processes

Article

Chaotic Search Based Equilibrium Optimizer for Dealing with Nonlinear Programming and Petrochemical Application

Abd Allah A. Mousa [1,*], Mohammed A. El-Shorbagy [2,3], Ibrahim Mustafa [4] and Hammad Alotaibi [1]

[1] Department of Mathematics and Statistics, College of Science, Taif University, P.O. Box 11099, Taif 21944, Saudi Arabia; hm.alotaibi@tu.edu.sa
[2] Department of Mathematics, College of Science and Humanities in Al-Kharj, Prince Sattam bin Abdulaziz University, Al-Kharj 11942, Saudi Arabia; ma.hassan@psau.edu.sa
[3] Department of Basic Engineering Science, Faculty of Engineering, Menofia University, Shebin El-Kom 32511, Egypt
[4] Department of Biomedical Engineering, Faculty of Engineering at Helwan, Helwan University, Cairo, Helwan 11795, Egypt; ibm_hassan@yahoo.com
* Correspondence: a.mousa@tu.edu.sa

Abstract: In this article, chaotic search based constrained equilibrium optimizer algorithm (CS-CEOA) is suggested by integrating a novel heuristic approach called equilibrium optimizer with a chaos theory-based local search algorithm for solving general non-linear programming. CS-CEOA is consists of two phases, the first one (phase I) aims to detect an approximate solution, avoiding being stuck in local minima. In phase II, the chaos-based search algorithm improves local search performance to obtain the best optimal solution. For every infeasible solution, repair function is implemented in a way such that, a new feasible solution is created on the line segment defined by a feasible reference point and the infeasible solution itself. Due to the fast globally converging of evolutionary algorithms and the chaotic search's exhaustive search, CS-CEOA could locate the true optimal solution by applying an exhaustive local search for a limited area defined from Phase I. The efficiency of CS-CEOA is studied over multi-suites of benchmark problems including constrained, unconstrained, CEC'05 problems, and an application of blending four ingredients, three feed streams, one tank, and two products to create some certain products with specific chemical properties, also to satisfy the target costs. The results were compared with the standard evolutionary algorithms as PSO and GA, and many hybrid algorithms in the same simulation environment to approve its superiority of detecting the optimal solution over selected counterparts.

Keywords: chaotic mapping; constrained optimization; equilibrium optimizer; non-linear optimization; petrochemical engineering application

check for updates

Citation: Mousa, A.A.A.; El-Shorbagy, M.A.; Mustafa, I.; Alotaibi, H. Chaotic Search Based Equilibrium Optimizer for Dealing with Nonlinear Programming and Petrochemical Application. *Processes* **2021**, *9*, 200. https://doi.org/10.3390/pr9020200

Received: 13 December 2020
Accepted: 18 January 2021
Published: 21 January 2021

1. Introduction

An exhaustive investigation of both theoretical and practical areas of the constrained non-linear programming problems (CNPPs) can be the subject matter of this paper. CNPPs have many characteristics, such as non-differentiable, non-convex, unimodal, and multi-modal. Owing to the complexities of the CNPPs that often occur, researchers are trying to implement efficient optimizers to deal with non-linear programming problems NLP.

From the view of mathematical optimization methods, there are two main classifications: (1) deterministic optimization techniques, and (2) stochastic optimization techniques. Linear programming and non-linear programming methods [1,2] are of the most common deterministic methods that are used by searching for space and finding a solution using problem gradient knowledge. These methods are useful for problems with linear search areas (unimodal functions), but in problems having non-linear search areas, like real-world applications with non-convex formulation, they are vulnerable to local optima impairment [3,4]. This problem can be combated by modifying or hybridizing the algorithm [5] with different initial design. Another alternative method for these traditional methods is the stochastic-based optimization methods that implement random variables. These methods are used to explore the search space globally to detect optimal global or optimal solution close-to-global solution. Their advantages include simplicity, independence, problem flexibility, and non-gradient nature [6].

Among the existing stochastic methods, famous algorithms such as genetic algorithms based approaches [7–10], artificial immune system [11], neural network-based methods [12], particle swarm based methods [13–15], ant colony based methods [16], artificial bee colony based methods [17], bacterial foraging based algorithm (BFA) [18,19], cat swarm based optimization algorithm (CSO) [20], glowworm swarm based optimization algorithm(GSOA) [21], firefly-based optimization algorithm (FOA) [22], monkey-based algorithm (MA) [23], krill herd algorithm (KHA) [24], cuckoo search based algorithm [25], whale optimization algorithm (WOA) [26], sine cosine algorithm [27,28], grasshopper based optimization algorithm (GOA) [29], salp swarm based algorithm [30], equilibrium optimizer based optimization algorithm (EOA) [31], gradient-based optimizer (GBO) [32], slime mold-based algorithm (SMA) [33], and Harris hawks optimization (HHO) [34], and others.

There are many stochastic-based methods that have recently been used to deal with CNPPs such as a carnivorous plant algorithm [35], modified Sine Cosine Algorithm [36] enhanced a modified SCA with a novel mutation operator, and a transition parameter, water turbulent flow optimization (TFWO) algorithm [37]. Chaos mechanism based on quasi-opposition was presented [38], an ABC algorithm with adaptive heterogeneous competition [39], an improved FOA [40], Bare-Bones Based SCA [41], Group teaching optimization algorithm (GTOA) [42], Political Optimizer (PO) [43], Refined selfish herd optimizer [44], etc.

Many scientists and researchers investigated the hybridization between chaos theory and evolutionary optimization techniques to enhance optimization algorithm performance. Wei et al. [45] implemented the tent chaotic map to randomly generate the initial population for the genetic algorithm to guarantee well-distributed throughout the search space. Dashet al. [46] presented a novel hybrid evolutionary swarm algorithm by combining conditional mutual information maximization with a chaotic firefly approach. Fuertes et al. [47] proposed a new contribution to the chaos-based genetic algorithm and they investigated the entropy effectiveness in the initial population. Mousa et al. [48] presented a hybrid evolutionary algorithm based on the sinusoidal chaotic mapping. El-Shorbagy et al. [49] presented an interesting comparison between 14 chaotic mappings representing chaotic local search. Abo-Elnaga et al. [50] presented a chaos local Search based genetic algorithm for dealing with bilevel programming problems.

EOA is a novel optimization approach that simulates control volume mass balance models, which implemented to determine both dynamic and equilibrium states was proposed by Faramarzi et al. [31]. Each particle (solution) represents a search agent with its position (concentration) in the EOA. In order to ultimately achieve an optimal balance (optimum result), search agents randomly update their attention on the best available choices, i.e., matching applicants. It has been shown that the term "generation rate" improves the capability of EOA to escape local minima and to establish a balance between exploitation operator and exploration operator.

The rising literature shows that EOA is becoming more common in different fields. For example, a binary EOA for 0–1 knapsack problem had been proposed in [51]. While an effective EOA with a mutation strategy for numerical optimization had been provided in [52]. In addition, it was used to calculate the optimal estimate of Schottky diode parameters [53], to determine the solar photovoltaic parameter [54], and to reconfigure and distribute generation in power systems [55], etc. The efficacy of EOA enables multidisciplinary researchers to further improve its applicability. There are three ways to strengthen the initial EOA as follows:

1. Changing EOA parameters or algorithmic procedures to boost algorithm performance;
2. The development of the EOA by modern learning methods to improve the use of information;
3. Hybridization of the EOA by other search methods;
4. Combining EOA with chaotic search methods.

In this study, CS-CEOA is suggested to solve non-linear programming and an application of petrochemical engineering. CS-CEOA is an integration between a chaos-based local search algorithm, and a new heuristic approach called equilibrium optimizer algorithm (EOA). The principle of co-evolution, reparation, elitism, and chaotic search are the main features of the proposed method. The repair method was implemented to co-evolves any infeasible solution until it becomes feasible, in a way such that, a new feasible solution is created on the segment defined by the feasible reference point and the unfeasible solution itself. The elitist strategy is used to elite the best-found solution all the generation, which gives the proposed algorithm a faster convergence to the optimal solution, while the chaotic search increasing the CS-CEOA capability to get the global solution. CS-CEOA is examined using a set of the most well-known benchmark test problems "CEC'05" and eight constrained benchmark problems elicited from the literature [56,57]. Further, the proposed algorithm is implemented in solving an application of blending four ingredients, three feed streams, one tank, and two products to obtain certain products with certain (required) chemical properties and determining costs. The efficiency of our algorithm was achieved compared with other algorithms in the literature.

This paper is structured as follows. In Section 2, we explain the standard formulation of the constrained non-linear programming problems. The suggested algorithm is investigated in Section 3. Section 4 addresses the simulation experiments. The limitation of the proposed study is presented in Section 5, Finally, our observations and future work are discussed in Section 6.

2. Constrained Non-linear Programming Problem (CNPP)

In mathematics, a constrained non-linear programming problem (CNPP) is the process of handling an optimization problem where any of the constraints or objective function are nonlinear. Linear programming problem is a special case of NPP.

The general CNPP is written generally as [58]:

$$\begin{aligned} &\min \quad f(x) \\ &\textit{subject to} \\ &\qquad c_m(x) = 0,\ m \in E, \\ &\qquad c_m(x) \geq 0,\ m \in \mathcal{I}, \\ &\qquad l_i \leq x_i \leq u_i,\ i \in 1,\ldots,n \end{aligned} \tag{1}$$

where $x \in R^n$ are the decision variables, $l \in R^n$, $u \in R^n$ represent lower bounds and upper bounds of the decision variables, $|\varepsilon|$ is the set of equality constraints and $|\mathcal{I}|$ is the set of inequality constraints, the function f is the objective function, and $c_m \forall m \in \mathcal{E} \cup \mathcal{I}$ are the set of constraint functions, the functions $f, c_m \forall m$ are mapping from R^n to R.

