions achieves better searching quality, improves the performance and increases the variety of solutions, while avoiding entrapment in local optima and accelerating the optimum searching operation and the convergence features. Our proposed algorithm operates in two phases. Firstly, we transformed the SNLEs into an optimization problem. Secondly, we used CSSA to solve this optimization problem. SSA was used to update the feasible solutions, and the infeasible solutions were updated by the CST. When we allowed some infeasible solutions to exist in the population, the search trajectory became significantly shorter, resulting in significant savings in the number of function evaluations. We used the logistic map to generate the chaos sequence, because it is more convenient to use [37,38,58].

### *4.1. Steps of the Proposed Algorithm*

The following are detailed descriptions of our algorithm's steps:

1. Initialize the parameters. Search agents (all salps' number) are set as $j = L_f + L_{in}$, where $(L_f)$ is the number of salps in the feasible list, and $(L_{in})$ is the number of salps in the infeasible list. The maximum number of iterations $(T)$ are set up to be utilized as the algorithm's end conditions, using $t = 0$ as the iteration counter, lower boundary $(Lb)$, upper boundary $(ub)$ and the number of variables $(m)$.
2. Initialize the salps' positions. In the first generation, the salps' positions are initialized randomly to fulfill the solution space $S$ (each variable's upper and lower limits), using the following equation.

$$\text{Each individual's position}|_i = Lb + (Ub - Lb) \times rand; \quad (21)$$

where $i = 1, 2, \ldots, m$, and random numbers dispersed uniformly throughout the range [0, 1] are known as rand.

3. Evaluate the salps. According to the needed objective function, every salp is estimated based on the quality of its place, as given in Equation (2), where the best solution (destination) has been documented thus far.
4. Update the salps' positions. Update salps' positions in the feasible list according to Equation (5) and salps' positions in the infeasible list with CST according to the following equations:

$$Z_{q+1} = 4Z_q\left(1 - Z_q\right), \quad (22)$$

$$X_u = Z_{q+1} \times X_{inf} + \left(1 - Z_{q+1}\right) \times X_b \quad (23)$$

where $Z_0 \in (0,1), Z_0 \notin (0.25,0.5,0.75,1)$, $q = 1, 2, \ldots, L_{in}$, $X_u$ is the updated salps' position, $X_{inf}$ is the position of the infeasible salps and $X_b$ is the best position of feasible salps. Figure 2 shows the updated infeasible solution using the CST.

5. Check the feasibility of solutions. Maybe after updating positions, the feasibility of solutions is changed, so we should check the salps' positions; if the salp exists in the search space, the salp is then considered feasible and is added to the feasible list. Then, we compute the fitness values of the salps. In this case, if the salp is infeasible, it should be added to the infeasible list, and set the fitness value of infeasible as the best value.

6. Evaluation of the feasible solutions. Feasible solutions are evaluated.
7. Update the best value. Compare the present fitness value with the updated value. If the present value is better than the updated value, then set the present value as the best value. If the updated value is better than the present value, then set the updated value as the best value; then, the corresponding position is the best.
8. Termination criteria. Set $t = t + 1$. If the iterations' number is greater than the maximum ($T$), go to Step 9; otherwise, go to Step 4.
9. Output the results. Output the best position and the best fitness value.

The pseudo-code for CSSA is presented in Figure 3, and Figure 4 shows the flow diagram.

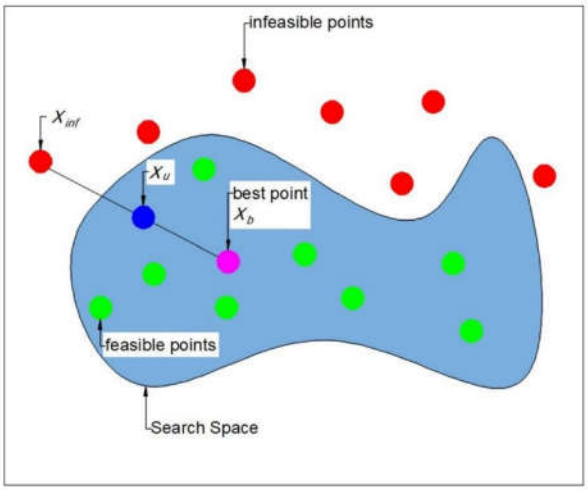


**Figure 2.** Updating infeasible solutions using the CST.

**Initialize** $lb, ub, T_{max}, j$
**Initialize** $j$ salps ($x_i \forall i = 1,2, \ldots, m$) randomly.
**Calculate** the fitness of each search agent (salp)
**Determine** the best search agent $F$
**while** $t \leq T_{max}$
  **For** each salp ($x_i$)
    **If** $x_i \in S$ then
      **Add** to feasible list ($L_f$)
    **Else**
      **Add** to infeasible list ($L_{in}$)
    **End If**
  **End For**
  **For** feasible list ($L_f$)
    **Update** $k_1$ using Equation (6)
    **Update** the position of salp using Equations (5) and (8)
  **End For**
  **For** infeasible list ($L_{in}$)
    **Update** the position of salp using Equations (22) and (23)
  **End For**
  $\boldsymbol{t = t + 1}$
**end**
**Return** $F$.

**Figure 3.** The pseudo-code of the CSSA.

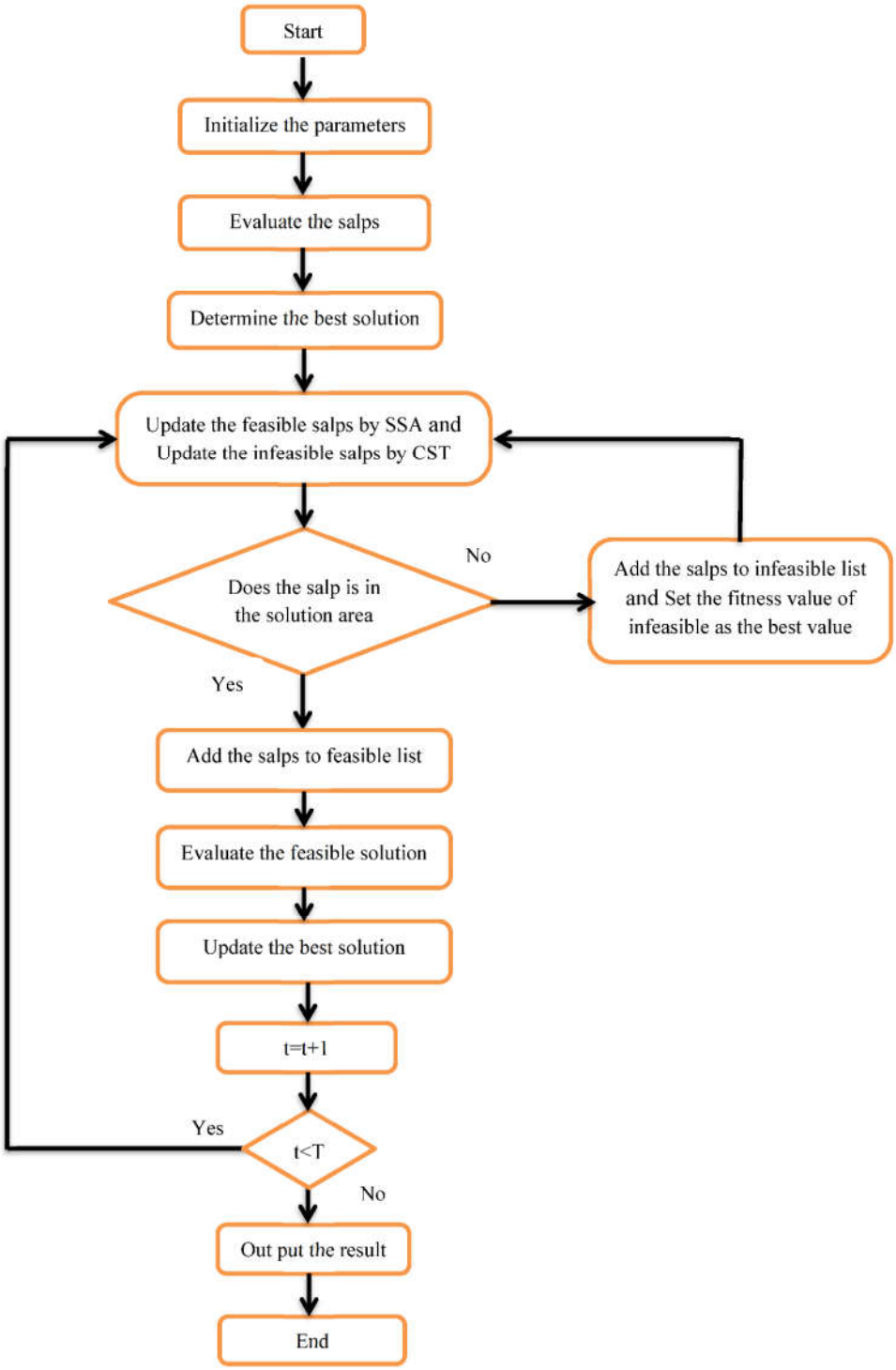


**Figure 4.** Flowchart of the proposed CSSA algorithm.

*4.2. Computational Complexity of the CSSA*

The computational complexity, which is linked to the algorithm's structure and implementation, must be considered when determining the processing time of any metaheuristic algorithms. It is worth mentioning that the recommended CSSA's computing cost is largely determined by two factors: the startup method and the solution update. $O(j \cdot m)$ is the complexity of the initialization process, where $j$ shows the population size

and $m$ is the number of parameters in the problem (dimension). $O(T \cdot L_f \cdot m)$ is the complexity of evaluating solutions by SSA, where $T$ indicates iterations and $L_f = j - L_{in}$ is the number feasible solution. Hence, the computational complexity of the proposed CSSA is $O(j \cdot m) + O(T \cdot L_f \cdot j)$. We can see that the computational complexity of the proposed CSSA, which evaluates only feasible solutions $L_f$, is less than other algorithms that evaluate all populations $k$. In the next section, we used several benchmark tests and real-world optimization problems to assess and certify the performance of the proposed CSSA in handling systems of nonlinear equations.

## 5. Numerical Studies

We checked the effectiveness of the proposed method by several various benchmark problems for a system of nonlinear equations. In each case study, we compared the best results with the obtained solutions of the gravitational search algorithm (GRAV) [59], quantum behaved particle swarm optimization (QPSO) [60,61], the intelligent tuned harmony search method (ITHS) [62], salp swarm algorithm (SSA), the evolutionary algorithm approach (EAA) [43] and with the results of other studies. The comparison established the suggested algorithm's efficiency and robustness. The proposed algorithm is coded in MATLAB R2013a, and the simulations were run on an Intel(R) Core (TM) i5-2430cpu 2.4GHZ, 2.4 GHz processor.

### *5.1. Testing CSSA on Benchmark Problems*

Benchmark 1: It has been solved in [63,64], where the following are its system equations:

$$\begin{aligned} f_1(y) &= 2y_1 + y_2 + y_3 + y_4 + y_5 - 6 = 0, \\ f_2(y) &= y_1 + 2y_2 + y_3 + y_4 + y_5 - 6 = 0, \\ f_3(y) &= y_1 + y_2 + 2y_3 + y_4 + y_5 - 6 = 0, \\ f_4(y) &= y_1 + y_2 + y_3 + 2y_4 + y_5 - 6 = 0, \\ f_5(y) &= y_1 y_2 y_3 y_4 y_5 - 1 = 0; \end{aligned} \tag{24}$$

where $-2 \le y_i \le 2, i = 1, 2, \ldots, 5$. Table 1 compares the results of the proposed CSSA approach to those of existing methods and literature studies. Figure 5 shows the history of convergence for benchmark 1. Table 1 shows that CSSA outperforms the other approaches, except for L-QPSO. Figure 5 indicates, on the other hand, that CSSA is superior to most other algorithms, since it found the optimal solution after 100 iterations, whereas other algorithms did not get close to the optimal solution within 200 iterations.

