mathematics

MDPI

Article

Guided Hybrid Modified Simulated Annealing Algorithm for Solving Constrained Global Optimization Problems

Khalid Abdulaziz Alnowibet [1], Salem Mahdi [2], Mahmoud El-Alem [3], Mohamed Abdelawwad [4] and Ali Wagdy Mohamed [5,6,*]

[1] Statistics and Operations Research Department, College of Science, King Saud University, P.O. Box 2455, Riyadh 11451, Saudi Arabia; knowibet@ksu.edu.sa
[2] Educational Research and Development Center Sanaa, Sanaa 31220, Yemen; samath2014@yahoo.com
[3] Department of Mathematics & Computer Science, Faculty of Science, Alexandria University, Alexandria 21544, Egypt; mmelalem@alexu.edu.eg
[4] Institute for Computer Architecture and System Programming, University of Kassel, 34127 Kassel, Germany; m.abdelawwad@uni-kassel.de
[5] Operations Research Department, Faculty of Graduate Studies for Statistical Research, Cairo University, Giza 12613, Egypt
[6] Department of Mathematics and Actuarial Science, School of Sciences Engineering, The American University in Cairo, Cairo 11835, Egypt
* Correspondence: aliwagdy@gmail.com

Abstract: In this paper, a hybrid gradient simulated annealing algorithm is guided to solve the constrained optimization problem. In trying to solve constrained optimization problems using deterministic, stochastic optimization methods or hybridization between them, penalty function methods are the most popular approach due to their simplicity and ease of implementation. There are many approaches to handling the existence of the constraints in the constrained problem. The simulated-annealing algorithm (SA) is one of the most successful meta-heuristic strategies. On the other hand, the gradient method is the most inexpensive method among the deterministic methods. In previous literature, the hybrid gradient simulated annealing algorithm (GLMSA) has demonstrated efficiency and effectiveness to solve unconstrained optimization problems. In this paper, therefore, the GLMSA algorithm is generalized to solve the constrained optimization problems. Hence, a new approach penalty function is proposed to handle the existence of the constraints. The proposed approach penalty function is used to guide the hybrid gradient simulated annealing algorithm (GLMSA) to obtain a new algorithm (GHMSA) that finds the constrained optimization problem. The performance of the proposed algorithm is tested on several benchmark optimization test problems and some well-known engineering design problems with varying dimensions. Comprehensive comparisons against other methods in the literature are also presented. The results indicate that the proposed method is promising and competitive. The comparison results between the GHMSA and the other four state-Meta-heuristic algorithms indicate that the proposed GHMSA algorithm is competitive with, and in some cases superior to, other existing algorithms in terms of the quality, efficiency, convergence rate, and robustness of the final result.

Keywords: nonlinear function; constrained optimization; hybrid algorithm; global optima; line search; gradient method; meta-heuristics; simulated annealing algorithm; constraint handling; penalty function; evolutionary computation; numerical comparisons

MSC: 65D05

Citation: Alnowibet, K.A.; Mahdi, S.; El-Alem, M.; Abdelawwad, M.; Mohamed, A.W. Guided Hybrid Modified Simulated Annealing Algorithm for Solving Constrained Global Optimization Problems. *Mathematics* **2022**, *10*, 1312. https://doi.org/10.3390/math10081312

Academic Editor: Savin Treanta

Received: 3 March 2022
Accepted: 11 April 2022
Published: 14 April 2022

1. Introduction

Optimization problems arise in different applications fields, such as technical sciences, industrial engineering, economics, networks, chemical engineering, etc. See for example [1–5]

In general, the constrained optimization problem can be formulated as follows:

$$\begin{array}{ll} \min\limits_{x\in\mathbb{R}^n} & f(x), \\ s.t & g_l(x) \leq 0, \quad l = 1,2,\ldots,q, \\ & h_d(x) = 0, \quad d = 1,2,\ldots,m, \;\; m < n \\ & a_i \leq x_i \leq b_i, \; i = 1,2,\ldots,n, \end{array} \tag{1}$$

where $a_i \in \{\mathbb{R} \cup \{-\infty\}\}$, and $b_i \in \{\mathbb{R} \cup \{\infty\}\}$.

The functions $f(x)$, $g_l(x)$, $h_j(x) : \mathbb{R}^n \to \mathbb{R}$ are real valued functions, n denotes the number of variables in x, q is the number of inequality constraints, m is the number of equality constraints, a is a lower bounded on x and b is an upper bounded on x. The objective function f, the inequality constraints g_l, $l = 1,2,\ldots,q$, and the equality constraint h_d, $d = 1,2,\ldots,m$, are assumed to be continuously differentiable nonlinear functions.

Recently, there has been great development of optimization algorithms that are proposed to find global solutions to optimization problems. See for example [2,6–8].

The global optimization methods are used to prevent convergence to local optima and increase the probability of finding the global optimum [9].

The numerical global optimization algorithms can be classified into two classes: deterministic and stochastic methods. In stochastic methods, the minimization process depends partly on probability. In deterministic methods, in contrast, no probabilistic information is used [9].

So, for finding the global minimum of the unconstrained problem by using deterministic methods, it needs an exhaustive search over the feasible region of the function f and additional assumptions for the function f. On the contrary, to find the global minimum of the unconstrained problems, by using stochastic methods, one can prove the asymptotic convergence in probability, i.e., these methods are asymptotically successful with probability 1, see for example [10–12]. In general, the computational results of the stochastic methods are better than those of the deterministic methods [13].

Due to those reasons, a meta-heuristics strategy (stochastic method) is used to guide the search process [13]. Hence a meta-heuristic is a technique designed for solving a problem more quickly when classic methods are too slow, or for finding an approximate solution when classic methods fail to find any exact or near-exact solution. This is achieved by trading optimality, completeness, accuracy, or precision for speed [14–16].