3. The Suggested Algorithm (CS-CEOA)

In this section, the suggested algorithm, a chaotic search based on a constrained equilibrium optimizer algorithm (CS-CEOA) is presented.

3.1. Brief Discribtion of Equilibrium Optimizer Algorithm

The equilibrium optimizer algorithm (EOA) is a simulated optimizer that was originally presented by Faramarzi [31] in 2020. The simulated optimizer simulates the equilibrium and dynamic m states related to the mass balance models where each particle concentration (particle position) is updated in a random way with a target of reaching the equilibrium state (particle fitness). The equilibrium optimizer has a very simple procedure, also it has an adaptive dynamic control parameter. It is initialized with initial positions of the particles (initial positions C_i, i = 1, 2, ..., No. of Particles$\frac{1}{2}$) with a special number (No. of particles) and problem's dimensions (dim) as in the following equation:

$$C_{initial} = \text{rand}(\text{No. of particles}, \text{dim}) \times (ub - lb) + lb, \tag{2}$$

where $C_{initial}$ locates the initial positions of the particles; the decision variables bounds lb and ub are the specified lower and upper bounds respectively of the decision optimization variables.

- Equilibrium pool and candidates (Ceq)

The terminology of the equilibrium state is called the final convergence state of EOA. At the initialization of the algorithm, equilibrium candidates are assigned to support a search pattern for the particles. There are four best-so-far particles identified during the algorithm optimization process with another particle, whose position is the arithmetic mean of the other four particles. EOA has an exploration scheme using four candidates and an exploitation scheme using the average mean. These five particles are called equilibrium candidates which are used to construct the equilibrium pool:

$$C_{eq,pool} = \left(C_{eq,1}, C_{eq,2}, C_{eq,3}, C_{eq,4}, C_{eq,av}\right), \tag{3}$$

The position of every particle in each iteration of the whole algorithm is updated using an equilibrium pool by random selection among candidates chosen with the same probability. Then, the particle positions are repeatedly updated with respect to the equilibrium pool, which is extracted as the best-so-far candidates. The procedure of updating the mechanism of the EO as in the following equation:

$$C_{new} = C_{eq} + \frac{G}{\lambda}(1 - F) + (C_{old} - C_{eq}) \times F, \tag{4}$$

$$F = a_1 sign(r - 0.5)(e^{-\lambda t} - 1), \tag{5}$$

$$G = \begin{cases} 0.5r_1 & \text{if } r_2 \geq GP \\ 0 & \text{if } r_2 < GP \end{cases}, \tag{6}$$

$$t = \left(1 - \frac{T}{T_{\max}}\right)^{a_2 \frac{T}{T_{\max}}}; \tag{7}$$

where C_{old} is the current position (concentration) vector, and C_{new} is the new updated position vectors of the particle? From the equilibrium pool, we randomly pick one concentration vector which denoted by C_{eq}. λ is a random vector between 0 and 1; a1 and a2 are constants ($a_1 = 2$ and $a_1 = 1$), r, r_1, r_2 are random numbers generated between 0 and 1, GP is the generation probability, T is the current iteration counter and $T_{\max}$ is the predetermined maximum number of the iterations. In each generation repetition, the problem objective function is calculated for each particle's position to determine their states. In addition, the equilibrium pool $C_{eq,pool} = \left(C_{eq,1}, C_{eq,2}, C_{eq,3}, C_{eq,4}, C_{eq,av}\right)$ is updated each iteration to contain the four best so far particles.

3.2. Basic Algorithm

The combined algorithm CS-CEOA is constructed of two phases, the first one (phase I) aims to locate the approximate solution, avoiding being stuck in local minima. In phase II, the chaos-based search algorithm increases CS-CEOA's performance and obtain the best optimal solution. CEOA's main steps are defined as follows:

- Phase I: Constrained equilibrium optimizer algorithm

Step 1. Initialization stage: Initial population in the first generation are randomly initialized according to Equation (1).

Step 2. Initial feasible particle: The algorithm requires to get at least one initial feasible reference point (satisfying the set of constraints) to evolve the algorithm process. If the algorithm has difficulties in finding such an initial reference point (RP), the algorithm shall implement one of the following two ways: (1) doubling the number of tests to obtain the initial reference point, or (2) increasing temporally feasible space [59].

Step 3. Repairing infeasible particles: This step co-evolves any infeasible solution until it becomes feasible. A feasible solution is created on the segment defined by the feasible reference point and the infeasible solution [60].

Step 4. Elitist strategy for selection: To make the algorithm converge faster to the optimal solution, using the elitist strategy. The elitist particle represents the best solution for the population. By using an elitist solution, the best fitness particle can never be increased from one generation to the next until the optimization process is over.

Step 5. Evolution process stage: The algorithm applies EOA procedures to create a new population using Equations (4)–(7).

Step 6. Stopping criteria: The proposed algorithm is stopped for any of the following two conditions:

- Reaching the maximum predetermined number of generations $T_{\max}$.
- When the population's particles converge. Particle convergence happens when all solutions in the population are similar.

Optimization by using phase-I yields an approximate solution $x^* = \left(x_1^*, x_2^*, \ldots, x_n^*\right)$ close to its true global solution. Chaotic local search (CLS) has the capability to perturb the position x^*; where local zone around x^* will be exhaustively explored. There are various chaotic maps that have been used in optimization algorithms to enhance their efficiency. In the suggested algorithm, the chaotic circle map was used in the CLS Phase. The detailed procedure of the CLS scheme are presented as follows:

- Phase II: Chaotic local search (CLS):

Step 1. Determine the range of CLS $[a_i, b_i],\ i = 1, 2, .., n$ by $x_i^* - \varepsilon > a_i,\ x_i^* + \varepsilon < b_i$; where ε is the predetermined radius of chaotic local search.

Step 2. Chaotic random numbers z^L are generated using the chaotic circle map; where $\alpha = 0.5,\ \beta = 0.2$ as follows:

$$z^{L+1} = z^L + \beta - (\alpha - 2\pi)\sin\left(2\pi z^L\right) Mod(1), \tag{8}$$

where L is the CLS iterations and *Mod* is a mathematical function, that returns the remainder or signed remainder of a division after one number is divided by another.

Step 3. Map the chaos variable z^L into the decision variables range of optimization valuable $[a_i, b_i]$ by

$$x_i^L = a_i + (b_i - a_i)z^L, \tag{9}$$

By substituting the value of $a_i = x_i^* - \varepsilon$ and $b_i = x_i^* + \varepsilon$, then Equation (9) can be rewritten as:

$$x_i^L = x_i^* - \varepsilon + 2\varepsilon z^L \quad \forall i = 1, \ldots, n, \tag{10}$$

Step 4. If $f\left(x^L\right) < f(x^*)$ then set $x^* = x^L$, otherwise break the iteration.

Step 5. If $f(x^*)$ has not been improved for all L iterations, terminate chaos search algorithm and put out x^* as the best optimal global solution.

The proposed algorithm is said to have convergence if

$$\frac{\|X_{T+1} - X^*\|}{\|X_T - X^*\|} \le \tau,\ \tau \ge 0, \tag{11}$$

where X_T and X_{T+1} denote the solutions obtained at the end of iterations T and T + 1, respectively, X^* represents the optimum solution, and $\|X\|$ denotes the length or norm of the vector X. The proposed optimization method is said to have super-linear convergence (corresponds to fast convergence) if:

$$\lim_{T\to\infty} \frac{\|X_{T+1} - X^*\|}{\|X_T - X^*\|} \to 0 \tag{12}$$

The pseudo-code of chaotic is illustrating local search is declared in Figure 1, while the flow chart of the proposed algorithm is shown in Figure 2.

```
CS Procedure, given $x^* = (x_1^*, x_2^*, ...., x_n^*)$, $\varepsilon$ and $z^0$.
    While: $f(x^*)$ is improved
        $L \leftarrow 1$
        Generate $z^k$ using Chaotic circle map, $x_i^L = x_i^* - \varepsilon + 2\varepsilon z^L$    $\forall i = 1, ..., n$
            If $f(x^L) < f(x^*)$ then $x^* = x^k$
            Else if $f(x^L) \geq f(x^*)$ continue,
            End if
        If termination criteria satisfied,
        Break
        End if
          $L \leftarrow L + 1$
    End while
End
```

Figure 1. The pseudo-code of the chaotic local search (CLS).