**Table 1.** Best results for benchmark 1.

| | CSSA | SSA | L-QPSO | QPSO | ITHS | GRAV |
|---|---|---|---|---|---|---|
| $y_1$ | 0.995903811683832 | 0.992951171841475 | 1.418227087330760 | 1.000144199216134 | 0.999999983928838 | 0.916566028481882 |
| $y_2$ | 0.995903811683831 | 0.992980846443041 | 0.916354582533385 | 1.000120111355508 | 0.999999984261559 | 0.916262407670433 |
| $y_3$ | 0.995903811683832 | 0.993004941523888 | 0.916354582533385 | 1.000131686228283 | 0.999999984646038 | 0.916981249858047 |
| $y_4$ | 0.995903811683831 | 0.992853011911152 | 0.916354582533385 | 1.000129832805469 | 0.999999984062309 | 0.916141646585271 |
| $y_5$ | 1.020480941580842 | 1.034169125949290 | 0.916354582533385 | 0.999369060482274 | 1.000000076182086 | 1.417478352612262 |
| $f_1$ | 0.00000000000000 | $-1.08973048967975 \times 10^{-3}$ | 0.00000000000000 | $3.90893038000 \times 10^{-5}$ | $-2.9903315370 \times 10^{-9}$ | $-4.2863102200 \times 10^{-6}$ |
| $f_2$ | 0.00000000000000 | $-1.06005588811442 \times 10^{-3}$ | 0.00000000000000 | $1.50014431730 \times 10^{-5}$ | $-2.6576110240 \times 10^{-9}$ | $-3.0790712167 \times 10^{-4}$ |
| $f_3$ | 0.00000000000000 | $-1.03596080726653 \times 10^{-3}$ | 0.00000000000000 | $2.65763159490 \times 10^{-5}$ | $-2.2731319030 \times 10^{-9}$ | $4.10935065933 \times 10^{-4}$ |
| $f_4$ | 0.00000000000000 | $-1.18789042000245 \times 10^{-3}$ | 0.00000000000000 | $2.47228931350 \times 10^{-5}$ | $-2.8568605230 \times 10^{-9}$ | $-4.2866820683 \times 10^{-4}$ |
| $f_5$ | 0.00000000000000 | $5.30235449108307 \times 10^{-3}$ | 0.00000000000000 | $-1.05338195890 \times 10^{-4}$ | $1.3080826866 \times 10^{-8}$ | $5.32877402310 \times 10^{-5}$ |

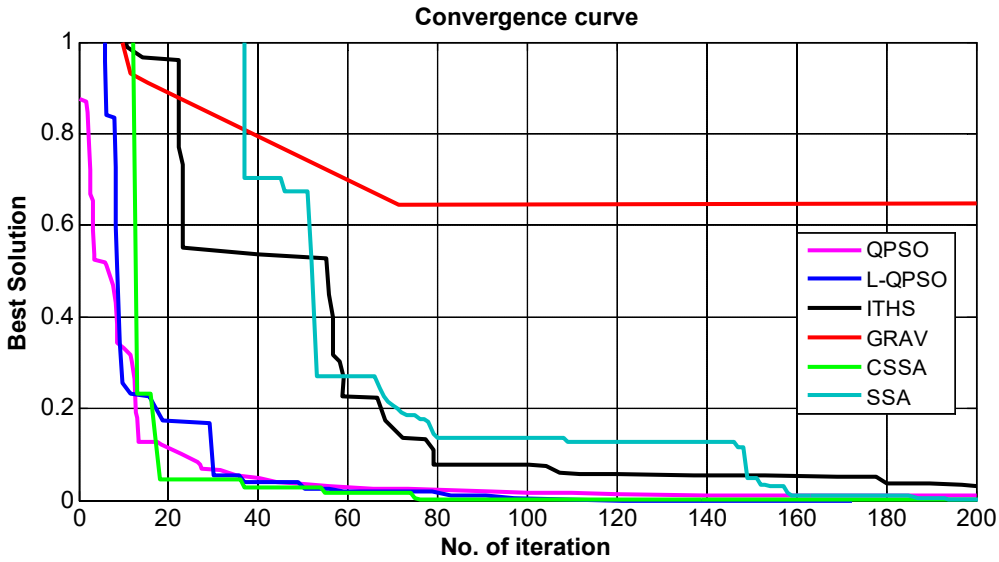


**Figure 5.** Convergence history for benchmark 1.

Benchmark 2: This benchmark has been studied and solved in [63], where its system equations are as follows:

$$\begin{aligned} f_1(y) &= y_1 + \frac{{y_2}^4 y_4 y_6}{4} + 0.75 = 0, \\ f_2(y) &= y_2 + 0.405\, e^{(1+y_1 y_2)} - 1.405 = 0, \\ f_3(y) &= y_3 - \frac{y_4 y_6}{2} + 1.5 = 0, \\ f_4(y) &= y_4 - 0.605\, e^{(1-{y_3}^2)} - 0.395 = 0, \\ f_5(y) &= y_5 - \frac{y_2 y_6}{2} + 1.5 = 0, \\ f_6(y) &= y_6 - y_1 y_5 = 0. \end{aligned} \tag{25}$$

Table 2 displays the best benchmark 2 results obtained by CSSA and other comparative techniques, and Figure 6 displays the objective function's convergence history for all comparison techniques. In solving benchmark 2, CSSA beats the other techniques, as seen in Table 2. According to the objective function's convergence history, CSSA approaches the best solution at 800 iterations, whereas the others, except for SSA, fail to approach the best solution until 1700 iterations, as shown in Figure 6.

**Table 2.** Best results for benchmark 2.

| | CSSA | SSA | LQPSO | QPSO | ITHS | GRAV |
|---|---|---|---|---|---|---|
| $y_1$ | −0.999999999976066 | −0.999999946014511 | −1.0000007185712 | −1.06006024401217 | −1.00469360193614 | −0.94769257629329 |
| $y_2$ | 0.999999999982547 | 0.999999996807600 | 1.000000450917130 | 1.037892210927400 | 1.002699552010810 | 0.959910343740040 |
| $y_3$ | −1.00000000000000 | −1.000000099827310 | −0.9999992090087 | −0.96490791149742 | −0.99763045117792 | −1.03187245704251 |
| $y_4$ | 1.000000000000000 | 0.999999868085587 | 1.000009575627950 | 1.043046172266930 | 1.002704645251820 | 0.961169009727670 |
| $y_5$ | −1.00000000000000 | −1.000000017251620 | −0.99999966616156 | −0.96783925322234 | −0.99763694114471 | −1.03086507223002 |
| $y_6$ | 0.999999999976066 | 0.999999935719242 | 1.000000474541350 | 1.025883305127200 | 1.002627773888710 | 0.976336307899310 |
| $f_1$ | $4.9782400424192 \times 10^{-13}$ | $1.7442991495642 \times 10^{-9}$ | $9.037302239889 \times 10^{-8}$ | $3.6034275521 \times 10^{-4}$ | $-6.3374448056 \times 10^{-4}$ | $1.49483833089 \times 10^{-3}$ |
| $f_2$ | $-6.910028105267 \times 10^{-13}$ | $1.9964645714410 \times 10^{-8}$ | $-2.272550148060 \times 10^{-8}$ | $-7.3227698937 \times 10^{-4}$ | $-2.8872794332 \times 10^{-4}$ | $-1.81606786960 \times 10^{-3}$ |
| $f_3$ | $1.1967316027040 \times 10^{-11}$ | $-1.7297323573473 \times 10^{-9}$ | $7.493891107651 \times 10^{-8}$ | $7.0261199790 \times 10^{-5}$ | $-3.0021434625 \times 10^{-4}$ | $-1.08455815450 \times 10^{-3}$ |
| $f_4$ | 0.000000000000000 | $-1.1123369669797 \times 10^{-8}$ | $4.630491545750 \times 10^{-10}$ | $-1.4208871243 \times 10^{-4}$ | $-1.6590037166 \times 10^{-4}$ | $-8.92448862560 \times 10^{-4}$ |
| $f_5$ | $2.0693669000593 \times 10^{-11}$ | $1.6484962284125 \times 10^{-8}$ | $-1.288909150520 \times 10^{-7}$ | $-2.1739907838 \times 10^{-4}$ | $-3.0415100062 \times 10^{-4}$ | $5.37267309180 \times 10^{-4}$ |
| $f_6$ | 0.0000000000000000 | $-2.7546884551200 \times 10^{-8}$ | $8.980881838205 \times 10^{-8}$ | $-8.4609808280 \times 10^{-5}$ | $3.0832206541 \times 10^{-4}$ | $-6.06868221310 \times 10^{-4}$ |

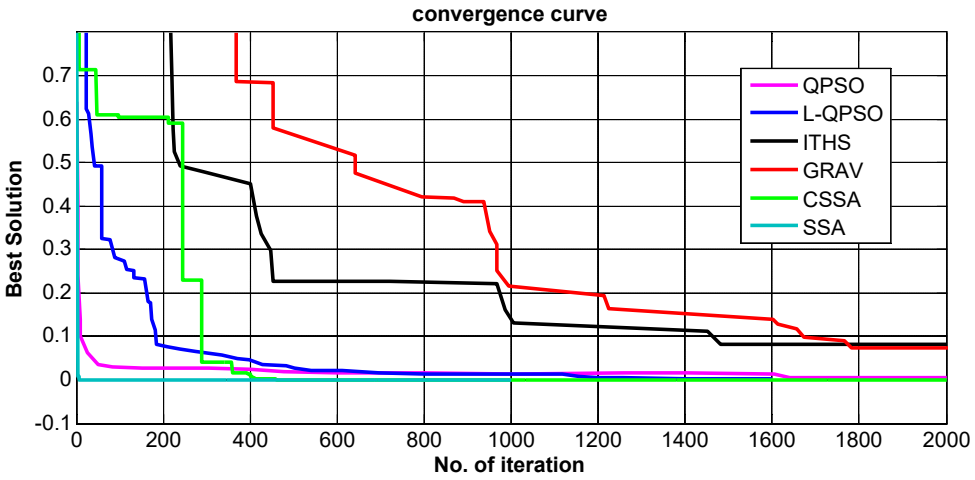


**Figure 6.** Convergence history for benchmark 2.

Benchmark 3: It has been solved in [63], and it is made up of the following system equations:

$$y_i - cos\left(2y_i - \sum_{j=1}^{4} y_j\right) = 0\,, \qquad 1 \le i \le 4. \tag{26}$$

Table 3 displays the simulation results obtained by all algorithms for benchmark 3, and Figure 7 depicts the convergence behaviors. The results show that our algorithm is competitive with other methods.

**Table 3.** Best results for benchmark 3.

| | CSSA | | SSA | LQPSO | QPSO | ITHS | GRAV |
|---|---|---|---|---|---|---|---|
| $y_1$ | 0.514933264661129 | 0.984951602469349 | 0.98495160246935 | 0.514933264661129 | 0.514933264661129 | 0.514933264661129 | 0.514931340034248 |
| $y_2$ | 0.514933264661129 | −0.81124902849969 | 0.98495160246935 | 0.514933264661129 | 0.514933264661129 | 0.514933264661129 | 0.514934600978757 |
| $y_3$ | 0.514933264661129 | 0.984951602469349 | −0.81124902849969 | 0.514933264661129 | 0.514933264661130 | 0.514933264661129 | 0.514923716523520 |
| $y_4$ | 0.514933264661129 | 0.984951602469349 | 0.98495160246935 | 0.514933264661129 | 0.514933264661129 | 0.514933264661129 | 0.514944432722957 |
| $f_1$ | 0.000000000000000 | 0.000000000000000 | 1.11022302462516 × $10^{-16}$ | 0.000000000000000 | 0.000000000000000 | 0.000000000000000 | 2.25940766129 × $10^{-6}$ |
| $f_2$ | 0.000000000000000 | 0.000000000000000 | 0.00000000000000 | 0.000000000000000 | 0.000000000000000 | 0.000000000000000 | −7.0413942500 × $10^{-8}$ |
| $f_3$ | 0.000000000000000 | 0.000000000000000 | −1.11022302462516 × $10^{-16}$ | 0.000000000000000 | 0.000000000000000 | 0.000000000000000 | 7.70620340042 × $10^{-6}$ |
| $f_4$ | 0.000000000000000 | 0.000000000000000 | 0.00000000000000 | 0.000000000000000 | 0.000000000000000 | 0.000000000000000 | −7.0946910128 × $10^{-6}$ |

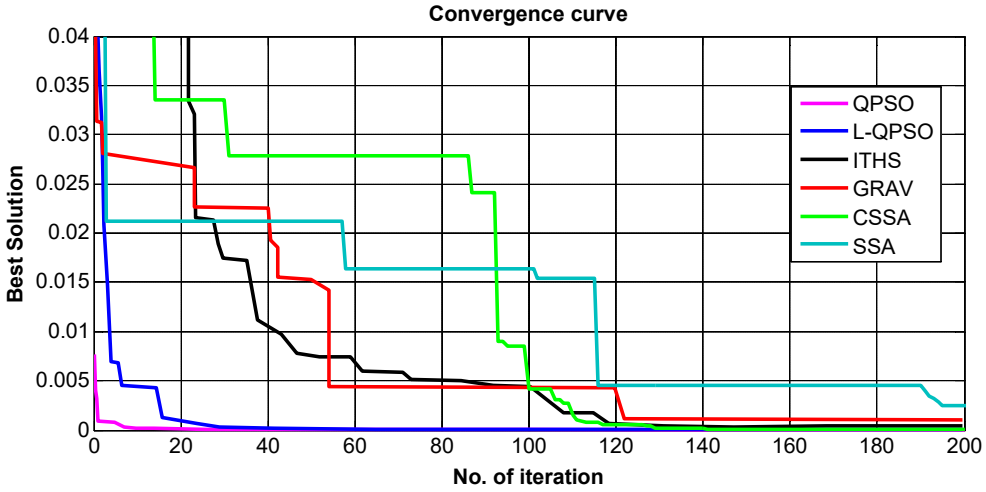


**Figure 7.** Convergence history for benchmark 3.

Benchmark 4 (neurophysiology application): The neurophysiology application [43,63] is used to test the effectiveness and robustness of our method. There are six nonlinear equations in this problem, which are defined as:

$$\begin{aligned} f_1(y) &= {y_1}^2 + {y_3}^2 - 1 = 0, \\ f_2(y) &= {y_2}^2 + {y_4}^2 - 1 = 0, \\ f_3(y) &= y_5{y_3}^3 + y_6{y_4}^3 = c_1, \\ f_4(y) &= y_5{y_1}^3 + y_6{y_2}^3 = c_2, \\ f_5(y) &= y_5y_1{y_3}^2 + y_6{y_4}^2y_2 = c_3, \\ f_6(y) &= y_5{y_1}^2y_3 + y_6{y_2}^2y_4 = c_4; \end{aligned} \tag{27}$$

where $-10 \le y_i \le 10,\ i = 1,2,\ldots,6$. The values of $c_i$ in this equation are sets are chosen at random. For simplicity, we setthese values equal to 0.0 in this study.

A comparison of the best results for benchmark 3 discovered by CSSA and other comparative techniques can be seen in Table 4, and Figure 8 shows its convergence behavior. CSSA beats other algorithms, as seen in Table 4 and Figure 8, where the optimal solution was determined after just 300 iterations.