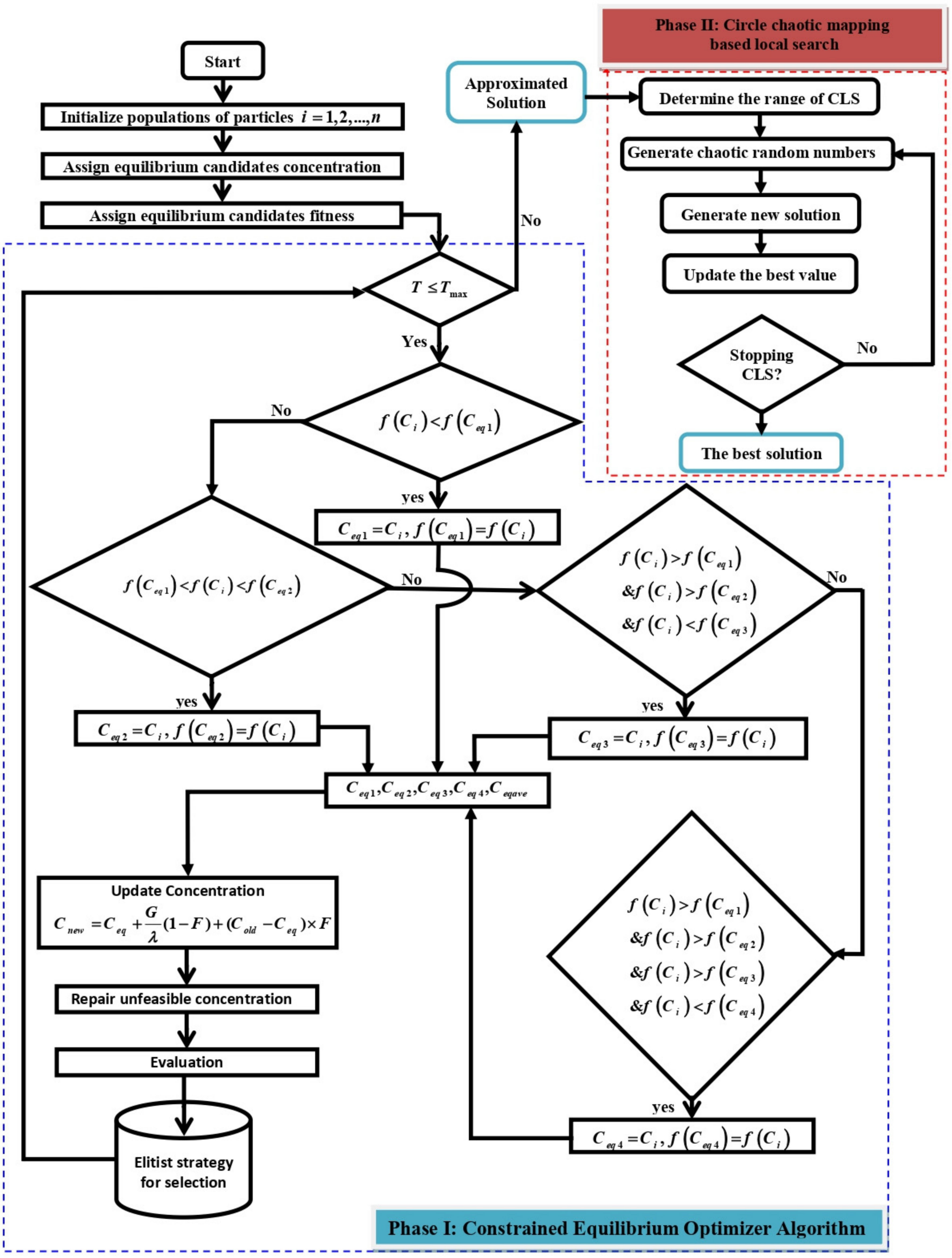
Phase II: Circle chaotic mapping based local search
Start
Initialize populations of particles $i = 1,2,...,n$
Assign equilibrium candidates concentration
Assign equilibrium candidates fitness
Approximated Solution
Determine the range of CLS
Generate chaotic random numbers
Generate new solution
Update the best value
Stopping CLS?
No
The best solution
No
$T \le T_{\max}$
Yes
$f(C_i) < f(C_{eq1})$
No
yes
$C_{eq1} = C_i, f(C_{eq1}) = f(C_i)$
$f(C_{eq1}) < f(C_i) < f(C_{eq2})$
No
$f(C_i) > f(C_{eq1})$ & $f(C_i) > f(C_{eq2})$ & $f(C_i) < f(C_{eq3})$
No
yes
$C_{eq2} = C_i, f(C_{eq2}) = f(C_i)$
yes
$C_{eq3} = C_i, f(C_{eq3}) = f(C_i)$
$C_{eq1}, C_{eq2}, C_{eq3}, C_{eq4}, C_{eqave}$
Update Concentration
$C_{new} = C_{eq} + \frac{G}{\lambda}(1-F) + (C_{old} - C_{eq}) \times F$
Repair unfeasible concentration
Evaluation
Elitist strategy for selection
$f(C_i) > f(C_{eq1})$ & $f(C_i) > f(C_{eq2})$ & $f(C_i) > f(C_{eq3})$ & $f(C_i) < f(C_{eq4})$
yes
$C_{eq4} = C_i, f(C_{eq4}) = f(C_i)$
Phase I: Constrained Equilibrium Optimizer Algorithm

Figure 2. Flowchart of the proposed algorithm.

4. Experimental Findings

This section is developed to validate the proposed algorithm to handle the non-linear programming problems and to petrochemical engineering application; where it is tested by a set of well-known benchmark test problems "CEC'05", set of eight constrained benchmark test problems [56,57], and petrochemical engineering application. The efficiency of our algorithm is achieved compared with other recent algorithms in the literature. All the experiments are coded in Matlab 14.0, and the numerical simulations are done on an Intel Core machine (Intel i7, 2.9 GHz, 16 GB DDR4 RAM). The controlled parameters of the proposed algorithm are shown in Table 1.

Table 1. The parameters setting.

Constrained Equilibrium Optimizer Algorithm		Chaotic Local Search (CLS)	
Parameter	**Value**	**Parameter**	**Value/Description**
Number of Particles	50	Chaotic Mapping	Chaotic Circle
The maximum number of iterations ($T_{\max}$)	100	Specified neighborhood radius (ε)	1×10^{-6}
Probability of generation (GP)	0.5	α, β	0.5, 0.2
a_1, a_2	2, 1	Chaos search iteration (L)	100

4.1. Benchmark Unconstrained Problem Suite

This subsection focuses on the reliability and robustness of the proposed algorithm (CS-CEOA) evaluated by 17 unconstrained benchmark functions. The results are compared against, integrated particle swarm with genetic algorithms (Integrated PSO-GAs) [15], a hybrid optimization algorithm from PSO, and GA (H_PSO_GA) [61], a continuous genetic algorithm (CGA) [62], continuous hybrid algorithm CHA [63], PSO based hybrid GA (GA-PSO) [64] and the original constrained equilibrium optimizer algorithm (CEOA). To avoid biasing the optimization results to the random of the initial population and to make unbiased comparisons, we run each problem 30 times, starting with various randomly selected positions in the hyperrectangular search space. The numerical comparison between the results calculated by the proposed algorithm versus the global optimal solutions is shown in Table 2. While Table 3 illustrates the obtained experimental results using the proposed algorithm versus five recent evolutionary algorithms according to average error. The numerical simulations have demonstrated the superiority of the proposed approach to locating the global optimal solution.

Table 2. Calculated solution versus Global optimal solution.

Test Problem	F Optimal	Integrated PSO-GAs	H_PSO_GA	CEOA	CS-CEOA
RC	0.397887	0.397887	0.397887	0.398019	0.397887
B2	0	0	0	0.001785	0
ES	−1	−1	−1	−0.999993	−1
GP	3	3	3	3.0000478	3
SH	−186.7309	−186.7309	−186.7309	−186.6298	−186.731
DJ	0	2.6022×10^{-64}	0	9.3799×10^{-53}	0
$H_{3,4}$	−3.86278	−3.86343347	−3.86343347787	−3.861023	−3.86278
$H_{6,4}$	−3.32237	−3.322368	−3.322368	−3.32226	−3.32237
$S_{4,5}$	−10.1532	−10.1532	−10.1532	−10.0487	−10.1532
$S_{4,7}$	−10.40294	−10.402916	−10.40291634	−10.179683	−10.403
$S_{4,10}$	−10.53641	−10.5363855	−10.53638559	−9.998507	−10.5365
R_2	0	1.38584×10^{-21}	1.5061×10^{-24}	3.50704×10^{-13}	-1×10^{-30}
R_5	0	1.7476×10^{-11}	1.7634×10^{-13}	8.4143×10^{-4}	0
R_{10}	0	1.1367×10^{-9}	2.3369×10^{-13}	6.4923×10^{-5}	0
Z_2	0	1.8461×10^{-18}	0	4.2805×10^{-14}	0
Z_5	0	3.8176×10^{-9}	0	6.1409×10^{-6}	0
Z_{10}	0	2.0996×10^{-9}	0	5.2118×10^{-7}	0

Table 3. Results provided by CS-CEOA, CEOA, H_PSO_GA, Integrated PSO-GAs, CGA, CHA and GA-PSO.

Test Function	Average Error						
	CS-CEOA	**CEOA**	**H_PSO_GA [61]**	**Integrated PSO-GAs [15]**	**CGA [62]**	**CHA [63]**	**GA-PSO [64]**
RC	0.0	0.0	0.0	4.59×10^{-7}	0.0001	0.0001	0.00009
B2	0.0	0.0	0.0	1×10^{-25}	0.0003	0.0000002	0.00001
ES	0.0	0.0	0.0	1×10^{-30}	0.0010	0.0010	0.00003
GP	0.0	0.0	0.0	-6.3060×10^{-14}	0.0010	0.0010	0.00012
SH	0.0	0.0	0.0	8.83064×10^{-6}	0.0050	0.0050	0.00007
DJ	0.0	0.0	0.0	8.443663×10^{-15}	0.0002	0.0002	0.00004
$H_{3,4}$	3×10^{-6}	3×10^{-6}	0.00002	0.00003	0.0050	0.0050	0.00020
$H_{6,4}$	4×10^{-8}	4×10^{-8}	5×10^{-7}	2×10^{-6}	0.0400	0.0080	0.00024
$S_{4,5}$	0.0	0.0	0.0	0.0	0.1400	0.0090	0.00014
$S_{4,7}$	0.000017	0.000017	0.000013	0.00002	0.1200	0.0100	0.00015
$S_{4,10}$	0.000091	0.000091	0.000011	0.00002	0.1500	0.0150	0.00012
R_2	1×10^{-30}	1×10^{-30}	1×10^{-32}	1×10^{-30}	0.0040	0.0040	0.00064
R_5	0.0	0.0	1×10^{-25}	1×10^{-20}	0.1500	0.0180	0.00013
R_{10}	0.0	0.0	1×10^{-20}	1×10^{-18}	0.0200	0.0080	0.00005
Z_2	0.0	0.0	0.0	1×10^{-15}	0.000003	0.000003	0.00005
Z_5	0.0	0.0	0.0	1×10^{-17}	0.0004	0.00006	0.00000
Z_{10}	0.0	0.0	0.0	1×10^{-25}	0.000001	0.000001	0.00000

4.2. Benchmark Constrained Problem Suite

This subsection focuses on the reliability, robustness, and ability of the CS-CEOA to solve constraining problems as it is evaluated through 8 constrained standard functions [57]. For comparison, we have chosen the constrained PSO algorithm according to [57]. Table 4 shows a comparison between the constrained PSO algorithm [57], the original constrained equilibrium optimizer algorithm (CEOA), and our approach CS-CEOA according to the absolute error. It is observed that CS-CEOA optimized the constrained problems effectively; where the average error of our solutions is less than that obtained by the constrained PSO algorithm in most problems.