**Table 4.** Best results for benchmark 4.

| | CSSA | SSA | LQPSO | QPSO | ITHS | GRAV | EAA |
|---|---|---|---|---|---|---|---|
| $y_1$ | 0.670705000548366 | 0.275927502515214 | 0.446209184554328 | −0.796616684320047 | 0.757992217157792 | 0.835326847252122 | 0.045943625 |
| $y_2$ | 0.710697050025244 | −0.275921585990909 | −0.446209184554328 | 0.796616684320047 | 0.757995636725586 | 0.782860693935276 | −0.1626952821 |
| $y_3$ | 0.741724209015329 | −0.961189213584485 | 0.894928691918726 | −0.604484787453692 | 0.652290147139058 | 0.549753670392631 | −0.9215324786 |
| $y_4$ | 0.703498189823838 | 0.961168603463013 | −0.894928691918726 | 0.604484787453692 | 0.652305698905455 | −0.622197020332733 | 0.9841530788 |
| $y_5$ | 0.000000000000000 | −1.398559646254270 | 0.366779058332292 | −0.343529649687506 | 0.026046699540825 | −0.000000012589636 | −0.6789794019 |
| $y_6$ | 0.000000000000000 | −1.398649615180130 | 0.366779058332292 | −0.343529649687506 | −0.026009939089497 | 0.000000014361731 | −0.9070329917 |
| $f_1$ | 0.000000000000000 | 2.0690955444546 × $10^{-5}$ | 0.000000000000000 | 0.000000000000000 | 3.463732647923 × $10^{-5}$ | 3.98503403642 × $10^{-8}$ | 1.489636110 × $10^{-1}$ |
| $f_2$ | 0.000000000000000 | 2.2194101222616 × $10^{-5}$ | 0.000000000000000 | 0.000000000000000 | 6.011011956097 × $10^{-5}$ | −1.78024639200 × $10^{-9}$ | 4.972962500 × $10^{-3}$ |
| $f_3$ | 0.000000000000000 | 0.00000000000000 | 0.000000000000000 | 0.000000000000000 | 9.686086290410 × $10^{-6}$ | −5.55110631700 × $10^{-9}$ | 3.332320690 × $10^{-1}$ |
| $f_4$ | 0.000000000000000 | 6.9388939039072 × $10^{-18}$ | 0.000000000000000 | 0.000000000000000 | 1.585609322857 × $10^{-5}$ | −4.47429924000 × $10^{-10}$ | 3.853671100 × $10^{-3}$ |
| $f_5$ | 0.000000000000000 | 0.00000000000000 | 0.000000000000000 | 0.000000000000000 | 1.141785526275 × $10^{-5}$ | 1.17420316830 × $10^{-9}$ | 1.183698936 × $10^{-1}$ |
| $f_6$ | 0.000000000000000 | 0.00000000000000 | 0.000000000000000 | 0.000000000000000 | 1.345652755927 × $10^{-5}$ | −1.03059189220 × $10^{-8}$ | 2.249327540 × $10^{-2}$ |

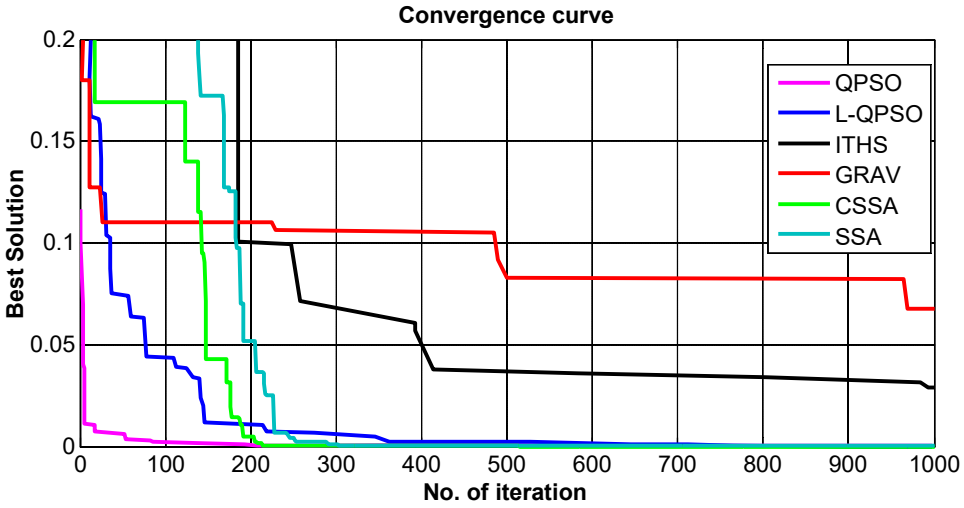


**Figure 8.** Convergence history for benchmark 4.

Benchmark 5: This benchmark shows the interval arithmetic problem that has been presented in [43,63,65]. It was formulated as follows:

$$\begin{aligned}
f_1(y) &= y_1 - 0.25428722 - 0.18324757y_4y_3y_9 = 0,\\
f_2(y) &= y_2 - 0.37842197 - 0.16275449y_1y_{10}y_6 = 0,\\
f_3(y) &= y_3 - 0.27162577 - 0.16955071y_1y_2y_{10} = 0,\\
f_4(y) &= y_4 - 0.19807914 - 0.15585316y_7y_1y_6 = 0,\\
f_5(y) &= y_5 - 0.44166728 - 0.19950920y_7y_6y_3 = 0,\\
f_6(y) &= y_6 - 0.14654113 - 0.18922793y_8y_5y_{10} = 0,\\
f_7(y) &= y_7 - 0.42937161 - 0.21180476y_2y_5y_8 = 0,\\
f_8(y) &= y_8 - 0.07056438 - 0.17081208y_1y_7y_6 = 0,\\
f_9(y) &= y_9 - 0.34504906 - 0.19612740y_{10}y_6y_8 = 0,\\
f_{10}(y) &= y_{10} - 0.42651102 - 0.21466544y_4y_8y_1 = 0.
\end{aligned} \tag{28}$$

Table 5 shows the best results obtained by all algorithms for benchmark 5, and Figure 9 depicts the convergence history. The results show that the proposed algorithm produces solutions very close to the optimal solution and better than the SSA, ITHS, GRAV and EAA methods. Therefore, CSSA is competitive with other methods for benchmark 5.

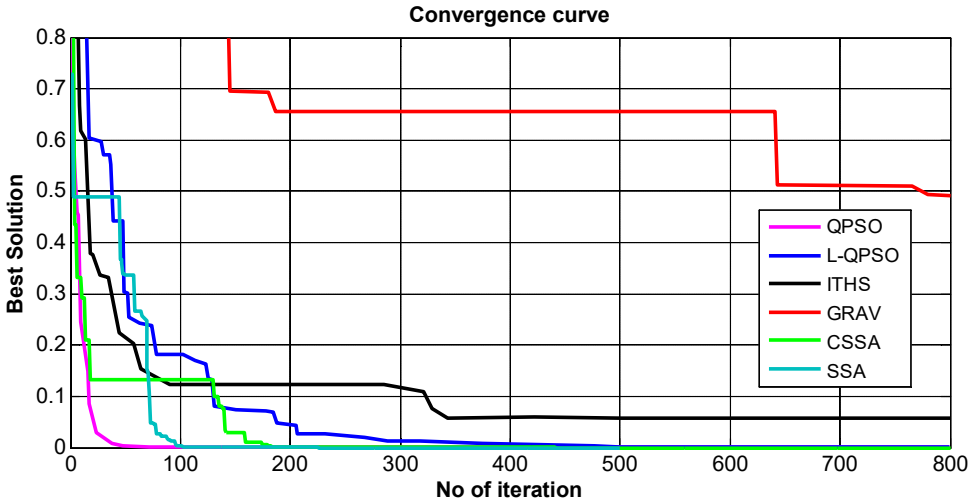


**Figure 9.** Convergence history for benchmark 5.

**Table 5.** Best results for benchmark 5.

| | CSSA | SSA | LQPSO | QPSO | ITHS | GRAV | EAA |
|---|---|---|---|---|---|---|---|
| $y_1$ | 0.2578333937003700 | 0.257833393701697 | 0.257833393700504 | 0.257833393700504 | 0.254686410312621 | 0.257839946926554 | 0.0464905115 |
| $y_2$ | 0.3810971546027980 | 0.381097154598265 | 0.381097154602807 | 0.381097154602807 | 0.378523004753339 | 0.381079261668136 | 0.1013568357 |
| $y_3$ | 0.2787450173464350 | 0.278745017349933 | 0.278745017346440 | 0.278745017346440 | 0.276525468374490 | 0.278737809172705 | 0.0840577820 |
| $y_4$ | 0.2006689642178670 | 0.200668964222069 | 0.200668964225344 | 0.200668964225344 | 0.201804033260634 | 0.200676775829179 | −0.1388460309 |
| $y_5$ | 0.4452514248306970 | 0.445251424835867 | 0.445251424841042 | 0.445251424841042 | 0.443869219215441 | 0.445251560409610 | 0.4943905739 |
| $y_6$ | 0.1491839199689970 | 0.149183919962789 | 0.149183919969355 | 0.149183919969355 | 0.147985685015705 | 0.149185582343140 | −0.0760685163 |
| $y_7$ | 0.4320096977378360 | 0.432009697733842 | 0.432009698983720 | 0.432009698983720 | 0.432376554488803 | 0.432006811493179 | 0.2475819110 |
| $y_8$ | 0.0734027777680547 | 0.073402777761918 | 0.073402777776249 | 0.073402777776249 | 0.069871690818600 | 0.073403712784558 | −0.0170748156 |
| $y_9$ | 0.3459668268754490 | 0.345966826879356 | 0.345966826875554 | 0.345966826875554 | 0.349297348759015 | 0.345965056291278 | 0.0003667535 |
| $y_{10}$ | 0.4273262759931690 | 0.427326275992379 | 0.427326275993291 | 0.427326275993291 | 0.432318039408281 | 0.427333090362260 | 0.1481119311 |
| $f_1$ | $2.25514051876985 \times 10^{-17}$ | $1.16767316302 \times 10^{-12}$ | 0.000000000000000 | 0.000000000000000 | −0.003172703109397 | $6.52503548950 \times 10^{-6}$ | $2.077959240 \times 10^{-1}$ |
| $f_2$ | $-1.51788304147971 \times 10^{-17}$ | $-4.43043605275 \times 10^{-12}$ | 0.000000000000000 | 0.000000000000000 | −0.002550893777805 | $-1.80334006456 \times 10^{-5}$ | $2.769798846 \times 10^{-1}$ |
| $f_3$ | $1.56125112837913 \times 10^{-17}$ | $3.55920674877 \times 10^{-12}$ | 0.000000000000000 | 0.000000000000000 | −0.002166747254159 | $-7.16837992760 \times 10^{-6}$ | $1.876863212 \times 10^{-1}$ |
| $f_4$ | $8.67361737988404 \times 10^{-19}$ | $4.32001049516 \times 10^{-12}$ | 0.000000000000000 | 0.000000000000000 | 0.001185071568625 | $7.73423072990 \times 10^{-6}$ | $3.367887114 \times 10^{-1}$ |
| $f_5$ | $2.73218947466347 \times 10^{-17}$ | $5.30765232112 \times 10^{-12}$ | 0.000000000000000 | 0.000000000000000 | −0.001328103066942 | $2.12270381200 \times 10^{-7}$ | $5.303913210 \times 10^{-2}$ |
| $f_6$ | $-1.25767452008319 \times 10^{-17}$ | $-6.01313096538 \times 10^{-12}$ | 0.000000000000000 | 0.000000000000000 | −0.001092587590357 | $1.58576124970 \times 10^{-6}$ | $2.223730535 \times 10^{-1}$ |
| $f_7$ | $2.25514051876985 \times 10^{-17}$ | $-3.77326642154 \times 10^{-12}$ | 0.000000000000000 | 0.000000000000000 | 0.000518467284375 | $-2.79803507510 \times 10^{-6}$ | $1.816084752 \times 10^{-1}$ |
| $f_8$ | $-1.73472347597681 \times 10^{-18}$ | $-6.00684389382 \times 10^{-12}$ | 0.000000000000000 | 0.000000000000000 | −0.003476285138268 | $8.50208862100 \times 10^{-7}$ | $8.748963860 \times 10^{-2}$ |
| $f_9$ | $6.83047368665868 \times 10^{-18}$ | $4.02272019948 \times 10^{-12}$ | 0.000000000000000 | 0.000000000000000 | 0.003371565377805 | $-1.80713729210 \times 10^{-6}$ | $3.447200366 \times 10^{-1}$ |
| $f_{10}$ | $-1.34441069388203 \times 10^{-17}$ | $-7.42877656092 \times 10^{-13}$ | 0.000000000000000 | 0.000000000000000 | 0.005036117718070 | $6.75152562420 \times 10^{-6}$ | $2.784227489 \times 10^{-1}$ |

Benchmark 6 (combustion application): The combustion problem is taken as a benchmark problem that occurred at a temperature of 3000 °C [43,63,65]. The nonlinear system equations for this problem are described as:

$$\begin{aligned}
&f_1(y) = y_2 + 2y_6 + y_9 + 2y_{10} - 10^{-5} = 0,\\
&f_2(y) = y_3 + y_8 - 3 \times 10^{-5} = 0,\\
&f_3(y) = y_1 + y_3 + 2y_5 + 2y_8 + y_9 + y_{10} - 5 \times 10^{-5} = 0,\\
&f_4(y) = y_4 + 2y_7 - 10^{-5} = 0,\\
&f_5(y) = 0.5140437 \times 10^{-7}\, y_5 - {y_1}^2 = 0,\\
&f_6(y) = 0.1006932 \times 10^{-6}\, y_6 - {y_1}^2 = 0,\\
&f_7(y) = 0.7816278 \times 10^{-15}\, y_7 - {y_4}^2 = 0,\\
&f_8(y) = 0.1496236 \times 10^{-6}\, y_8 - y_1 y_3 = 0,\\
&f_9(y) = 0.6194411 \times 10^{-7}\, y_9 - y_1 y_2 = 0,\\
&f_{10}(y) = 0.2089296 \times 10^{-14}\, y_{10} - y_1 {y_2}^2 = 0.
\end{aligned} \tag{29}$$

Table 6 describes the comparison between CSSA and other comparative techniques according to the best results obtained. Figure 10 depicts the objective function's convergence history for all algorithms. As shown in Table 2, CSSA surpasses the other methods in obtaining the best solution for benchmark 6. CSSA and SSA, on the other hand, have a faster convergence to the optimal solution than other techniques, as seen by the objective function's convergence history in Figure 10.