Table 4. Error provided by constrained PSO, CEOA, and CS-CEOA.

Benchmark Problem	Error = ∣Optimal Value–Best Found Value∣		
	CS-CEOA	**CEOA**	**Constrained PSO [57]**
P_{-1}	10×10^{-30}	10×10^{-17}	5×10^{-4}
P_{-2}	1×10^{-12}	1×10^{-4}	2×10^{-5}
P_{-3}	0.00	0.00	0.00
P_{-4}	10×10^{-3}	0.01	1.76
P_{-5}	0.00	10×10^{-9}	0.00
P_{-6}	0.00	10×10^{-8}	0.00
P_{-7}	0.00	10×10^{-5}	0.00
P_{-8}	0.00	10×10^{-11}	0.00

Additionally, by comparing the proposed algorithm (CS-CEOA) with the original CEOA, it can be shown that chaotic search (CS) improves outcomes for both unconstrained benchmark problem suite (Tables 2 and 3) and constrained benchmark problem suite (Table 4). On the other hand, implementing chaotic local search influences most significantly the algorithm convergence time, saving up to 12% of the time without affecting the result accuracy.

4.3. CEC 2005 Benchmark Unconstrained Problems

The proposed approach is tested by 25 problems the set of CEC'05 "special session 2005 on real-parameter optimization problems" [56]. Table 5 shows the comparison results between the average error obtained by CS-CEOA and the other nine reported optimization algorithms in the literature [65–74]; where all reported algorithms have been run fifty times for each test problem. The algorithm stops either when the maximal number of evaluations (1×10^5) is achieved, or when the obtained error is less than 1×10^{-8}, or. Further, for each problem, we ranked the various methods according to the average error values obtained, as in Tables 6 and 7. Figure 3 shows the relative weight of each algorithm, which computed according to its rank. On the other hand, Figure 4 shows the comparison of different problems between the different algorithms according to their ranks. Overall, the proposed algorithm CS-CEOA performs well on almost all the test problems used for this suite.

Table 5. Average error of CEC'05 obtained by CS-CEOA versus other optimization algorithms.

Problem	PSO [65]	IPOP-CMA-E [66]	CHC [67,68]	SSGA [69,70]	SS-BLX [71]	SS-Arit [72]	DE-Bin [73]	DE-Exp [73]	SaDE [74]	CS-CEOA
F_1	1.23×10^{-4}	0	2.46	8.42×10^{-9}	3.40×10^1	1.06	7.72×10^{-9}	8.26×10^{-9}	8.42×10^{-9}	0.000
F_2	2.60×10^{-2}	0	1.18×10^2	8.72×10^{-5}	1.73	5.28	8.34×10^{-9}	8.18×10^{-9}	8.21×10^{-9}	0.000
F_3	5.17×10^4	0	2.70×10^5	7.95×10^4	1.84×10^5	2.54×10^5	4.23×10^1	9.94×10^1	6.56×10^3	0.000
F_4	2.488	2.93×10^3	9.19×10^1	2.59×10^{-3}	6.23	5.76	7.69×10^{-9}	8.35×10^{-9}	8.09×10^{-9}	0.000
F_5	4.10×10^2	8.10×10^{-10}	2.64×10^2	1.34×10^2	2.19	1.44×10^1	8.61×10^{-9}	8.51×10^{-9}	8.64×10^{-9}	0.000
F_6	7.31×10^2	0	1.42×10^6	6.17	1.15×10^2	4.95×10^2	7.96×10^{-9}	8.39×10^{-9}	1.61×10^{-2}	0.000
F_7	2.68×10^1	1.27×10^3	1.27×10^3	1.27×10^3	1.97×10^3	1.91×10^3	1.27×10^3	1.27×10^3	1.26×10^3	1.6231
F_8	2.043×10^1	2.00×10^1	2.03×10^1	2.04×10^1	2.04×10^1	2.04×10^1	2.03	2.04×10^{31}	2.03×10^1	2.025
F_9	1.44×10^1	2.84×10^1	5.89	7.29×10^{-9}	4.20	5.96	4.55	8.15×10^{-9}	8.33×10^{-9}	5.523×10^{-9}
F_10	1.40×10^1	2.33×10^1	7.12	1.71×10^1	1.24×10^1	2.18×10^1	1.23×10^{31}	1.12+31	1.55×10^1	1.7632
F_11	5.590	1.34	1.60	3.26	2.93	2.86	2.43	2.07	6.80	1.9390
F_12	6.36×10^2	2.13×10^2	7.06×10^2	2.79×10^2	1.51×10^2	2.41×10^2	1.06×10^2	6.31×10^1	5.63×10^1	5.98530
F_13	1.503	1.13	8.30×10^1	6.71×10^1	3.25×10^1	5.48×10^1	1.57	6.40×10^1	7.07×10^1	1.4434
F_14	3.304	3.78	2.07×10^1	2.26	2.80	2.97	3.07	3.16	3.42	2.7518
F_15	3.398×10^2	1.93×10^2	2.75×10^2	2.92×10^2	1.14×10^2	1.29×10^2	3.72×10^2	2.94×10^2	8.42×10^1	7.124×10^3
F_16	1.33×10^2	1.17×10^2	9.73×10^1	1.05×10^2	1.04×10^2	1.13×10^2	1.12×10^2	1.13×10^2	1.23×10^2	1179×10^2
F_17	1.50×10^2	3.39×10^2	1.05×10^2	1.19×10^2	1.18×10^2	1.28×10^2	1.42×10^2	1.31×10^2	1.39×10^2	1.269×10^2
F_18	8.51×10^2	5.57×10^2	8.80×10^2	8.06×10^2	7.67×10^2	6.58×10^2	5.10×10^2	4.48×10^2	5.32×10^2	4.043×10^2
F_19	8.50×10^2	5.29×10^2	8.80×10^2	8.90×10^2	7.56×10^2	7.01×10^2	5.01×10^2	4.34×10^2	5.20×10^2	7.650×10^2
F_20	8.51×10^2	5.26×10^2	8.96×10^2	8.89×10^2	7.46×10^2	6.41×10^2	4.93×10^2	4.19×10^2	4.77×10^2	8.100×10^2
F_21	9.14×10^2	4.42×10^2	8.16×10^2	8.52×10^2	4.85×10^2	5.01×10^2	5.24×10^2	5.42×10^2	5.14×10^2	4.0111×10^2
F_22	8.07×10^2	7.65×10^2	7.74×10^2	7.52×10^2	6.83×10^2	6.94×10^2	7.72×10^2	7.72×10^2	7.66×10^2	7.505×10^2
F_23	1.03×10^2	8.54×10^2	1.08×10^3	1.0×10^3	5.74×10^2	5.83×10^2	6.34×10^2	5.82×10^2	6.51×10^2	1.201×10^2
F_24	4.12×10^3	6.10×10^2	2.96×10^2	2.36×10^2	2.51×10^2	2.01×10^2	2.06×10^2	2.02×10^2	2.00×10^2	3.040×10^2
F_25	5.10×10^2	1.82×10^3	1.76×10^3	1.75×10^3	1.79×10^2	1.80×10^2	1.74×10^3	1.74×10^3	1.74×10^3	4.120×10^2

Table 6. Average error ranking of the 25 CEC'05 problems for all algorithms.

Problem	PSO [65]	IPOP-CMA-E [66]	CHC [67,68]	SSGA [69,70]	SS-BLX [71]	SS-Arit [72]	DE-Bin [73]	DE-Exp [73]	SaDE [74]	CS-CEOA
F_1	4	1	3	9	5	2	6	7	8	1
F_2	5	1	4	9	2	3	8	6	7	1
F_3	6	1	4	8	2	3	5	9	7	1
F_4	6	2	10	5	4	3	7	9	8	1
F_5	6	7	5	3	2	4	9	8	10	1
F_6	7	1	4	2	3	6	8	9	5	1
F_7	10	5	6	7	9	8	4	3	2	1
F_8	10	1	5	8	6	7	4	9	3	2
F_9	5	6	3	8	1	4	2	9	10	7
F_10	6	10	2	8	5	9	4	3	7	1
F_11	9	1	2	8	7	6	5	4	10	3
F_12	9	4	10	6	3	5	8	2	7	1
F_13	3	1	10	8	5	6	4	7	9	2
F_14	8	8	1	2	4	5	6	7	9	3
F_15	7	7	4	5	1	2	8	6	10	9
F_16	8	6	10	2	1	5	3	4	7	9
F_17	9	10	1	3	2	5	8	6	7	4
F_18	9	5	10	8	7	6	3	2	4	1
F_19	8	4	9	10	6	5	2	1	3	7
F_20	8	4	10	9	6	5	3	1	2	7
F_21	10	2	8	9	3	4	6	7	5	1
F_22	10	5	9	4	1	2	7	8	6	3
F_23	2	10	3	1	5	7	8	6	9	4
F_24	9	10	7	5	6	2	4	3	1	8
F_25	10	8	5	4	6	7	3	2	1	9

Table 7. Statistical frequency table of ranking values.

Rank / Method	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	R_10
PSO [65]	0	1	1	1	2	4	2	4	5	5
IPOP-CMA-ES [66]	7	2	0	3	3	2	2	2	0	4
CHC [67,68]	2	2	3	4	3	1	1	1	2	6
SSGA [69,70]	1	3	2	2	3	1	1	7	4	1
SS-BLX [71]	4	4	3	2	4	5	2	0	1	0
DE-Exp [73]	2	3	3	2	0	4	4	2	5	0
DE-Bin [73]	0	2	4	5	2	3	2	6	1	0
SS-Arit [72]	0	4	3	3	6	4	3	1	1	0
SaDE [74]	2	2	2	1	2	1	6	2	3	4
CS-CEOA	11	2	3	2	0	0	3	1	3	0

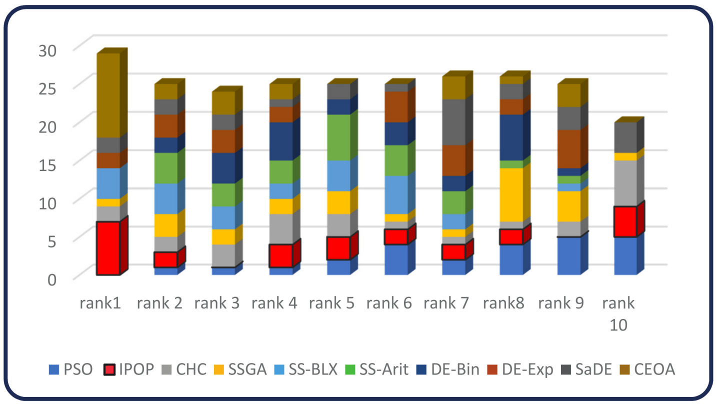

Figure 3. The relative weight of each algorithm by rank.

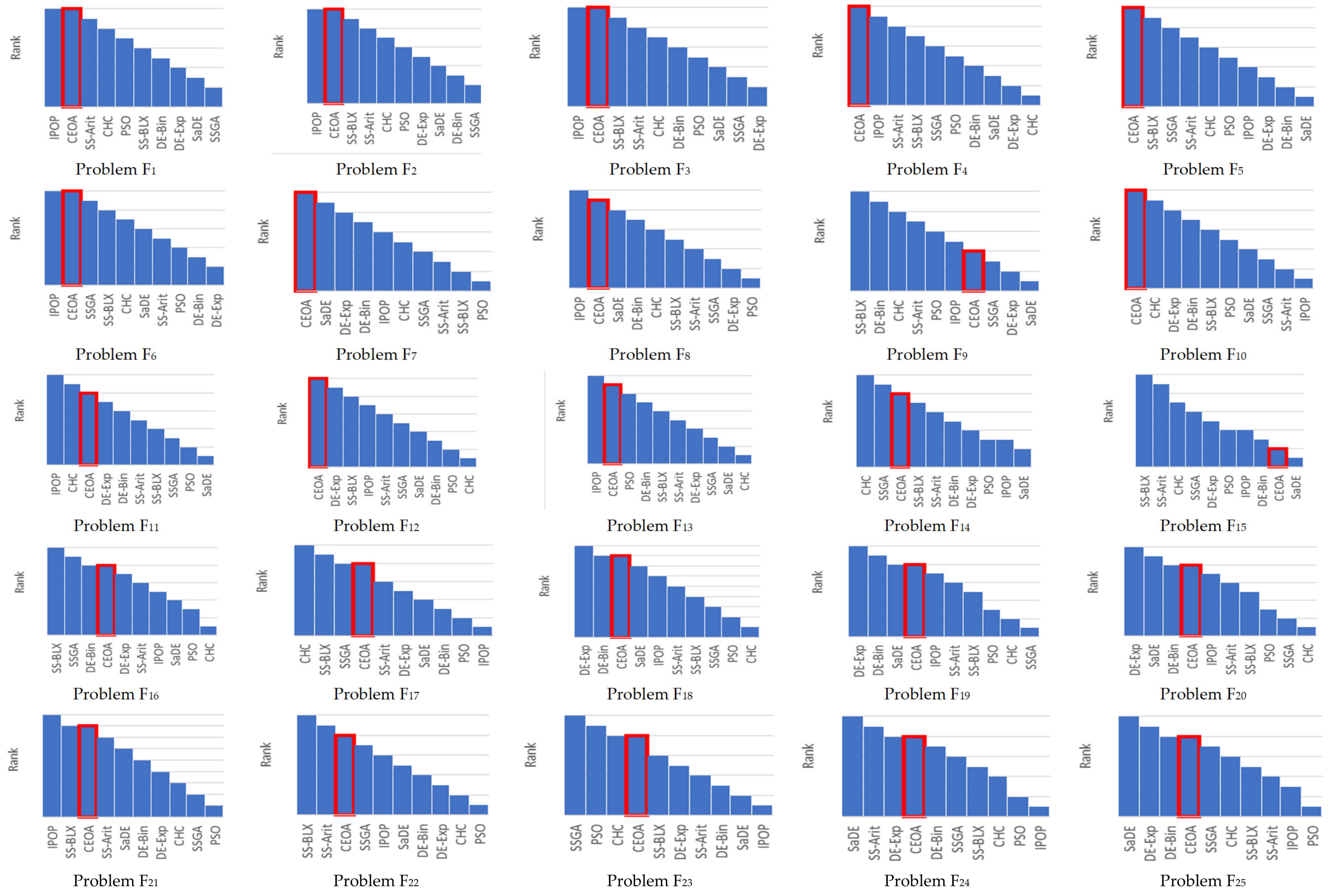

Figure 4. Comparison between the different algorithms according to its ranks with different problems.

4.4. Petrochemical Engineering Application (Blending Four Ingredients, Three feed Streams, One Pool, and Two Products)

Optimization has a lot of applications in various fields in chemical and petroleum engineering such as design, development, scheduling, analysis, planning, and operating chemical processes. It is helpful as it enables the formulation of unstable systems and utilizing sparsity and development process models. The pooling network system constructed of any number of feeder streams, pools (tanks), and products, in which any feeder stream may connect any tank and any product. These applications are familiar in chemical engineering and petrochemical engineering. Figure 5 illustrates a graphical structure of a simple pooling network system involving three feed streams, one bending tank, and two products.

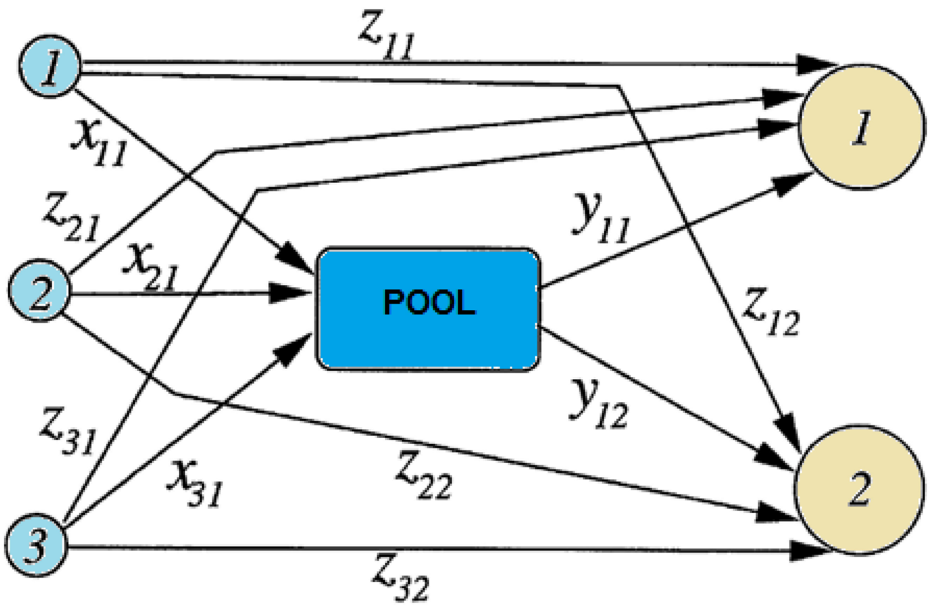

Figure 5. Structure representation of four ingredients polling system pooling problem.

The purpose of this application is to calculate the flow stream of these four ingredients in various pools in order to obtain certain specific products with certain (required) chemical properties and determining costs. The four ingredients can be blended in the pool or be directly reach any of the finite products. These applications were investigated, among others [75–79]. The vectors x_{ij}, y_{ij}, and z_{ij} represent the flow streams between different feeder i-pool l, pool l-product j, and feeder i-product j, respectively. Ben-Tal et al. [75] presented the substitution of flowrate x_{il} which represents the flow stream from feeder i to pool l; with a fractional flowrate q_{il}; representing the fraction of flow stream from feed i to pool l. With these notions, we can define the following sets.

I is the set of feed streams, J is the set of required products, L is the number of mixed tanks, and K is the set of components, whose quality is being monitored. For this petrochemical engineering application, we can define the parameters of the physical problem as follows:

A_i is the maximum output flow of feed i;
D_j the maximum predicted demand for product j;
S_l the size of Tank l: C_{ik} the percentage of ingredient k in feeder i,
P_{jk} the maximum percentage of ingredient k in product j, c_i is the unit price of feeder i, and d_j is the unit price of finite product j.