**Table 6.** Best results for benchmark 6.

| | CSSA | SSA | LQPSO | QPSO | ITHS | GRAV | EAA | |
|---|---|---|---|---|---|---|---|---|
| $y_1$ | 2.905306344721 66 × $10^{-10}$ | 4.96555026855323 × $10^{-6}$ | −5.92864500 × $10^{-8}$ | −4.88278463 × $10^{-7}$ | 0.00724587840930 6 | −0.0012978833822 00 | −0.0552429896 | −0.3383785580 |
| $y_2$ | −2.15930703246316 × $10^{-8}$ | −4.65992582665477 × $10^{-6}$ | −6.94279000 × $10^{-5}$ | 6.47373030 × $10^{-3}$ | 0.01018017693116 3 | 0.00918114480881 1 | −0.0023377533 | 0.0185669333 |
| $y_3$ | 1.31232272837984 × $10^{-8}$ | 1.00000000000000 × $10^{-5}$ | −2.98022727 × $10^{-1}$ | 9.88680886 × $10^{-1}$ | −0.0029051737965 40 | 0.35197654344787 5 | 0.0455880930 | 0.0534924988 |
| $y_4$ | −7.33338519034429 × $10^{-9}$ | 9.85198390404581 × $10^{-6}$ | −8.85260400 × $10^{-5}$ | 6.88493552 × $10^{-3}$ | −0.0023229158180 70 | 0.01344978613674 8 | −0.1287029472 | 0.0392783417 |
| $y_5$ | 1.30613689078792 × $10^{-12}$ | 8.49139051091713 × $10^{-6}$ | −4.12726852 × $10^{-1}$ | 2.49330592 × $10^{-1}$ | −0.0025709928896 10 | 0.38106003737840 5 | 0.0539771728 | 0.0183882247 |
| $y_6$ | −4.81568921923995 × $10^{-12}$ | 6.64457118737294 × $10^{-6}$ | −5.47120683 × $10^{-2}$ | −4.75443378 × $10^{-3}$ | −0.0001285135424 00 | 0.32834051546693 0 | −0.0151036079 | 0.0005246892 |
| $y_7$ | 5.00366669259517 × $10^{-6}$ | 7.40081065763225 × $10^{-8}$ | 4.92534440 × $10^{-5}$ | −3.43749623 × $10^{-3}$ | 0.00107527109720 3 | −0.0067220101518 5 | 0.1063159019 | −0.1024269629 |
| $y_8$ | 2.99868767727162 × $10^{-5}$ | 9.99999958141491 × $10^{-6}$ | 2.98052730 × $10^{-1}$ | −9.88651264 × $10^{-1}$ | 0.00282171705293 6 | −0.3519435919284 2 | 0.0386267592 | 0.0500461848 |
| $y_9$ | −2.99959425329520 × $10^{-5}$ | −5.26744550810345 × $10^{-6}$ | 9.45338532 × $10^{-1}$ | 9.76976826 × $10^{-1}$ | 0.00017060562664 2 | −0.1515959194047 4 | −0.1144905135 | −0.1013361102 |
| $y_{10}$ | 2.00087726173275 × $10^{-5}$ | 3.31911440986020 × $10^{-6}$ | −4.17917503 × $10^{-1}$ | −4.86965844 × $10^{-1}$ | −0.0049371582467 20 | −0.2571440431172 4 | 0.0872294353 | 0.0404252678 |
| $f_1$ | −1.6940658945086 × $10^{-21}$ | −1.402919477773 × $10^{-13}$ | −3.718425434 × $10^{-8}$ | 7.579336100 × $10^{-12}$ | 2.094389795 × $10^{-4}$ | −3.182989656 × $10^{-5}$ | 2.741338780 × $10^{-2}$ | 8.794626000 × $10^{-4}$ |
| $f_2$ | 0.0000000000000000 | −1.000000041859 × $10^{-5}$ | 3.403141215 × $10^{-9}$ | −3.777878940 × $10^{-7}$ | −1.134567436 × $10^{-4}$ | 2.951519453 × $10^{-6}$ | 8.418485220 × $10^{-2}$ | 1.035086837 × $10^{-1}$ |
| $f_3$ | 0.0000000000000000 | −6.450259434649 × $10^{-13}$ | −1.524300123 × $10^{-8}$ | 3.476367522 × $10^{-8}$ | 2.560031932 × $10^{-5}$ | 1.215884437 × $10^{-4}$ | 1.482418893 × $10^{-1}$ | 9.556261970 × $10^{-2}$ |
| $f_4$ | 0.0000000000000000 | 1.171984584440 × $10^{-13}$ | −1.915259325 × $10^{-8}$ | −5.695425910 × $10^{-8}$ | −1.823736237 × $10^{-4}$ | −4.234166957 × $10^{-7}$ | 8.391885670 × $10^{-2}$ | 2.441423777 × $10^{-1}$ |
| $f_5$ | −1.7266905562087 × $10^{-20}$ | −2.422019488989 × $10^{-11}$ | −2.121593269 × $10^{-8}$ | 1.281644356 × $10^{-8}$ | 5.250288608 × $10^{-5}$ | −1.664913123 × $10^{-6}$ | 3.051785100 × $10^{-3}$ | 1.144999500 × $10^{-3}$ |
| $f_6$ | −5.6931520725757 × $10^{-19}$ | −2.398762633404 × $10^{-11}$ | −1.514954299 × $10^{-8}$ | −8.381884680 × $10^{-5}$ | 2.072720176 × $10^{-4}$ | −1.685537783 × $10^{-4}$ | 1.093170000 × $10^{-5}$ | 6.894619000 × $10^{-4}$ |
| $f_7$ | −5.3774627344972 × $10^{-17}$ | −9.706158684552 × $10^{-11}$ | −7.836859219 × $10^{-9}$ | −4.740233710 × $10^{-5}$ | 5.395937898 × $10^{-6}$ | −1.808967471 × $10^{-4}$ | 1.656444860 × $10^{-2}$ | 1.542796700 × $10^{-3}$ |
| $f_8$ | 4.4867406427906 × $10^{-12}$ | −4.815926674816 × $10^{-11}$ | 2.196726011 × $10^{-8}$ | 3.348260222 × $10^{-7}$ | 2.105095828 × $10^{-5}$ | 4.567718476 × $10^{-4}$ | 2.518428300 × $10^{-3}$ | 1.810078900 × $10^{-3}$ |
| $f_9$ | −1.8580656903664 × $10^{-12}$ | 2.281280871601 × $10^{-11}$ | 5.855288223 × $10^{-8}$ | 6.367894304 × $10^{-8}$ | −7.376431366 × $10^{-5}$ | 1.190666480 × $10^{-5}$ | 1.291515000 × $10^{-4}$ | 6.282589000 × $10^{-4}$ |
| $f_{10}$ | 4.1804113131279 × $10^{-20}$ | −1.078195361637 × $10^{-16}$ | −5.071595221 × $10^{-16}$ | 2.046233460 × $10^{-11}$ | 7.509338719 × $10^{-8}$ | 1.094030285 × $10^{-7}$ | 3.019000000 × $10^{-7}$ | 1.166490000 × $10^{-5}$ |

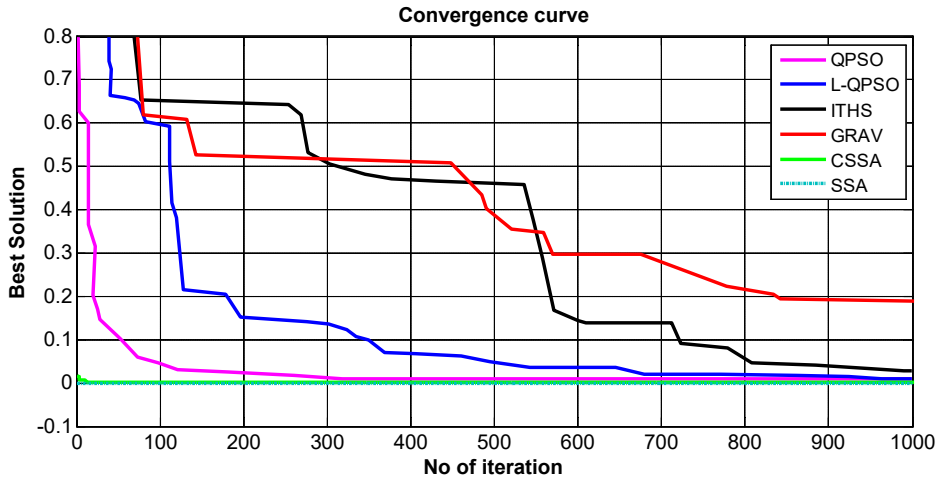


**Figure 10.** Convergence history for benchmark 6.

Benchmark 7: This problem consists of eight equations [63,64,66], demonstrated as follows:

$$\begin{aligned}
f_1(y) &= 4.731 \times 10^{-3}\, y_1 y_3 - 0.3578 y_2 y_3 - 0.1238 y_1 + y_7 - 1.637 \times 10^{-3}\, y_2 - 0.9338 y_4 - 0.3571 = 0\\
f_2(y) &= 0.2238 y_1 y_3 + 0.7623 y_2 y_3 + 0.2638 y_1 - y_7 - 0.007745 y_2 - 0.6734 y_4 - 0.6022 = 0\\
f_3(y) &= y_6 y_8 + 0.3578 y_1 + 4.731 \times 10^{-3} y_2 = 0\\
f_4(y) &= -0.7623\, y_1 + 0.2238 y_2 + 0.3461 = 0\\
f_5(y) &= {y_1}^2 + {y_2}^2 - 1 = 0\\
f_6(y) &= {y_3}^2 + {y_4}^2 - 1 = 0\\
f_7(y) &= {y_5}^2 + {y_6}^2 - 1 = 0\\
f_8(y) &= {y_7}^2 + {y_8}^2 - 1 = 0
\end{aligned} \tag{30}$$

According to [64], this problem has 16 solutions. Table 7 displays a comparison of CSSA and other comparative approaches based on the best findings obtained. Figure 11 shows the convergence history of the objective function for all methods. As demonstrated in Table 2 and Figure 11, CSSA gives good results and outperforms most other algorithms in terms of convergence speed and the best solution.

**Table 7.** Best results for benchmark 7.

| | CSSA | SSA | LQPSO | QPSO | ITHS | GRAV | Abdollahi et al. [66] | Wang et al. [67] |
|---|---|---|---|---|---|---|---|---|
| $y_1$ | 0.164431665854327 | 0.67155424583808 | 0.16443166585432 | 0.1644316658540 | 0.16364139102365 | 0.67170801323877 | 0.16443166585433 | 0.6715446500 |
| $y_2$ | −0.986388476850967 | 0.74095543967366 | −0.98638847685090 | −0.9863884768500 | −0.98627002980711 | 0.74077329293697 | −0.98638847685097 | 0.7409711100 |
| $y_3$ | 0.683029894886492 | −0.67897885223585 | 0.94762379640808 | 0.9476237964080 | 0.95152343463466 | −0.25300250019163 | 0.71845260102760 | 0.9518945900 |
| $y_4$ | −0.730390417989823 | −0.73415797499040 | −0.31938869811110 | −0.3193886981110 | −0.30706727299400 | −0.96749642401711 | −0.69557591970731 | −0.3064372500 |
| $y_5$ | 0.997901815405138 | −0.96316859991350 | −0.99842747393700 | −0.9984274739370 | −0.99686071791680 | 0.95789826890561 | 0.99796438397043 | 0.9638147000 |
| $y_6$ | 0.064745399922544 | −0.26889813599773 | −0.05605871286120 | −0.0560587128610 | 0.03838334619587 | 0.28712600807964 | 0.06377372755700 | −0.2665740500 |
| $y_7$ | −0.547789599039993 | −0.42195486257455 | −0.25758509950460 | −0.2575850995040 | −0.25765415743645 | −0.52841723679413 | −0.52780910528355 | 0.4046369300 |
| $y_8$ | −0.836616133709842 | 0.90661694998613 | 0.96625561654937 | 0.9662556165493 | −0.96464846447580 | −0.84896958414590 | −0.84936302508396 | 0.9144747000 |
| $f_1$ | 0.00000000000000 | $-2.7662843682386 \times 10^{-9}$ | 0.00000000000000 | 0.00000000000000 | $-1.01417138663 \times 10^{-2}$ | $-1.85198337832 \times 10^{-4}$ | $2.77555756156 \times 10^{-16}$ | $-3.7500 \times 10^{-6}$ |
| $f_2$ | 0.00000000000000 | $-3.6325513486091 \times 10^{-8}$ | 0.00000000000000 | 0.00000000000000 | $-1.13945729434 \times 10^{-2}$ | $-9.17918618815 \times 10^{-5}$ | $-1.11022302463 \times 10^{-16}$ | $1.5370 \times 10^{-5}$ |
| $f_3$ | $2.60208521396521 \times 10^{-18}$ | $-3.8569253142835 \times 10^{-8}$ | $2.60208521390 \times 10^{-18}$ | $1.73472347590 \times 10^{-18}$ | $1.68584102280 \times 10^{-2}$ | $8.04779088699 \times 10^{-5}$ | $1.73472347598 \times 10^{-18}$ | $8.9900 \times 10^{-6}$ |
| $f_4$ | 0.00000000000000 | $2.5796597469263 \times 10^{-8}$ | 0.00000000000000 | $1.66533453690 \times 10^{-16}$ | $6.28934951840 \times 10^{-4}$ | $-1.57955532621 \times 10^{-4}$ | $1.66533453694 \times 10^{-16}$ | $1.0840 \times 10^{-5}$ |
| $f_5$ | 0.00000000000000 | $6.8685144682945 \times 10^{-8}$ | 0.00000000000000 | 0.00000000000000 | $-4.92923448120 \times 10^{-4}$ | $-6.32734221494 \times 10^{-5}$ | 0.00000000000000 | $1.0390 \times 10^{-5}$ |
| $f_6$ | 0.00000000000000 | $2.1402552352612 \times 10^{-7}$ | 0.00000000000000 | 0.00000000000000 | $-3.12843197070 \times 10^{-4}$ | $5.95955891025 \times 10^{-5}$ | 0.00000000000000 | $7.0900 \times 10^{-6}$ |
| $f_7$ | 0.00000000000000 | $-4.0597615824645 \times 10^{-8}$ | 0.00000000000000 | 0.00000000000000 | $-4.79542780918 \times 10^{-3}$ | $1.04380881087 \times 10^{-5}$ | 0.00000000000000 | $4.9000 \times 10^{-7}$ |
| $f_8$ | 0.00000000000000 | $2.0005246814669 \times 10^{-7}$ | 0.00000000000000 | 0.00000000000000 | $-3.06767514011 \times 10^{-3}$ | $-2.55869053997 \times 10^{-5}$ | 0.00000000000000 | $-4.9800 \times 10^{-6}$ |