The mathematical formulation of a pooling system application is formulated as follows:

$$\begin{aligned}
&Max \sum_{j=1}^{J}\sum_{l=1}^{L}(d_j - \sum_{i\in I} c_i q_{il})y_{ij} + \sum_{i=1}^{I}\sum_{j=1}^{J}(d_j - c_i)z_{ij} \\
&Subject \quad to \\
&\qquad \sum_{l=1}^{L}\sum_{j=1}^{J} q_{il}y_{lj} + \sum_{j=1}^{J} z_{ij} \le A_i, \forall i \in I \\
&\qquad \sum_{j=1}^{J} y_{lj} \le S_l, \quad \forall l \in I \\
&\qquad \sum_{j=1}^{J} y_{lj} + \sum_{i=1}^{I} z_{ij} \le D_j, \quad \forall j \in J \\
&\qquad \sum_{l=1}^{L}\left(\sum_{i=1}^{I} C_{ik}q_{il} - P_{jk}\right)y_{lj} + \sum_{i=1}^{I}\left(C_{ik} - P_{jk}\right)z_{ij}, \quad \forall j \in J, \quad \forall k \in K, \\
&\qquad 0 \le q_{il} \le 1, \quad \forall i \in I, \quad \forall l \in L \\
&\qquad 0 \le y_{lj} \le D_j, \quad \forall l \in L, \quad \forall j \in J \\
&\qquad 0 \le z_{ij} \le D_j, \quad \forall i \in I, \quad \forall j \in J
\end{aligned} \tag{13}$$

Figure 6 shows the application network system with four feeder streams, one pool, and two products.

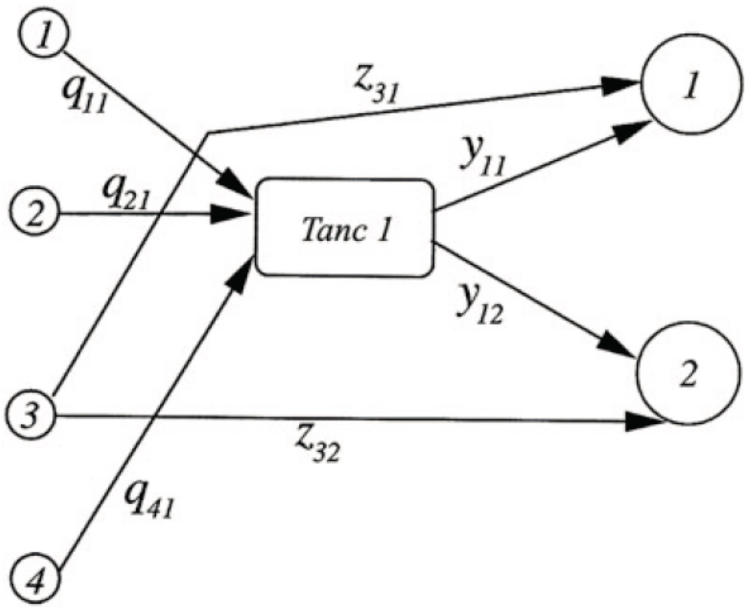

Figure 6. Structure representation of petrochemical pooling problem.

The data of this application [79] is as follows:

$$\begin{aligned}
&A = (\infty, \infty, \infty, 50), \\
&D = (100, 200), \\
&C = (3\ , 1\ , -1\ ), \\
&P = (5/2, 3/2), \\
&c = (6, 16, -, 15), \\
&d = (9, 15), \\
&S_1 = \infty.
\end{aligned}$$

The mathematical formulation of this network system is as follows:

$$
\begin{array}{l}
Max(9-6q_{11}-16q_{21}-15q_{41})y_{11}+(15-6q_{11}-16q_{21}-15q_{41})y_{12}-z_{31}+5z_{32} \\
Subject \quad to \\
\qquad q_{41}y_{11}+q_{41}y_{12}\leq 50, \\
\qquad y_{11}+z_{31}\leq 100, \\
\qquad y_{12}+z_{23}\leq 200, \\
\qquad (3q_{11}+q_{21}+q_{41}-2.5)y_{11}-0.5z_{31}\leq 0, \\
\qquad (3q_{11}+q_{21}+q_{41}-1.5)y_{12}-0.5z_{32}\leq 0, \\
\qquad q_{11}+q_{21}+q_{41}=1, \\
\qquad 0\leq y_{11}\leq 100, \quad 0\leq y_{12}\leq 200 \\
\qquad 0\leq z_{31}\leq 100, \quad 0\leq z_{32}\leq 200 \\
\qquad 0\leq q_{11}\leq 1, \quad 0\leq q_{21}\leq 1, \quad 0\leq q_{41}\leq 1
\end{array}
\tag{14}
$$

The results of this problem are presented in Table 8, which demonstrate the validity of the proposed algorithm to solve real-life applications.

Table 8. The results of this problem are presented in Figure 6.

Parameters	**Algorithms**				
	SNOPT [79]	**MINOS [79]**	**KNITRO [79]**	**CONOPT [79]**	**CS-CEOA**
q_{11}	1	1	1	1	1
q_{21}	0	0	1.8754×10^{-7}	0	0
q_{41}	0	0	6.6576×10^{-8}	0	0
y_{11}	50	50	50	50	50
y_{12}	50	50	50	50	50
z_{31}	50	50	50	50	50
z_{32}	150	150	150	150	150
Objective function	1300	1300	1299.9995	1300	1300

In this subsection, a comparative study has been investigated to examine the proposed algorithm concerning the solutions quality. First, evolutionary-based-approaches suffer from the quality of the solution, where they get an approximated optimal solution, and thus CS-CEOA has been used to evolve the quality of the obtained solution by applying a chaotic local search that guarantees fast convergence towards the true optimal solution. On the other hand, unlike conventional approaches, CS-CEOA searches using a population of particles, not a single point, so CS-CEOA can provide a globally search algorithm, that can locate the global zone from the search space. In addition, CS-CEOA implements only the objective function values, not derivatives, or any other auxiliary knowledge; therefore, it can handle non-continuous, non-smooth, and non-differentiable functions which are presented in practical real-life optimization problems. Furthermore. In addition, the equilibrium optimizer can be hybridized with other search processes, and its parameters can be modified to improve the efficiency of CS-CEOA. The findings of the simulation also show the superiority of CS-CEOA over those stated in the literature, as it is substantially better than other methods. Finally, owing to the simplicity of the procedures, the reality of using CS-CEOA to deal with complex problems of realistic dimensions has been approved.

5. Limitations of the Proposed Algorithm

The core advantage of the traditional optimization techniques is that it guarantees to find the truly global solution, unlike population-based approaches, but they have critical limitations with large-scale real-life, nondifferentiable, nonconvex, ill-defined problems, and non-formulated problems. Population-based methods are usually very efficient and robust in finding near-global solutions, especially with complex problems. There are some critical limitations of the proposed technique. The proposed technique randomly generates the position for agents, which produce degeneracy. The degeneracy occurs when multiple agents represent the same position, which may lead to an inefficient solution.

To date, mathematical theoretical convergence analysis of population-based algorithms is still at an early stage and has been experimentally studied in the literature, making mathematical convergence analysis an important subject in a future study. The advantages and disadvantages of the proposed algorithm can be stated in Table 9.

Table 9. Advantages and Disadvantages.

Advantages	Have greater success at locating global solution for small problems and nearby global optimal solution in the very large complex problems. Do not require nether, convexity, continuous, differentiability, formulated, nor well-defined problem Can be applied with both, discrete, continuous and mixed variables.
Disadvantages	Time consuming algorithm, especially for very large scale and complex problems. Mathematical theoretical analysis for convergence to the global solution is still at an early stage, and need further investigation Produce a nearby Global solution in the very large scale complex problems.

6. Conclusions

Recently developed nature-inspired optimization approaches are good techniques for finding global solutions for real-life optimization applications. The equilibrium optimizer algorithm (EOA) is a novel optimization approach, which inspired by control volume mass balance models implemented to determine both dynamic and equilibrium states. In this paper, the chaotic search-based constrained equilibrium optimizer algorithm (CS-CEOA) has been proposed as a new algorithm for optimizing constrained optimization problems. CS-CEOA integrates the algorithm of evolving individuals modeled by EOA with the algorithm of Local-improvement of chaotic local search (CLS); thus, CS-CEOA synthesizes the merits of both EOA and chaotic search, and it is a simple and yet robust model to deal with different types of optimization problems. CS-CEOA is computed in two phases, the first one (phase I) intends to locate the approximate optimal solution, avoiding being trapped in local minima, while in phase II, the chaos-based search algorithm increases local search performance and obtain the best optimal solution. In addition, a repair function was implemented to co-evolves any infeasible solution until it becomes feasible, in a way such that, a new feasible solution is created on the segment defined by the feasible reference point and the infeasible solution itself. Due to the fast globally converging characteristics of evolutionary algorithms, and the chaotic search's exhaustive search, CS-CEOA was able to locate the true optimal solution by implementing an exhaustive local search on a small zone. The superior performance of CS-CEOA in comparison to the performance of the recent competitive algorithms has been validated by multi-benchmark suites of problems including constrained, unconstrained, CEC'05 problems, and an application of blending four ingredients, three feed streams, one tank, and two products to obtain some certain products with specific chemical properties and determining costs. The results were compared with the standard evolutionary algorithm, which concludes the superiority of CS-CEOA to handle non-linear programming problems. The following observations reveal some major benefits of the proposed approach:

1. CS-CEOA has been used to increase the solution quality by combining the merits of EOA and CLS.
2. Implementing chaotic local search influences the algorithm convergence time, saving up to 12% of the time.
3. Unlike traditional techniques, CS-CEOA searches using a population of particles, therefore it can be considered as a global search algorithm.
4. CS-CEOA uses only the objective function values, therefore it can handle all types of functions that existed in practical real-life optimization problems.
5. The numerical simulation approves the superiority of CS-CEOA to the reported algorithms in the literature.

To date, theoretical convergence analysis of evolutionary algorithms is still at an early stage and has been experimentally studied in the literature, making mathematical convergence analysis an important subject in a future study.

Author Contributions: Conceptualization, writing—original draft preparation, methodology, formal analysis, investigation, A.A.A.M., M.A.E.-S., I.M., and H.A. writing—review and editing, A.A.A.M., M.A.E.-S., and I.M. Supervision, A.A.A.M. and M.A.E.-S. 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.

Acknowledgments: The authors extend their appreciation the great support of Deanship of Scientific Research, in Taif University for funding Taif University Researcher Supporting Project Number (Tursp-2020/48), Taif University, Taif, Saudi Arabia.

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

References

1. Luenberger, D.G. *Linear and Non-Linear Programming*, 2nd ed.; Addison-Wesley: Boston, MA, USA, 1984.
2. Boyd, S.; Vandenberghe, L. *Convex Optimization*; Cambridge University Press: Cambridge, UK, 2004.
3. Afshar, M.H.; Faramarzi, A. Size Optimization of Truss Structures by Cellular Automata. *J. Comput. Sci. Eng.* **2010**, *3*, 1–9.
4. Faramarzi, A.; Afshar, M.H. A novel hybrid cellular automata–linear programming approach for the optimal sizing of planar truss structures. *Civ. Eng. Environ. Syst.* **2014**, *31*, 209–228. [CrossRef]
5. Faramarzi, A.; Afshar, M.H. Application of cellular automata to size and topology optimization of truss structures. *Sci. Iran.* **2012**, *19*, 373–380. [CrossRef]
6. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
7. El-Shorbagy, M.A.; Mousa, A.A.; Farag, M. Solving non-linear single-unit commitment problem by genetic algorithm based clustering technique. *Rev. Comput. Eng. Res.* **2017**, *4*, 11–29. [CrossRef]
8. El-Shorbagy, M.A.; Mousa, A.A.; Farag, M.A. An intelligent computing technique based on a dynamic-size subpopulations for unit commitment problem. *OPSEARCH* **2019**, *56*, 911–944. [CrossRef]
9. El-Shorbagy, M.A.; Ayoub, A.Y.; Mousa, A.A.; El-Desoky, I.M. An Enhanced Genetic Algorithm with New Mutation for Cluster Analysis. *Comput. Stat.* **2019**, *34*, 1355–1392. [CrossRef]
10. El-Desoky, I.M.; El-Shorbagy, M.A.; Nasr, S.M.; Hendawy, Z.M.; Mousa, A.A. A Hybrid Genetic Algorithm for Job Shop Scheduling Problems. *Int. J. Adv. Eng. Technol. Comput. Sci.* **2016**, *3*, 6–17.
11. Saurabh, P.; Verma, B. An efficient proactive artificial immune system based anomaly detection and prevention system. *Expert Syst. Appl.* **2016**, *60*, 311–320. [CrossRef]
12. Mousa, A.A.; El-Shorbagy, M.A. Identifying a Satisfactory Operation Point for Fuzzy Multiobjective Environmental/Economic Dispatch Problem. *Am. J. Math. Comput. Model.* **2016**, *1*, 1–14.
13. Mousa, A.A.; El-Shorbagy, M.A. Enhanced particle swarm optimization based local search for reactive power compensation problem. *Appl. Math.* **2012**, *3*, 1276–1284. [CrossRef]
14. 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.
15. El-Wahed, W.F.A.; Mousa, A.A.; El-Shorbagy, M.A. Integrating particle swarm optimization with genetic algorithms for solving non-linear optimization problems. *J. Comput. Appl. Math.* **2011**, *235*, 1446–1453. [CrossRef]
16. Mousa, A.A.; El_Desoky, I.M. Stability of Pareto optimal allocation of land reclamation by multistage decision-based multi-pheromone ant colony optimization. *Swarm Evol. Comput.* **2013**, *13*, 13–21. [CrossRef]
17. Karaboga, D. *An Idea Based on Honey Bee Swarm for Numerical Optimization*; Technical Report-TR06; Erciyes University, Engineering Faculty, Computer Engineering Department: Kayseri, Turkey, 2005.
18. Passino, K.M. Biomimicry of Bacteria Foraging for Distributed Optimization and Control. *IEEE Control Syst. Mag.* **2002**, *22*, 52–67.
19. Zhao, W.; Wang, L. An effective bacterial foraging optimizer for global optimization. *Inf. Sci.* **2016**, *329*, 719–735. [CrossRef]
20. Guo, L.; Meng, Z.; Sun, Y.; Wang, L. Parameter identification and sensitivity analysis of solar cell models with cat swarm optimization algorithm. *Energy Convers Manag.* **2016**, *108*, 520–528. [CrossRef]
21. Marinaki, M.; Marinakis, Y. A Glowworm Swarm Optimization algorithm for the Vehicle Routing Problem with Stochastic Demands. *Expert Syst. Appl.* **2016**, *46*, 145–163. [CrossRef]
22. Verma, S.; Mukherjee, V. Firefly algorithm for congestion management in deregulated environment. *Eng. Sci. Technol. Int. J.* **2016**, *19*, 1254–1265. [CrossRef]

23. Zhou, Y.; Chen, X.; Zhou, G. An improved monkey algorithm for a 0–1 knapsack problem. *Appl. Soft Comput.* **2016**, *38*, 817–830. [CrossRef]
24. Bolaji, A.L.; 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. [CrossRef]
25. Shehab, M.; Khader, A.T.; Laouchedi, M.; Alomari, O.A. Hybridizing cuckoo search algorithm with bat algorithm for global numerical optimization. *J. Supercomput.* **2019**, *75*, 2395–2422. [CrossRef]
26. Mirjalili, S.; Lewis, A. The Whale Optimization Algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
27. Farag, M.A.; El-Shorbagy, M.A.; Mousa, A.A.; El-Desoky, I.M. A New Hybrid Metaheuristic Algorithm for Multiobjective Optimization Problems. *Int. J. Comput. Intell. Syst.* **2020**, *13*, 920–940. [CrossRef]
28. El-Shorbagy, M.A.; Farag, M.A.; Mousa, A.A.; El-Desoky, I.M. A Hybridization of Sine Cosine Algorithm with Steady State Genetic Algorithm for Engineering Design Problems. In Proceedings of the International Conference on Advanced Machine Learning Technologies and Applications AMLTA 2019, AISC 921, Cairo, Egypt, 28–30 March 2019; pp. 1–13.
29. Saremi, S.; Mirjalili, S.; Lewis, A. Grasshopper Optimisation Algorithm: Theory and application. *Adv. Eng. Softw.* **2017**, *105*, 30–47. [CrossRef]
30. Mirjalili, S.; Gandomi, A.H.; Mirjalili, S.Z.; Saremi, S.; Faris, H.; Mirjalili, S.M. Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. *Adv. Eng. Softw.* **2017**, *114*, 163–191. [CrossRef]
31. Faramarzi, A.; Heidarinejad, M.; Stephens, B.; Mirjalili, S. Equilibrium optimizer: A novel optimization algorithm. *Knowl.-Based Syst.* **2020**, *191*, 105190. [CrossRef]
32. Ahmadianfar, I.; Bozorg-Haddad, O.; Chu, X. Gradient-based optimizer: A new Metaheuristic optimization algorithm. *Inf. Sci.* **2020**, *540*, 131–159. [CrossRef]
33. Li, S.; Chen, H.; Wang, M.; Heidari, A.A.; Mirjalili, S. Slime mould algorithm: A new method for stochastic optimization. *Future Gener. Comput. Syst.* **2020**, *111*, 300–323. [CrossRef]
34. Heidari, A.A.; Mirjalili, S.; Faris, H.; Aljarah, I.; Mafarja, M.; Chen, H. Harris hawks optimization: Algorithm and applications. *Future Gener. Comput. Syst.* **2019**, *97*, 849–872. [CrossRef]
35. Meng, O.K.; Pauline, O.; Kiong, S.C. A carnivorous plant algorithm for solving global optimization problems. *Appl. Soft Comput. J.* **2020**, *98*, 106833. [CrossRef]
36. Gupta, S.; Deep, K.; Mirjalili, S.; Kim, J.H. A modified Sine Cosine Algorithm with novel transition parameter and mutation operator for global optimization. *Expert Syst. Appl.* **2020**, *154*, 113395. [CrossRef]
37. Ghasemi, M.; Davoudkhani, I.F.; Akbari, E.; Rahimnejad, A.; Ghavidel, S.; Li, L. A novel and effective optimization algorithm for global optimization and its engineering applications: Turbulent Flow of Water-based Optimization (TFWO). *Eng. Appl. Artif. Intell.* **2020**, *92*, 103666. [CrossRef]
38. Chen, H.; Li, W.; Yang, X. A whale optimization algorithm with chaos mechanism based on quasi-opposition for global optimization problems. *Expert Syst. Appl.* **2020**, *158*, 113612. [CrossRef]
39. Chu, X.; Cai, F.; Gao, D.; Li, L.; Cui, J.; Xu, S.X.; Qin, Q. An artificial bee colony algorithm with adaptive heterogeneous competition for global optimization problems. *Appl. Soft Comput. J.* **2020**, *93*, 106391. [CrossRef]
40. Wu, J.; Wang, Y.-G.; Burrage, K.; Tian, Y.-C.; Lawson, B.; Ding, Z. An improved firefly algorithm for global continuous optimization problems. *Expert Syst. Appl.* **2020**, *149*, 113340. [CrossRef]
41. Li, N.; Wang, L. Bare-Bones Based Sine Cosine Algorithm for global optimization. *J. Comput. Sci.* **2020**, *47*, 101219. [CrossRef]
42. Zhang, Y.; Jin, Z. Group teaching optimization algorithm: A novel metaheuristic method for solving global optimization problems. *Expert Syst. Appl.* **2020**, *148*, 113246. [CrossRef]
43. Askari, Q.; Younas, I.; Saeed, M. Political Optimizer: A novel socio-inspired meta-heuristic for global optimization. *Knowl.-Based Syst.* **2020**, *195*, 105709. [CrossRef]
44. Yimit, A.; Iigura, K.; Hagihara, Y. Refined selfish herd optimizer for global optimization problems. *Expert Syst. Appl.* **2020**, *139*, 112838. [CrossRef]
45. Wei, X.; Yuan, S.; Ye, Y. Optimizing facility layout planning for reconfigurable manufacturing system based on chaos genetic algorithm. *Prod. Manuf. Res.* **2019**, *7*, 109–124. [CrossRef]
46. Dash, S.; Thulasiram, R.; Thulasiraman, P. Modified Firefly Algorithm with Chaos Theory for Feature Selection: A Predictive Model for Medical Data. *Int. J. Swarm Intell. Res.* **2019**, *10*, 1–20. [CrossRef]
47. Fuertes, G.; Vargas, M.; Alfaro, M.; Soto-Garrido, R.; Sabattin, J.; Peralta, M.A. Chaotic genetic algorithm and the effects of entropy in performance optimization. *Chaos* **2019**, *29*, 013132. [CrossRef] [PubMed]
48. Mousa, A.A.; Qassm, S.M.; Alnefaie, A.A. Sinusoidal Chaotic Genetic Algorithm for Constrained Optimization: Recent Trends in Applied Optimization. *Int. J. Eng. Res. Technol.* **2019**, *12*, 2787–2802.
49. L-Shorbagy, M.A.E.; Mousa, A.A.; Nasr, S.M. A chaos-based evolutionary algorithm for general non-linear programming problem. *Chaos Solitons Fractals* **2016**, *85*, 8–21. [CrossRef]
50. Abo-Elnaga, Y.; Nasr, S.M.; El-Desoky, I.M.; Hendawy, Z.M.; Mousa, A.A. Enhanced Genetic Algorithm and Chaos Search for Bilevel Programming Problems. In Proceedings of the International Conference on Advanced Machine Learning Technologies and Applications, Cairo, Egypt, 28–30 March 2019; pp. 478–487.
51. Abdel-Basset, M.; Mohamed, R.; Mirjalili, S. A Binary Equilibrium Optimization Algorithm for 0–1 Knapsack Problems. *Comput. Ind. Eng.* **2020**. [CrossRef]