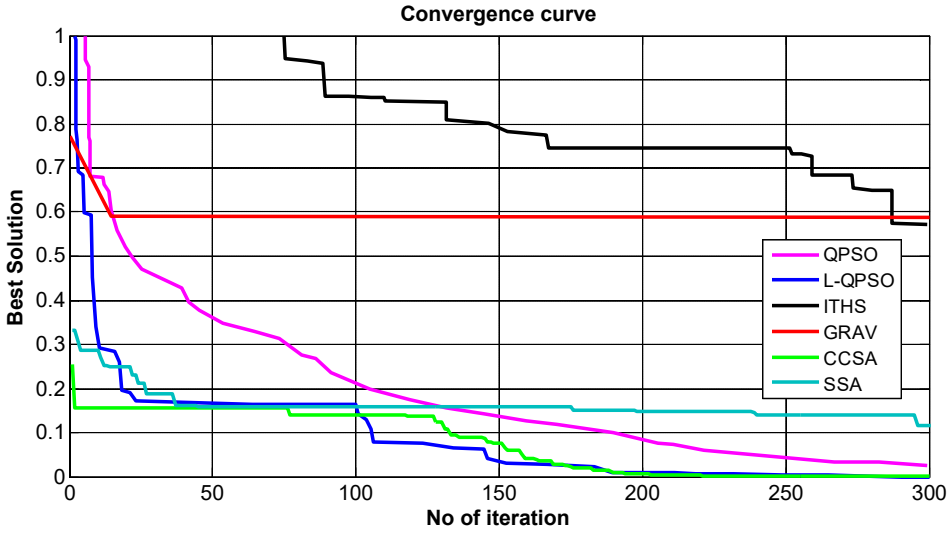


**Figure 11.** Convergence history for benchmark 7.

Benchmark 8: The goal of this benchmark is to find the optimum solution for a thin wall rectangular girder section [29,32,63,66,68]. The following is a description of the non-linear equations system for this problem:

$$\begin{aligned} f_1 &= wL - (w-2k)(L-2k) = 165, \\ f_2 &= \frac{wL^3}{12} - \frac{(w-2k)(L-2k)}{12} = 9369, \\ f_3 &= \frac{2k(L-k)^2(w-k)^2}{L+w-2k} = 6835; \end{aligned} \tag{31}$$

where $w$ is the width of the section, $L$ is the height of the section and $k$ is the thickness of the section. Table 8 demonstrates the best solutions provided by our suggested competitive technique CSSA and other algorithms. The suggested algorithm outperformed some of the other approaches in terms of efficacy and optimum value. Figure 12, on the other hand, presents the history of the convergence behavior of this benchmark problem. As shown in Figure 12, CSSA converged to the optimum result with lower iterations compared to the other algorithms.

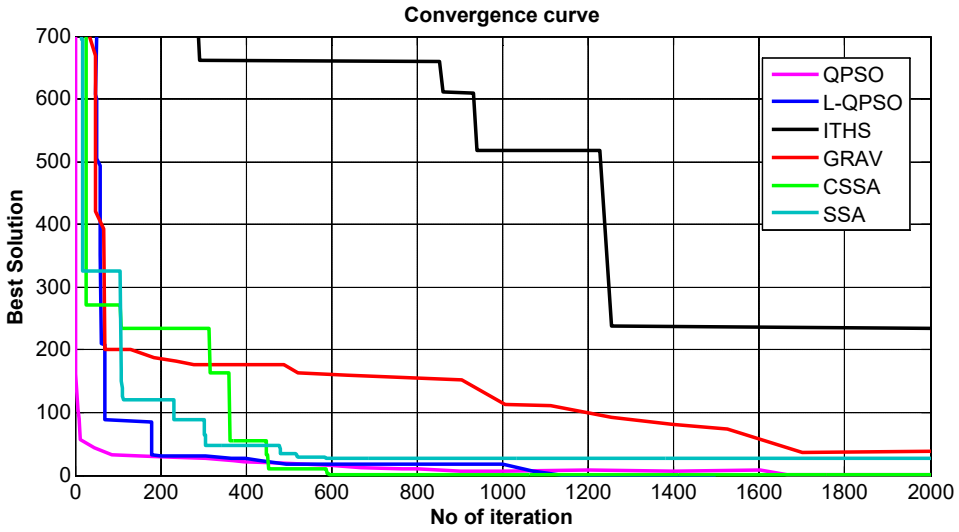


**Figure 12.** Convergence history for benchmark 8.

**Table 8.** Best results for benchmark 8.

| | $w$ | $L$ | $k$ | $f_1$ | $f_2$ | $f_3$ |
|---|---|---|---|---|---|---|
| **CSSA** | 13.17896439 | 20.43927047 | 2.98361764 | 165 | 9369 | 6835 |
| **SSA** | 13.19004771 | 20.43358431 | 2.970077717 | 164.4441539 | 9368.996533 | 6835.012704 |
| **LQPSO** | 12.25651961 | 22.89493892 | 2.789817737 | 165 | 9369 | 6835 |
| **QPSO** | 12.25667461 | 22.90303527 | 2.784985796 | 165.1859 | 9369.0052 | 6834.9887 |
| **ITHS** | 8.915790282 | 23.2914526 | 12.88535322 | 165.8741 | 9366.4924 | 6831.8042 |
| **GRAV** | 12.26024391 | 22.77563468 | 2.857554308 | 167.5713 | 9362.2016 | 6836.7276 |
| **Abdollahi et al. [66]** | 8.943088779 | 23.27148188 | 12.91277429 | 165 | 9369 | 6835 |
| **Jaberipour et al. [29]** | 43.15556605 | 10.1289502 | 12.94404846 | 709.2412 | 9369 | 6835 |
| | −7.602995198 | −24.54198238 | −11.57671567 | 208.1851 | 9369 | 6835 |
| **Mo et al. [32]** | 8.943089 | 23.271482 | 12.912774 | 165 | 9369 | 6835 |
| **Luo et al. [68]** | 12.5655 | 22.8949 | 2.7898 | 166.7229 | 9369 | 6835 |
| | −12.5655 | −22.8949 | -2.7898 | 166.7229 | 9369 | 6835 |
| | 8.943089 | 23.271482 | 12.912774 | 165 | 9369 | 6835 |
| | −8.943089 | −23.271482 | −12.912774 | 165 | 9369 | 6835 |
| | −2.3637 | 35.7564 | 3.0151 | 165 | 9369 | 6835 |
| | 2.3637 | −35.7564 | −3.0151 | 165 | 9369 | 6835 |

Finally, Table 9 compares the statistical data for each benchmark problem, which are best and worst objective values and standard and mean deviations, and compares these data to the other algorithms to evaluate the efficacy and feasibility of our proposed CSSA algorithm. Table 9 shows that CSSA outperforms other algorithms in terms of solution correctness and stability.

**Table 9.** Statistical results for all benchmark problems.

| | **LQPSO** | | | | **QPSO** | | | |
|---|---|---|---|---|---|---|---|---|
| | **Best** | **Sta. Dev.** | **Mean Dev.** | **Worst** | **Best** | **Sta. Dev.** | **Mean Dev.** | **Worst** |
| **Benchmark 1** | 0.000000000000 | 0.000000000000 | 0.000000000000 | 0.000000000000 | $1.190238420 \times 10^{-4}$ | 0.306154753220 | 0.196859612150 | 1.77293447420 |
| **Benchmark 2** | $3.210753720 \times 10^{-8}$ | 0.075730680450 | 0.025461613410 | 0.752622884890 | $8.634949370 \times 10^{-4}$ | 0.225990422210 | 0.228434870310 | 1.31704660560 |
| **Benchmark 3** | 0.000000000000 | $4.24594670 \times 10^{-33}$ | $1.40611580 \times 10^{-33}$ | $2.46519030 \times 10^{-32}$ | 0.0000000000000 | 0.007605882530 | 0.001043577890 | 0.07364986790 |
| **Benchmark 4** | 0.000000000000 | $2.99648960 \times 10^{-73}$ | $2.39145910 \times 10^{-74}$ | $3.77850530 \times 10^{-72}$ | 0.0000000000000 | 0.006962211870 | 0.002762056610 | 0.06168524010 |
| **Benchmark 5** | 0.000000000000 | 0.000000000000 | 0.000000000000 | 0.000000000000 | 0.0000000000000 | 0.141672132510 | 0.102737254690 | 0.55483088300 |
| **Benchmark 6** | $6.20831492 \times 10^{-6}$ | $2.06229320 \times 10^{-4}$ | $6.44202910 \times 10^{-4}$ | $1.13727040 \times 10^{-3}$ | $9.629560810 \times 10^{-5}$ | 0.460256660890 | 0.319865848140 | 2.67500696480 |
| **Benchmark 7** | $2.60208520 \times 10^{-18}$ | $3.18574390 \times 10^{-15}$ | $8.46968200 \times 10^{-16}$ | $1.92128600 \times 10^{-14}$ | $1.665424800 \times 10^{-16}$ | 0.175672734050 | 0.207396539150 | 0.73534258810 |
| **Benchmark 8** | $1.62982150 \times 10^{-10}$ | 102.025240148 | 168.7054837790 | 551.4606323440 | $1.860767480 \times 10^{-1}$ | 176.0526102180 | 228.5159259170 | 767.757960040 |
| | **GRAV** | | | | **ITHS** | | | |
| **Benchmark 1** | $6.710356250 \times 10^{-4}$ | $5.135141020 \times 10^{-3}$ | $6.216924520 \times 10^{-3}$ | $1.72821113 \times 10^{-2}$ | $1.41577466 \times 10^{-8}$ | $1.66313944 \times 10^{-2}$ | $1.23694080 \times 10^{-2}$ | $1.15315200 \times 10^{-1}$ |
| **Benchmark 2** | $2.856977540 \times 10^{-3}$ | $8.672001200 \times 10^{-4}$ | $4.668242410 \times 10^{-3}$ | $6.26278271 \times 10^{-3}$ | $8.88941958 \times 10^{-4}$ | $6.36274540 \times 10^{-2}$ | $8.84356492 \times 10^{-2}$ | $3.01422250 \times 10^{-1}$ |
| **Benchmark 3** | $1.148300900 \times 10^{-10}$ | $7.162948370 \times 10^{-8}$ | $6.735700620 \times 10^{-8}$ | $3.07558863 \times 10^{-7}$ | 0.00000000 | $2.14825657 \times 10^{-3}$ | $4.67800603 \times 10^{-4}$ | $1.82132216 \times 10^{-2}$ |
| **Benchmark 4** | $4.159110120 \times 10^{-8}$ | $2.975234980 \times 10^{-4}$ | $2.550744330 \times 10^{-4}$ | $8.30651132 \times 10^{-4}$ | $7.395709810 \times 10^{-5}$ | $7.20973854 \times 10^{-3}$ | $6.89635827 \times 10^{-3}$ | $5.41546139 \times 10^{-2}$ |
| **Benchmark 5** | $2.321516780 \times 10^{-5}$ | $9.275790350 \times 10^{-5}$ | $1.647237870 \times 10^{-4}$ | $3.81861144 \times 10^{-4}$ | $8.64367972 \times 10^{-3}$ | $1.29614798 \times 10^{-2}$ | $2.84851053 \times 10^{-2}$ | $8.82327076 \times 10^{-2}$ |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Benchmark 6** | 5.345489820 × $10^{-4}$ | 1.599892620 × $10^{-3}$ | 2.649030450 × $10^{-3}$ | 5.60918745 × $10^{-3}$ | 3.77206677 × $10^{-4}$ | 2.32644338 × $10^{-2}$ | 2.01043301 × $10^{-2}$ | 1.62034142 × $10^{-1}$ |
| **Benchmark 7** | 2.872005150 × $10^{-4}$ | 8.897540130 × $10^{-2}$ | 1.383110850 × $10^{-1}$ | 2.91779881 × $10^{-1}$ | 2.34529275 × $10^{-2}$ | 7.19144829 × $10^{-2}$ | 1.56549347 × $10^{-1}$ | 4.15978152 × $10^{-1}$ |
| **Benchmark 8** | 7.4708993593300 | 17.805724748100 | 37.155629330300 | 130.470752028 | 4.155044146940 | 338.2946656533 | 312.651184712 | 5555.0798892 |
| | **CSSA** | | | | | **SSA** | | |
| **Benchmark 1** | 0.0000000000000 00000 | 0.0000000000000 00000 | 0.0000000000000 00000 | 0.000000000000 000000 | 3.291049261929 84 × $10^{-5}$ | 3.9510813495940 9 × $10^{-5}$ | 3.3358717040868 0 × $10^{-5}$ | 1.29503618132905 × $10^{-4}$ |
| **Benchmark 2** | 5.7216990322050 4 × $10^{-22}$ | 1.2745384652539 3 × $10^{-9}$ | 9.5590243405657 0 × $10^{-10}$ | 2.550624112581 50 × $10^{-9}$ | 1.558935814850 26 × $10^{-15}$ | 2.9414638598831 2 × $10^{-5}$ | 2.7526616703633 6 × $10^{-5}$ | 6.63152999553047 × $10^{-5}$ |
| **Benchmark 3** | 0.0000000000000 00000 | 0.0000000000000 00000 | 0.0000000000000 00000 | 0.000000000000 000000 | 2.465190328815 68 × $10^{-32}$ | 7.4829241186504 4 × $10^{-7}$ | 7.0538517301082 9 × $10^{-7}$ | 1.60342282199816 × $10^{-6}$ |
| **Benchmark 4** | 0.0000000000000 00000 | 0.0000000000000 00000 | 0.0000000000000 00000 | 0.000000000000 000000 | 9.206937662879 13 × $10^{-10}$ | 2.4348909061753 1 × $10^{-2}$ | 2.2915721987255 6 × $10^{-2}$ | 5.33342563177076 × $10^{-2}$ |
| **Benchmark 5** | 2.6271005697092 3 × $10^{-33}$ | 2.4328966024523 5 × $10^{-24}$ | 1.8246724518392 5 × $10^{-24}$ | 4.865793369888 02 × $10^{-24}$ | 1.837054428284 98 × $10^{-22}$ | 4.5700831387281 3 × $10^{-15}$ | 4.3083067560107 8 × $10^{-15}$ | 9.77080461506118 × $10^{-15}$ |
| **Benchmark 6** | 3.5998906894356 6 × $10^{-9}$ | 1.9999997399205 4 × $10^{-8}$ | 1.4999998049403 3 × $10^{-8}$ | 4.359989429866 62 × $10^{-8}$ | 1.000000083852 23 × $10^{-10}$ | 8.3115375752624 2 × $10^{-6}$ | 7.8358395109649 4 × $10^{-6}$ | 1.77271255508841 × $10^{-5}$ |
| **Benchmark 7** | 6.7708474607363 7 × $10^{-36}$ | 8.5592086563015 3 × $10^{-15}$ | 6.4194064922261 5 × $10^{-15}$ | 1.711841731260 31 × $10^{-14}$ | 9.567397723119 61 × $10^{-14}$ | 4.7340175108614 0 × $10^{-13}$ | 4.4628571947463 5 × $10^{-13}$ | 1.10774526946331 × $10^{-12}$ |
| **Benchmark 8** | 0.0000000000000 00000 | 0.0000000000000 00000 | 0.0000000000000 00000 | 0.000000000000 000000 | 3.091383025728 01 × $10^{-1}$ | 6.5137396704817 6 × $10^{1}$ | 6.1406314257555 6 × $10^{1}$ | 1.39570903957062 × $10^{2}$ |

### *5.2. Wilcoxon Signed Ranks Test (WSRT)*

The Wilcoxon signed ranks test compares two related groups and is a nonparametric statistical test. The tests work by calculating the difference between sets of pairings and analyzing those differences to see if they are statistically significant. It is linked to the $p$-value, where $p$ signifies the chance that the null hypothesis is true. The result of the test is returned as $p < 0.05$, which shows that the null hypothesis was refused, and $p > 0.05$, which means the null hypothesis was not refused. The positive ranks' sum is $R+$, and the negative ranks' sum is $R-$. Table 10 shows the results of the Wilcoxon signed-rank test for the various methods, where our proposed CSSA is compared with SSA, QPSO, LQPSO, ITHS and GRAV. Table 10 demonstrates that all $p$-values $< 0.05$ and CSSA achieve better $R-$ values than $R+$ values in all benchmark problems, indicating that CSSA outperforms the other algorithms.

**Table 10.** WSRT for results in Table 9.

| **Compared Algorithms** | | **Solution Evaluations** | | | |
|---|---|---|---|---|---|
| | | $R-$ | $R+$ | $p$-**Value** | **Best Algorithm** |
| **CSSA** | **SSA** | 36 | 0 | 0.012 | CSSA |
| **CSSA** | **LQPSO** | 14 | 1 | 0.08 | CSSA |
| **CSSA** | **QPSO** | 20 | 1 | 0.046 | CSSA |
| **CSSA** | **GRAV** | 36 | 0 | 0.012 | CSSA |
| **CSSA** | **ITHS** | 28 | 0 | 0.018 | CSSA |

### *5.3. Convergence Analysis*

For studying the convergence analysis of our proposed algorithm, the statistical findings for each approach, in terms of standard deviation, the best value, the worst value and the mean deviation value, were recorded in Table 9, and the Wilcoxon signed ranks (WSRs) test was employed as a non-parametric test. We can see from the simulation results in Table 9 that our proposed CSSA has better searching quality and outperforms other comparable algorithms. In addition to using non-parametric WSRs to determine the winning algorithm, Table 10 demonstrates that our proposed approach beats other methods in terms of the calculated $p$-value. Concerning the offered analyses, we may conclude that the ingrained property of this enhancement exists in the chaotic search technique,

with attention to the infeasible solution, which speeds the convergence behavior and avoids the algorithm's regular running without any changes in the results. Our proposed algorithm CSSA is extremely competitive compared with other algorithms in terms of computing the statistical measurements, and it has a great possibility to solve nonlinear systems and their applications.

## 6. Power system Applications

Power system analysis is a branch of electrical engineering for designing whole power systems that consist of generators, transformers, shunt reactance, capacitor banks, transmission lines and so on. The purpose of power system analysis is to make certain the equipment works together so that the required power is sent to the load centers at the specified voltage and frequency, and an overloaded component in the network or any fault condition will not endanger the system. The load flow analysis is an essential instrument in the power system; it is used for planning and determining the optimal power system operation and electricity reciprocation between utility firms. Under steady-state conditions, studies of power flow use disciplinarian mathematical methods for computing the phase angles, different bus voltages, active power and reactive power flows through various sections, transformers, generators settings and load [69]. Load flow analysis can give a balanced steady-state operation of the power system without considering system transient processes. It minimizes losses to the system and transformer tap settings for economic operation. Hence, the mathematical model of the load flow problem is a nonlinear equations system. The power system increases because of developments in the industry in society, so the load flow equation dimension also increases. There is not a numerical method that can lead to the exact solution. Therefore, engineers exert more effort to find more reliable methods. The power industry faces the problem of how to find the best method for power system analysis.

### *6.1. Formulation of Load Flow Problem*

#### 6.1.1. Classification of Buses

A bus is a node or point that connects one or more transmission lines, generators and loads. Each bus has four parameters: reactive power $(Q)$, active power $(P)$, voltage phase angle $(\delta)$ and voltage magnitude $(|V|)$ [70]. Two parameters of the bus are known, but the other two parameters must be calculated by solving the node active and reactive power equations. The buses are categorized into three types based on the two known parameters, as shown in Table 11.

**Table 11.** Bus classification.

| No. | Type of Bus | Variables | | | |
|---|---|---|---|---|---|
| | | $\boldsymbol{P}$ | $\boldsymbol{Q}$ | $\boldsymbol{\lvert V\rvert}$ | $\boldsymbol{\delta}$ |
| **1** | **Slack bus** | required | required | given | given |
| **2** | **Generator bus** $(\boldsymbol{PV})$ | given | required | given | required |
| **3** | **Load bus** $(\boldsymbol{PQ})$ | given | given | required | required |

- Slack bus

In the power system, there should be one and only one slack node, and it is commonly set as bus 1. To achieve the power balancing criteria, this bus is considered a reference bus. The slack bus is used as a unit for generating, which is ably modified to meet all needs to ensure a balance of power [70]. At this bus, the effective generator provides losses to the utility network, which is necessary because the value of the losses will be unknown until the current is calculated completely. In this bus, $|V|$ and $\delta$ are known variables, and $P$ and $Q$ are unknown variables.

- Generator $(\boldsymbol{PV})$ bus

This is a voltage control bus, and it is linked to a generator unit. The prime mover may be adjusted to control the active power generated by this bus, and the generator's excitation can be adjusted to control the voltage. Frequently, limits for reactive power values are defined based on the characteristics of each generator. $P$ and $|V|$ are known variables in this bus, and $Q$ and $\delta$ are unknown variables [70,71].

- Load ($\boldsymbol{PQ}$) bus

This is a bus that does not have a generator and may be computed using historical information, measurements or forecasts. Positive power is the energy that is supplied to a power system, whereas negative power is the energy that is consumed. In the power system, most nodes are from the $PQ$ type. $P$ and $Q$ are known variables in this bus, and |V| and δ are unknown variables [70,71].

6.1.2. Node Power Equations

At the $i^{\text{th}}$ bus, the node active and reactive power equations are as follows:

$$P_G^i - P_L^i = |V_i| \sum_{k=1}^{n} |V_k| [G_{ik}\, cos(\delta_i - \delta_k) + B_{ik}\, sin(\delta_i - \delta_k)], \quad (32)$$

$$Q_G^i - Q_L^i = |V_i| \sum_{k=1}^{n} |V_k| [G_{ik}\, sin(\delta_i - \delta_k) - B_{ik}\, cos(\delta_i - \delta_k)]; \quad (33)$$

where $\left(P_G^i + jQ_G^i\right)$ is the complex generation, $\left(P_L^i + jQ_L^i\right)$ is the complex load at the node $i$, $(G_{ik} + jB_{ik})$ is the complex admittance between the $i$ and $k$ nodes, $V_i$ is the voltage magnitude at the node $i$, $\delta_i$ is the phase angle of voltage at the node $i$ and $n$ is the number of nodes.

Power system application 1: Three-bus system

This electrical application is taken as a benchmark problem in [70,72]. Figure 13 shows the single line diagram of the three bus systems. Bus 1 is the slack bus. Its magnitude of voltage is 1.05 per unit, and its phase angle is zero. The loads on buses 2 and 3 as clear on the diagram. The impedance of the lines is shown per unit on a 100-MVA base, with the line charging capacitances ignored.

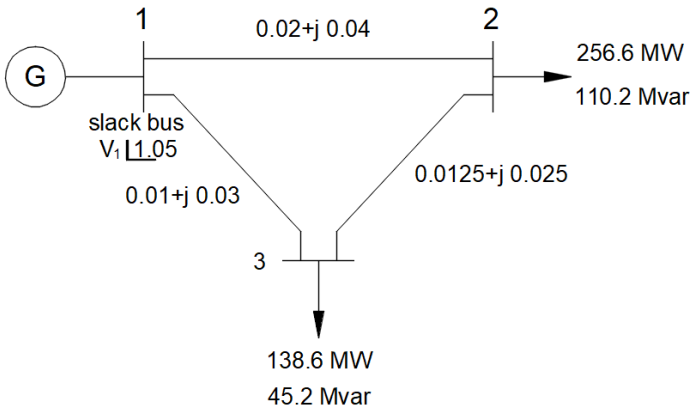


**Figure 13.** Single line diagram for the three-bus system.

Table 12 shows a comparison between our proposed algorithm CSSA with other algorithms. Further Figure 14 shows the history of the convergence behavior of this electrical application. According to the presented results, the suggested algorithm surpasses the other approaches in terms of optimal value. In addition, it has faster convergence than the standard SSA.

**Table 12.** Results for the three-bus system.

| | CSSA | SSA | SCA [72] | Q-SCA [72] | Saadat [70] |
|---|---|---|---|---|---|
| $\|V_1\|$ | 1.05 | 1.05 | 1.05 | 1.05 | 1.05 |
| $\|V_2\|$ | 0.981835016690686 | 0.982496442317607 | 0.981838079377364 | 0.981835016690686 | 0.98183 |
| $\|V_3\|$ | 1.001249219725040 | 1.001753708151100 | 1.001251283783532 | 1.001249219725039 | 1.00125 |
| $\delta_1$ | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| $\delta_2$ | −3.50353164478445 | −3.4603930381497400 | −3.502382236093232 | −3.503531644784462 | −3.5035 |
| $\delta_3$ | −2.86240522611174 | −2.8425024238824500 | −2.861380189597466 | −2.862405226111747 | −2.8624 |
| $Pg_1$ | 409.50 | 405.73129813204 | N.C. | N.C. | 409.50 |
| $Qg_1$ | 189.00 | 183.25286594842 | N.C. | N.C. | 189.00 |
| $f_1$ | 0.000000000000000 | $-3.4795486078437 \times 10^{-2}$ | $-5.05150474218397 \times 10^{-4}$ | $8.8817841970013 \times 10^{-16}$ | N.C. |
| $f_2$ | 0.000000000000000 | $-2.0558824779045 \times 10^{-3}$ | $-4.88402153445477 \times 10^{-4}$ | $4.4408920985006 \times 10^{-16}$ | N.C. |
| $f_3$ | 0.000000000000000 | $-1.8891887805239 \times 10^{-6}$ | $1.78159569490965 \times 10^{-4}$ | $-2.44249065417534 \times 10^{-15}$ | N.C. |
| $f_4$ | 0.000000000000000 | $-1.1836177929517 \times 10^{-2}$ | $1.54114917028381 \times 10^{-4}$ | $2.66453525910038 \times 10^{-15}$ | N.C. |
| $f_5$ | 0.000000000000000 | $1.0080130330437 \times 10^{-7}$ | N.C. | N.C. | N.C. |
| $f_6$ | 0.000000000000000 | $-3.6960016677433 \times 10^{-2}$ | N.C. | N.C. | N.C. |

N.C. means that the result is not calculated.