52. Gupta, S.; Deep, K.; Mirjalili, S. An efficient equilibrium optimizer with mutation strategy for numerical optimization. *Appl. Soft Comput. J.* **2020**, *96*, 106542. [CrossRef]
53. Rabehi, A.; Nail, B.; Helal, H.; Douara, A.; Ziane, A.; Amrani, M.; Akkal, B.; Benamara, Z. Optimal estimation of Schottky diode parameters using a novel optimization algorithm: Equilibrium optimizer. *Superlattices Microstruct.* **2020**, *146*, 106665. [CrossRef]
54. Abdel-Basset, M.; Mohamed, R.; Mirjalili, S.; Chakrabortty, R.K.; Ryan, M.J. Solar photovoltaic parameter estimation using an improved equilibrium optimizer. *Sol. Energy* **2020**, *209*, 694–708. [CrossRef]
55. Shaheen, A.M.; Elsayed, A.M.; El-Sehiemy, R.A.; Abdelaziz, A.Y. Equilibrium optimization algorithm for network reconfiguration and distributed generation allocation in power systems. *Appl. Soft Comput. J.* **2020**, *98*, 106867. [CrossRef]
56. Suganthan, P.N.; Hansen, N.; Liang, J.J.; Deb, K.; Chen, Y.P.; Auger, A.; Tiwari, S. *Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization*; Kanpur Genetic Algorithms Laboratory (KanGAL) Report 2005005, No. 2005; Indian Institute of Technology Kanpur (IIT Kanpur): Kanpur, India, 2005; Available online: https://bee22.com/resources/Liang%20CEC2014.pdf (accessed on 20 October 2020).
57. Sedlaczek, K.; Eberhard, P. Constrained Particle Swarm Optimization of Mechanical Systems. In Proceedings of the 6th World Congresses of Structural and Multidisciplinary Optimization, Rio de Janeiro, Brazil, 30 May–3 June 2005; Available online: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.60.4525&rep=rep1&type=pdf (accessed on 20 October 2020).
58. Mousa, A.A.; El-Shorbagy, M.A.; Farag, M.A. Steady-State Sine Cosine Genetic Algorithm Based Chaotic Search for non-linear Programming and Engineering Applications. *IEEE Access* **2020**, *8*, 212036–212054. [CrossRef]
59. Osman, M.S.; Abo-Sinna, M.A.; Mousa, A.A. A Combined Genetic Algorithm-Fuzzy Logic Controller (GA-FLC) In non-linear Programming. *J. Appl. Math. Comput.* **2005**, *170*, 821–840. [CrossRef]
60. Osman, M.S.; Abo-Sinna, M.A.; Mousa, A.A. IT-CEMOP: An Iterative Co-evolutionary Algorithm for Multiobjective Optimization Problem with non-linear Constraints. *J. Appl. Math. Comput.* **2006**, *183*, 373–389. [CrossRef]
61. El-Shorbagy, M.A.; Mousa, A.A. A Hybrid Optimization System Coupling Particle Swarm Optimization Algorithm and Genetic Algorithm Applied to non-linear Optimization Problems. *Online J. Math. Stat.* **2015**, *6*, 118–125.
62. Chelouah, R.; Siarry, P. A continuous genetic algorithm designed for the global optimization of multimodal functions. *J. Heurist.* **2000**, *6*, 191–213. [CrossRef]
63. Chelouah, R.; Siarry, P. Genetic and Nelder–Mead algorithms hybridized for a more accurate global optimization of continuous multiminima functions. *Eur. J. Oper. Res.* **2003**, *148*, 335–348. [CrossRef]
64. Kao, Y.; Zahara, E. A hybrid genetic algorithm and particle swarm optimization for multimodal functions. *Appl. Soft Comput.* **2008**, *8*, 849–857. [CrossRef]
65. Kennedy, J.; Eberhart, R. Particle swarm optimization. In *Proceedings of IV IEEE International Conference on Neural Networks*; IEEE: Piscataway, NJ, USA, 1995; pp. 1942–1948.
66. Auger, A.; Hansen, N. A restart CMA evolution strategy with increasing population size. In Proceedings of the 2005 IEEE Congress on Evolutionary Computation, Edinburgh, Scotland, UK, 2–5 September 2005; pp. 1769–1776.
67. Eshelman, L.J. The CHC adaptative search algorithm: How to have safe search when engaging in nontraditional genetic recombination. In *Foundations of Genetic Algorithms*; Rawlins, G.J.E., Ed.; Morgan Kaufmann: San Mateo, CA, USA, 1991; pp. 265–283.
68. Eshelman, L.J.; Schaffer, J.D. Real-coded genetic algorithms and interval schemata. In *Foundations of Genetic Algorithms*; Whitley, D., Ed.; Morgan Kaufmann: San Mateo, CA, USA, 1993; pp. 187–202.
69. Fernandes, C.; Rosa, A. A study of non-random matching and varying population size in genetic algorithm using a royal road function. In Proceedings of the 2001 Congress on Evolutionary Computation, Seoul, Korea, 27–30 May 2001; pp. 60–66.
70. Mülenbein, H.; Schlierkamp-Voosen, D. Predictive models for the breeding genetic algorithm in continuous parameter optimization. *Evol. Comput.* **1993**, *1*, 25–49. [CrossRef]
71. Herrera, F.; Lozano, M.; Molina, D. Continuous scatter search: An analysis of the integration of some combination methods and improvement strategies. *Eur. J. Oper. Res.* **2006**, *169*, 450–476. [CrossRef]
72. Laguna, M.; Marti, R. *Scatter Search. Methodology and Implementation in C*; Kluwer Academic Publishers: Dordrecht, The Netherlands, 2003.
73. Price, K.V.; Rainer, M.; Lampinen, J.A. *Differential Evolution: A Practical Approach to Global Optimization*; Springer: Berlin/Heidelberg, Germany, 2005.
74. Qin, A.K.; Suganthan, P.N. Self-adaptive differential evolution algorithm for numerical optimization. In Proceedings of the 2005 IEEE Congress on Evolutionary Computation, Scotland, UK, 2–5 September 2005; Volume 2, pp. 1785–1791.
75. Ben-Tal, A.; Eiger, G.; Gershovitz, V. Global minimization by reducting the duality gap. *Math. Program.* **1994**, *63*, 193–212. [CrossRef]
76. Haverly, C. Studies of the behavior of recursion for the pooling problem. *ACM-Sigmap Bull.* **1978**, *25*, 19–28. [CrossRef]
77. Lasdon, L.; Waren, A.; Sarkar, S.; Palacios, F. Solving the pooling problem using generalized reduced gradient and successive linear programming algorithms. *ACM-Sigmap Bull.* **1979**, *26*, 9–15. [CrossRef]

78. Visweswaran, V.; Floudas, C.A. Computational results for an efficient implementation of the GOP algorithm and its variants. In *Gloobal Optimization in Engineering Design*; Grassmann, I.E., Ed.; Kluwer Book Series in Nonconvex Optimization and Its Applications; Kluwer Academic: Dordrecht, The Netherlands, 1996; Chapter 4.
79. Andrei, N. *Non-Linear Optimization Applications Using the GAMS Technology*; Springer: New York, NY, USA; Berlin/Heidelberg, Germany; NDordrecht, The Netherlands; NLondon, UK, 2013; Available online: https://link.springer.com/book/10.1007/978-1-4614-6797-7 (accessed on 20 October 2020).