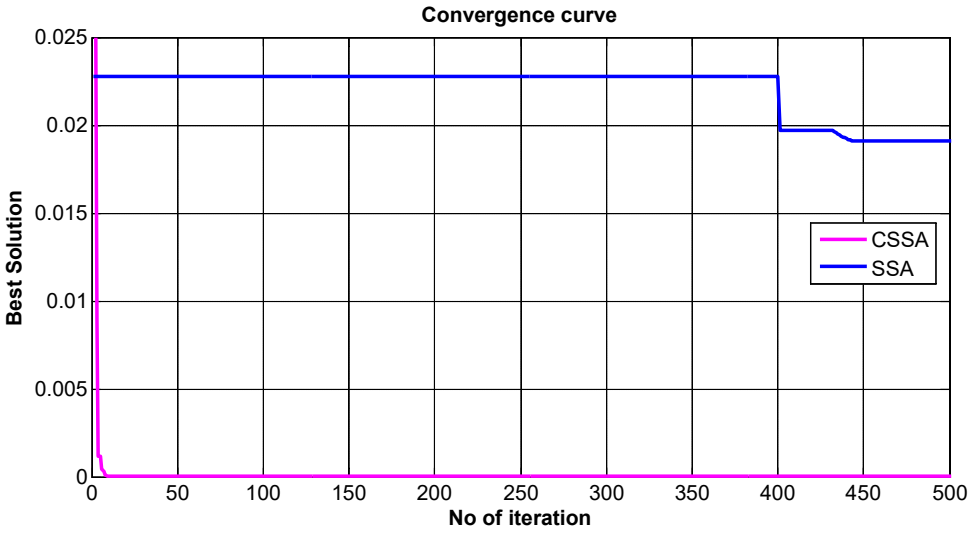


**Figure 14.** Convergence history for the three-bus system.

Power system application 2: Five-bus system

A five-bus system is an electrical network that has been selected as a benchmark problem [72]. Figure 15 shows a single-line diagram for the five-bus system, which contains a slack bus (bus 1), two generators buses and two load buses. The voltage magnitude of the slack bus is 1.06 per unit, and its phase angle is zero, Table 13 shows the system's whole data per unit on a 100-MVA base.

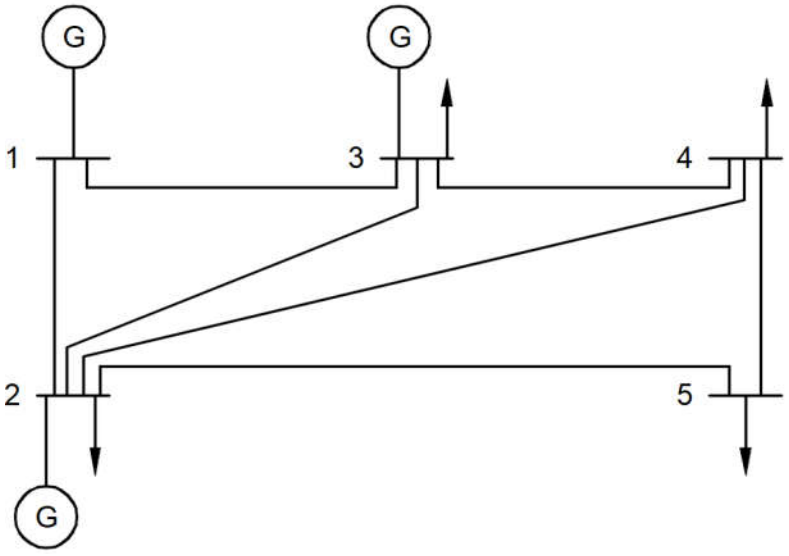


**Figure 15.** Single-line diagram for the five-bus system.

**Table 13.** The complete data for the five-bus system.

| Line $(L_{ij})$ | Impedance | | Admittance | |
|---|---|---|---|---|
| $L_{12}$ | 0.020 + j0.060 | | j0.03 | |
| $L_{13}$ | 0.080 + j0.240 | | j0.025 | |
| $L_{23}$ | 0.060 + j0.180 | | j0.02 | |
| $L_{24}$ | 0.060 + j0.180 | | j0.02 | |
| $L_{25}$ | 0.040 + j0.120 | | j0.015 | |
| $L_{34}$ | 0.010 + j0.030 | | j0.01 | |
| $L_{45}$ | 0.080 + j0.240 | | j0.025 | |
| **Node** | **Data of Generator** | **Data of Load** | **Voltage of Node** | **Type of Node** |
| **1** | $Pg_1 + jQg_1$ | - | $1.06 \angle 0$ | Slake |
| **2** | $0.4 + jQg_2$ | 0.2 + j0.1 | $1.045 \angle \delta_2$ | V. controlled |
| **3** | $0.3 + jQg_3$ | 0.2 + j0.15 | $1.03 \angle \delta_3$ | V. controlled |
| **4** | - | 0.5 + j0.3 | $V_4 \angle \delta_4$ | Load |
| **5** | - | 0.6 + j0.4 | $V_5 \angle \delta_5$ | Load |

In Table 14, the results of this power system application are shown, and the optimum solution for active and reactive power, voltage and phase angle at each node are mentioned. Further, Figure 16 displays the convergence behavior. The obtained results demonstrated robustness and efficiency for our proposed technique in addressing this power system application, and it beats the other algorithms. Furthermore, CSSA has a faster convergence than the usual SSA, as shown in Figure 16.

**Table 14.** Results for the five-bus system.

| | CSSA | SSA | SCA [72] | Q-SCA [72] |
|---|---|---|---|---|
| $\|V_1\|$ | 1.06 | 1.06 | 1.06 | 1.06 |
| $\|V_2\|$ | 1.056 | 1.056 | 1.056 | 1.056 |
| $\|V_3\|$ | 1.03 | 1.03 | 1.03 | 1.03 |
| $\|V_4\|$ | 1.01863059295273 | 1.01952646373577 | 1.018435127204843 | 1.018630577870106 |
| $\|V_5\|$ | 0.990099003985225 | 0.99750656900719 | 0.989538210278804 | 0.990098992150422 |
| $\delta_1$ | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| $\delta_2$ | −1.78246929197204 | −1.59427491015746 | −1.944789603009603 | −1.782148822376761 |
| $\delta_3$ | −2.6640102508242 | −2.42268934031169 | −2.919882018480426 | −2.66376758698785 |
| $\delta_4$ | −3.24314133420668 | −2.97827944424071 | −3.511490409163109 | −3.242881574384807 |
| $\delta_5$ | −4.40507424178388 | −4.11900384738152 | −4.675411331717214 | −4.404773706610992 |
| $\mathbf{Pg}_1$ | 83.052564864276 | 75.795550223517 | N.C | N.C |
| $\mathbf{Qg}_1$ | 7.27097023180857 | 2.36345263841315 | N.C | N.C |
| $Pg_2$ | 40 | 40 | 40 | 40 |
| $Qg_2$ | 41.8123141017395 | 34.2364944774385 | N.C | N.C |
| $Pg_3$ | 30 | 30 | 30 | 30 |
| $\mathbf{Qg}_3$ | 24.1494180415641 | 19.9147887667246 | N.C | N.C |
| $f_1$ | 0.0000000000000000 | $1.05025892427335 \times 10^{-8}$ | 0.011471071353534 | −0.279221090693227 |
| $f_2$ | $-1.77635683940025 \times 10^{-15}$ | $-7.06283406052970 \times 10^{-2}$ | 0.017876378728437 | −0.374561492932912 |
| $f_3$ | 0.0000000000000000 | $-1.01589935000534 \times 10^{-2}$ | 0.018260614359347 | −0.400235400377369 |
| $f_4$ | $8.88178419700125 \times 10^{-16}$ | $-2.51482193291963 \times 10^{-7}$ | 0.015992420160200 | −0.316413562018170 |
| $f_5$ | $1.77635683940025 \times 10^{-15}$ | $-4.03229812917516 \times 10^{-6}$ | −0.000366016023354 | −0.026645352591004 |
| $f_6$ | 0.0000000000000000 | $-1.05128974045066 \times 10^{-2}$ | −0.000323620656330 | 0.034972025275692 |
| $f_7$ | $-2.22044604925031 \times 10^{-16}$ | $-2.10402859592851 \times 10^{-2}$ | N.C | N.C |
| $f_8$ | $-8.88178419700125 \times 10^{-16}$ | $-1.97171408924390 \times 10^{-5}$ | N.C | N.C |
| $f_9$ | $-2.22044604925031 \times 10^{-16}$ | $-3.65790017599030 \times 10^{-2}$ | N.C | N.C |
| $f_{10}$ | 0.0000000000000000 | $-7.09655695185432 \times 10^{-2}$ | N.C | N.C |

N.C. means that the result is not calculated.

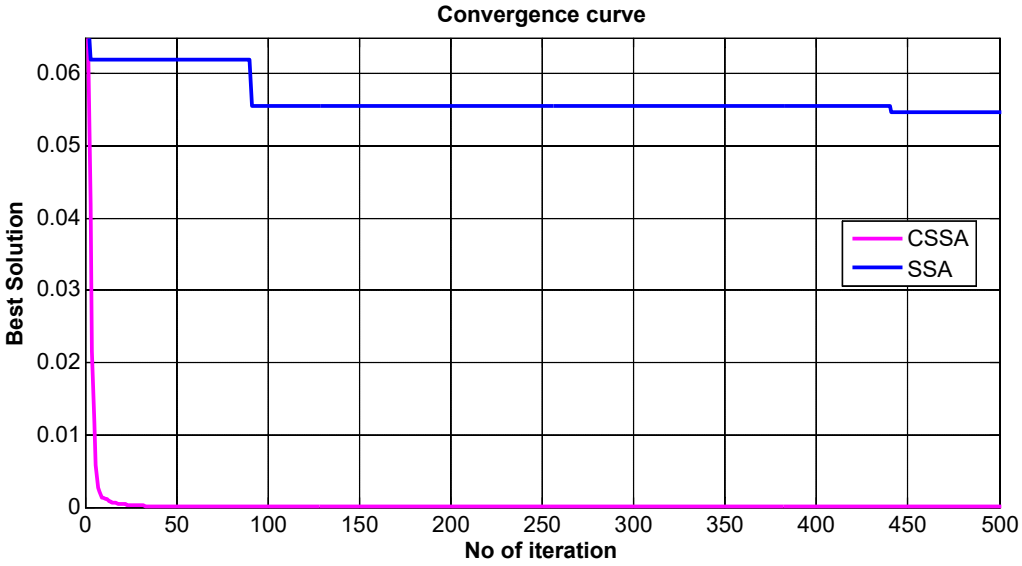


**Figure 16.** Convergence history for benchmark 10.

Finally, the obtained results demonstrate the effectiveness and robustness of CSSA in solving power system applications. Thus, we can deduce that it is suitable to solve real applications related to solving the nonlinear system equations.

## 7. Conclusions

This study presented the chaotic salp swarm algorithm (CSSA), an intelligent hybrid optimization technique for solving systems of nonlinear equations (SNLEs) that is a combination of the salp swarm algorithm (SSA) and the chaotic search strategy (CST). Firstly, SNLEs were transformed into an optimization problem. Then, CSSA was used to solve this optimization problem; SSA was used to update the feasible solutions, and the infeasible solutions were updated by CST. CSSA was tested using several benchmark SNLEs problems and two electrical applications. The suggested method demonstrated various advantages, which we mention below:

1. CSSA is simple in application, and it can solve many optimization problems.
2. CSSA combines the SSA's robust global searching capacity with the CST's substantial chaotic searching ability.
3. Only objective function information is used in CSSA; no derivatives or other auxiliary data is used.
4. CSSA can transact with the non-continuous, non-differentiable and non-smooth functions that are common in problems of optimization.
5. CSSA can give a globally optimal solution because it searches at a set of points, not a single point, unlike traditional techniques.
6. The combination between SSA and CST and not ignoring infeasible solutions led to enhancing the efficacy of the search, increasing solution versatility, avoiding the local optima trap, speeding up convergence and optimizing the search process.
7. Results have proven the superiority of CSSA over those reported in the literature, as it is significantly better than other comparison methods.
8. Statistical results showed that CSSA solutions are more accurate and stable than most other algorithms' solutions.
9. CSSA converges more quickly to the optimal solution in the early iteration.
10. A Wilcoxon signed ranks test showed the significance of the CSSA findings.
11. By addressing power system applications with CSSA, we can conclude that it is suited for tackling real-world applications that are related to nonlinear system equations.

Without any prejudice, our proposed approach, like other meta-heuristics approaches, has the probable drawback of not warranting a rise in computing speed or accuracy when we tackle any optimization issue. This is because meta-heuristics approaches are random techniques, and the computational effectiveness as well as the CSSA's solution quality are determined by the problem's nature and complexity. In future works, we plan to solve more applications to prove the effectiveness and efficiency of our algorithm, and we will concentrate on advancing new algorithms for solving optimization problems.

**Author Contributions:** Conceptualization, M.A.E.-S., I.N. and I.M.E.; Methodology, M.A.E.-S. and I.N.; Writing and original draft preparation, I.N.; Co-review and validation, M.A.E.-S., I.M.E., and A.M.E.-R.; Writing—editing; M.A.E.-S., I.N., M.M.B. and A.M.E.-R. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** All data used to support the findings of this study are included within the article.

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

## References


1. Allman, E.S.; Allman, E.S.; Rhodes, J.A. *Mathematical Models in Biology: An Introduction*; Cambridge University Press: Cambridge, UK, 2004.
2. Wang, R.; Zhang, Z.; Duan, Y.B. Nonlinear stochastic models of neurons activities. *Neurocomputing* **2003**, *51*, 401–411.
3. Kerkhoven, T. A Proof of Convergence of Gummel's Algorithm for Realistic Device Geometries. *SIAM J. Numer. Anal.* **1986**, *23*, 1121–1137.
4. Bartholomew-Biggs, M. *Nonlinear Optimization with Engineering Applications*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2008; Volume 19.
5. Cuyt, A.; Van der Cruyssen, P. Abstract Padé-approximants for the solution of a sytem of nonlinear equations. *Comput. Math. Appl.* **1983**, *9*, 617–624.
6. Gragg, W.B.; Stewart, G.W. A stable variant of the secant method for solving nonlinear equations. *SIAM J. Numer. Anal.* **1976**, *13*, 889–903.
7. Yang, X.S. Firefly algorithms for multimodal optimization. In *International Symposium on Stochastic Algorithms*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 169–178.
8. Johari, N.F.; Zain, A.M.; Noorfa, M.H.; Udin, A. Firefly algorithm for optimization problem. In *Applied Mechanics and Materials*; Trans Tech Publications Ltd.: Bäch, Switzerland, 2013; Volume 421, pp. 512–517.
9. Dorigo, M.; Stützle, T. The ant colony optimization metaheuristic: Algorithms, applications, and advances. In *Handbook of Metaheuristics*; Springer, Boston, MA, USA, 2003; pp. 250–285.
10. Zhao, W.; Wang, L. An effective bacterial foraging optimizer for global optimization. *Inf. Sci.* **2016**, *329*, 719–735.
11. Bolaji, A.L.A.; Al-Betar, M.A.; Awadallah, M.A.; Khader, A.T.; Abualigah, L.M. A comprehensive review: Krill Herd algorithm (KH) and its applications. *Appl. Soft Comput.* **2016**, *49*, 437–446.
12. Chu, S.C.; Tsai, P.W.; Pan, J.S. Cat swarm optimization. In *Pacific Rim International Conference on Artificial Intelligence*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 854–858.
13. Karaboga, D. *An Idea Based on Honey Bee Swarm for Numerical Optimization*; Technical Report-tr06; Erciyes University, Engineering Faculty, Computer Engineering Department: Melikgazi/Kayseri, Turkey, 2005; Volume 200, pp. 1–10.
14. Mirjalili, S. SCA: A sine cosine algorithm for solving optimization problems. *Knowl. Based Syst.* **2016**, *96*, 120–133.
15. El-Shorbagy, M.A.; Hassanien, A.E. Particle swarm optimization from theory to applications. *Int. J. Rough Sets Data Anal. (IJRSDA)* **2018**, *5*, 1–24.
16. El-Shorbagy, M.A.; Mousa, A.A. Chaotic particle swarm optimization for imprecise combined economic and emission dispatch problem. *Rev. Inf. Eng. Appl.* **2017**, *4*, 20–35.
17. Abd Allah, A.M.; El-Shorbagy, M.A. Enhanced particle swarm optimization based local search for reactive power compensation problem. *Appl. Math.* **2017**, *3*, 9.
18. Mousa, A.A.; El-Shorbagy, M.A.; Farag, M.A. K-means-clustering based evolutionary algorithm for multi-objective resource allocation problems. *Appl. Math. Inf. Sci.* **2017**, *11*, 1681–1692.
19. Abdelsalam, A.M.; El-Shorbagy, M.A. Optimization of wind turbines siting in a wind farm using genetic algorithm based local search. *Renew. Energy* **2018**, *123*, 748–755.
20. Saremi, S.; Mirjalili, S.; Lewis, A. Grasshopper optimisation algorithm: Theory and application. *Adv. Eng. Softw.* **2017**, *105*, 30–47.
21. Goel, R.; Maini, R. A hybrid of ant colony and firefly algorithms (HAFA) for solving vehicle routing problems. *J. Comput. Sci.* **2018**, *25*, 28–37.
22. Abd-El-Wahed, W.F.; Mousa, A.A.; El-Shorbagy, M.A. Integrating particle swarm optimization with genetic algorithms for solving nonlinear optimization problems. *J. Comput. Appl. Math.* **2011**, *235*, 1446–1453.
23. Al Malki, A.; Rizk, M.M.; El-Shorbagy, M.A.; Mousa, A.A. Hybrid genetic algorithm with K-means for clustering problems. *Open J. Optim.* **2016**, *5*, 71.
24. Nasr, S.M.; El-Shorbagy, M.A.; El-Desoky, I.M.; Hendawy, Z.M.; Mousa, A.A. Hybrid genetic algorithm for constrained nonlinear optimization problems. *J. Adv. Math. Comput. Sci.* **2015**, *7*, 466–480.
25. Wang, J.; Yang, W.; Du, P.; Niu, T. A novel hybrid forecasting system of wind speed based on a newly developed multi-objective sine cosine algorithm. *Energy Convers. Manag.* **2018**, *163*, 134–150.
26. Skoullis, V.I.; Tassopoulos, I.X.; Beligiannis, G.N. Solving the high school timetabling problem using a hybrid cat swarm optimization based algorithm. *Appl. Soft Comput.* **2017**, *52*, 277–289.
27. Naderi, E.; Azizivahed, A.; Asrari, A. A step toward cleaner energy production: A water saving-based optimization approach for economic dispatch in modern power systems. *Electr. Power Syst. Res.* **2022**, *204*, 107689.
28. Wu, J.; Cui, Z.; Liu, J. Using hybrid social emotional optimization algorithm with metropolis rule to solve nonlinear equations. In Proceedings of the IEEE 10th International Conference on Cognitive Informatics and Cognitive Computing (ICCI-CC'11), Banff, AB, Canada, 18–20 August 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 405–411.
29. Jaberipour, M.; Khorram, E.; Karimi, B. Particle swarm algorithm for solving systems of nonlinear equations. *Comput. Math. Appl.* **2011**, *62*, 566–576.
30. Wu, Z.; Kang, L. A fast and elitist parallel evolutionary algorithm for solving systems of non-linear equations. In Proceedings of the 2003 Congress on Evolutionary Computation, Canberra, ACT, Australia, 8–12 December 2003; IEEE: Piscataway, NJ, USA, 2003; CEC'03, Volume 2, pp. 1026–1028.

31. Dai, J.; Wu, G.; Wu, Y.; Zhu, G. Helicopter trim research based on hybrid genetic algorithm. In Proceedings of the2008 7th World Congress on Intelligent Control and Automation, Chongqing, China, 25–27 June 2008; IEEE: Piscataway, NJ, USA, 2008; pp. 2007–2011
32. Mo, Y.; Liu, H.; Wang, Q. Conjugate direction particle swarm optimization solving systems of nonlinear equations. *Comput. Math. Appl.* **2009**, *57*, 1877–1882.
33. dos Santos Coelho, L.; Mariani, V.C. Use of chaotic sequences in a biologically inspired algorithm for engineering design optimization. *Expert Syst. Appl.* **2008**, *34*, 1905–1913.
34. Zhang, L.; Zhang, C. Hopf bifurcation analysis of some hyperchaotic systems with time-delay controllers. *Kybernetika* **2008**, *44*, 35–42.
35. Yang, D.; Li, G.; Cheng, G. On the efficiency of chaos optimization algorithms for global optimization. *Chaos Solitons Fractals* **2007**, *34*, 1366–1375.
36. Abdullah, A.H.; Enayatifar, R.; Lee, M. A hybrid genetic algorithm and chaotic function model for image encryption. A*EU-Int. J. Electron. Commun.* **2012**, *66*, 806–816.
37. Jiang, B.L.W. Optimizing complex functions by chaos search. *Cybern. Syst.* **1998**, *29*, 409–419.
38. Saremi, S.; Mirjalili, S.; Lewis, A. Biogeography-based optimisation with chaos. *Neural Comput. Appl.* **2014**, *25*, 1077–1097.
39. Singh, H.K.; Alam, K.; Ray, T. Use of infeasible solutions during constrained evolutionary search: A short survey. In Proceedings of the Australasian Conference on Artificial Life and Computational Intelligence, Geelong, Australia, 31 January–2 February 2016; Springer: Cham, Switzerland, 2016; pp. 193–205.
40. 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.
41. Nie, P.Y. An SQP approach with line search for a system of nonlinear equations. *Math. Comput. Model.* **2006**, *43*, 368–373.
42. Nie, P.Y. A null space method for solving system of equations. *Appl. Math. Comput.* **2004**, *149*, 215–226.
43. Grosan, C.; Abraham, A. A new approach for solving nonlinear equations systems. *IEEE Trans. Syst. Man Cybern. Part A Syst. Hum.* **2008**, *38*, 698–714.
44. Chen, L.; McPhee, J.; Yeh, W.W.G. A diversified multiobjective GA for optimizing reservoir rule curves. *Adv. Water Resour.* **2007**, *30*, 1082–1093.
45. Pourrajabian, A.; Ebrahimi, R.; Mirzaei, M.; Shams, M. Applying genetic algorithms for solving nonlinear algebraic equations. *Appl. Math. Comput.* **2013**, *219*, 11483–11494.
46. Skiadas, C.H.; Skiadas, C. (Eds.) *Handbook of Applications of Chaos Theory*; CRC Press: Boca Raton, FL, USA, 2017.
47. Kaveh, A. *Advances in Metaheuristic Algorithms for Optimal Design of Structures*; Springer International Publishing: Cham, Switzerland, 2014; pp. 9–40.
48. Tavazoei, M.S.; Haeri, M. Comparison of different one-dimensional maps as chaotic search pattern in chaos optimization algorithms. *Appl. Math. Comput.* **2007**, *187*, 1076–1085.
49. Peitgen, H.O.; Jürgens, H.; Saupe, D. *Chaos and Fractals*. Springer: New York, NY, USA, 1992.
50. Li, Y.; Deng, S.; Xiao, D. A novel Hash algorithm construction based on chaotic neural network. *Neural Comput. Appl.* **2011**, *20*, 133–141.
51. May, R.M. Simple mathematical models with very complicated dynamics. In *Theory Chaotic Attractors*; Springer, New York, NY, USA, 2004; pp. 85–93.
52. Devaney, R.L.; Wiggins, S. A review of: "An introduction to chaotic dynamical systems" (benjamin/cummings, menlo park 1987, 1986). *Transp. Theory Stat. Phys.* **1987**, *16*, 1177–1180.
53. Hilborn, R.C. *Chaos and Nonlinear Dynamics: An Introduction for Scientists and Engineers*; Oxford University Press on Demand: Oxford, England, 2000.
54. Arora, J.S.; Elwakeil, O.A.; Chahande, A.I.; Hsieh, C.C. Global optimization methods for engineering applications: A review. *Struct. Optim.* **1995**, *9*, 137–159.
55. Erramilli, A.; Singh, R.P.; Pruthi, P. *Modeling Packet Traffic with Chaotic Maps*; KTH: Stockholm, Sweden, 1994.
56. He, D.; He, C.; Jiang, L.G.; Zhu, H.W.; Hu, G.R. Chaotic characteristics of a one-dimensional iterative map with infinite collapses. *IEEE Trans. Circuits Syst. I Fundam. Theory Appl.* **2001**, *48*, 900–906.
57. Ott, E. *Chaos in Dynamical Systems*; Cambridge University Press: Cambridge, UK, 2002.
58. El-Shorbagy, M.A.; Mousa, A.A.; Nasr, S.M. A chaos-based evolutionary algorithm for general nonlinear programming problems. *Chaos Solitons Fractals* **2016**, *85*, 8–21.
59. Rashedi, E.; Nezamabadi-Pour, H.; Saryazdi, S. GSA: A gravitational search algorithm. *Inf. Sci.* **2009**, *179*, 2232–2248.
60. Sun, J.; Feng, B.; Xu, W. Particle swarm optimization with particles having quantum behavior. In Proceedings of the 2004 Congress on Evolutionary Computation, Portland, OR, USA, 19–23 June 2004; IEEE Cat. No. 04TH8753; IEEE: Piscataway, NJ, USA, 2004; Volume 1, pp. 325–331.
61. Sun, J.; Xu, W.; Feng, B. Adaptive parameter control for quantum-behaved particle swarm optimization on individual level. In Proceedings of the 2005 IEEE International Conference on Systems, Man and Cybernetics, Waikoloa, HI, USA, 12 October 2005; IEEE: Piscataway, NJ, USA, 2005; Volume 4, pp. 3049–3054.
62. Yadav, P.; Kumar, R.; Panda, S.K.; Chang, C.S. An intelligent tuned harmony search algorithm for optimisation. *Inf. Sci.* **2012**, *196*, 47–72.

63. Turgut, O.E.; Turgut, M.S.; Coban, M.T. Chaotic quantum behaved particle swarm optimization algorithm for solving nonlinear system of equations. *Comput. Math. Appl.* **2014**, *68*, 508–530.
64. Floudas, C.A., Pardalos, P.M., Adjiman, C., Esposito, W.R., Gümüs, Z.H., Harding, S.T., Klepeis, J.L., Meyer, C.A.; Schweiger, C.A. *Handbook of Test Problems in Local and Global Optimization*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2013, Volume 33.
65. e Oliveira, H.A., Jr.; Petraglia, A. Solving nonlinear systems of functional equations with fuzzy adaptive simulated annealing. *Appl. Soft Comput.* **2013**, *13*, 4349–4357.
66. Abdollahi, M.; Isazadeh, A.; Abdollahi, D. Imperialist competitive algorithm for solving systems of nonlinear equations. *Comput. Math. Appl.* **2013**, *65*, 1894–1908.
67. Wang, C.; Luo, R.; Wu, K.; Han, B. A new filled function method for an unconstrained nonlinear equation. *J. Comput. Appl. Math.* **2011**, *235*, 1689–1699.
68. Luo, Y.Z.; Tang, G.J.; Zhou, L.N. Hybrid approach for solving systems of nonlinear equations using chaos optimization and quasi-Newton method. *Appl. Soft Comput.* **2008**, *8*, 1068–1073.
69. Mageshvaran, R.; Raglend, I.J.; Yuvaraj, V.; Rizwankhan, P.G.; Vijayakumar, T. Implementation of non-traditional optimization techniques (PSO, CPSO, HDE) for the optimal load flow solution. In Proceedings of the TENCON 2008 IEEE Region 10 Conference, Hyderabad, India, 19–21 November 2008; IEEE: Piscataway, NJ, USA, 2018; pp. 1–6.
70. Saadat, H. *Power System Analysis*, 2nd ed.; McGraw-Hill Higher Education: Singapore, 2009.
71. Milano, F. Continuous Newton's method for power flow analysis. *IEEE Trans. Power Syst.* **2008**, *24*, 50–57.
72. Rizk-Allah, R.M. A quantum-based sine cosine algorithm for solving general systems of nonlinear equations. *Artif. Intell. Rev.* **2021**, *54*, 3939–3990.