The simulated-annealing algorithm (SA) is one of the most successful meta-heuristic strategies. In fact, the numerical results display that the simulated annealing technique is very efficient and effective for finding the global minimizer. See, for example, [2,5,17–19].

On the other hand, the gradient method is the most inexpensive method for finding a local minimizer of a continuously differentiable function. It has been proved that the gradient algorithm converges locally to a local minimizer [20]. Therefore, if a line-search (L) is added to the gradient method (G) as a globalization strategy, the resulting algorithm is globally convergent to a local minimizer (GL) [9,21,22].

Hence, when the simulated-annealing algorithm (SA) as a global optimization algorithm is combined with the line-search gradient method (GL) as a globally convergent method, the result is the hybrid gradient simulated annealing algorithm (GLMSA) [23]. The idea behind this hybridization is to gain the benefits and advantages of both the GL algorithm and the MSA algorithm.

As a matter of fact, the numerical results demonstrated that the (GLMSA) algorithm is a very efficient, effective and strong competitor for finding the global minimizer. For example, Table 4 of [23] shows that the GL algorithm is able to reach the optimum point of all test problems whose objective functions have only one minimum point (no local minima except the global one. i.e., convex function) and it is stuck at a local minimum for test problems whose objective functions have several local minima (with one global minimum, i.e., non-convex function). Table 6 of [23] demonstrates that the SMA modified simulated annealing algorithm finds the global minimum of all test problems from any starting point

of the feasible search space S. However, the GLMSA hybrid gradient simulated annealing algorithm is faster than MSA; also, GLMSA is efficient and effective compared to other meta-heuristic algorithms.

All the above have motivated and encouraged us to generalize the GLMSA algorithm to solve Problem (1).

The literature review analysis shows that the handling constraint which is based on a penalty function is considered the most popular implemented mechanism; this is due to its simplicity and ease of implementation [24–27]. A penalty technique transforms Problem (1) into an unconstrained problem by adding the penalty term of each constraint violation to the objective function value. The remainder of this paper is organized as follows. The next section provides a brief description of the GLMSA algorithm. Constraint handling, the penalty function method, proposed penalty method and interior-point algorithm are presented in Section 3. A guided hybrid simulated annealing algorithm to solve constrained problems is presented in Section 4. Numerical results are given in Section 5. Section 6 contains some concluding remarks.

Note: Section Abbreviations provides a list of the abbreviations and symbols which are used in this paper.

2. Summarized Description of GLMSA Algorithm

The GLMSA algorithm has been designed for solving unconstrained optimization problems; in this paper the GLMSA algorithm is generalized to solve Problem (1). The GLMSA algorithm contains two approaches to find a new step at each iteration, the first one is the gradient method. In this approach, a candidate point is generated and it might be accepted or rejected. If the objective function f is decreased at this point, then it will be accepted, otherwise, the second approach will be used to generate another point.

2.1. The First Approach (Gradient Method)

The gradient method solves an unconstrained optimization problem iteratively, such that at each iteration, a step in the direction of the negative gradient is computed and added to the current point as follows. Given an initial guess $x_0 \in \mathbb{R}^n$, the gradient method generates a sequence $\{x_k\}, k \geq 0$ of the objective function of the unconstrained optimization problem such that:

$$x_{k+1} = x_k + d_k, \tag{2}$$

where d_k is the first step, and it is defined by:

$$d_k = -|\alpha_k| g(x_k), \tag{3}$$

where $g(x_k)$ the gradient vector of the function f at point x_k and α_k is a step length along the negative gradient direction $(-g(x_k))$. The step length α_k along the $-g(x_k)$ is defined by:

$$\alpha_k = \frac{f(x_k)}{\| g(x_k) \|_2^2}. \tag{4}$$

The G gradient algorithm is listed in Algorithm 1 of [23]. The step length λ_k that is computed by the backtracking line-search approach is very important for global convergence of the gradient method. The following section presents a brief description of the backtracking line-search approach for globalizing the gradient method.

Globalizing the First Approach (Gradient Method)

To make the gradient method capable of finding a local minimizer x^* of the objective function of the unconstrained optimization problem from any starting point x_0, the G algorithm (gradient algorithm) is combined with the L algorithm (line-search algorithm) in order to obtain globally convergent algorithm GL. This algorithm is listed in Algorithm 1

below and it contains the first approach (gradient algorithm G) and the backtracking line-search algorithm L.

Algorithm 1 Line-Search Gradient Algorithm "GL"

Input: $f : \mathbb{R}^n \to \mathbb{R}$, $f \in C^1$, $\gamma \in (0,1)$, $k = 0$, a starting point $x_k \in \mathbb{R}^n$ and $\varepsilon > 0$.
Output: $x^* = x_{ac}$ the local minimizer of f, $f(x^*)$, the value of f at x^*

Set $x_{ac} = x_0$. ▷ x_{ac} is accepted solution.
Compute $f_{ac} = f(x_{ac})$, $g_{ac} = g(x_{ac})$ and d_k.
while $\|g_{ac}\|_2 > \varepsilon$ **do** ▷ g_{ac} is the value of the gradient vector at the accepted point x_{ac}.
    Set $k = k + 1$.
    $x_k = x_{ac} + d_k$ ▷ x_{ac} is the accepted point form the previous iteration.
    Compute $f_k = f(x_k)$
    Set $\lambda = 1$.
    while $f_k > f_{ac} + \gamma\lambda g_{ac}^T d_k$ **do**
        Set $\lambda = \frac{\lambda}{2}$
        $x_k = x_{ac} - \lambda g_{ac}$ ▷ in this paper the value of γ is 10^{-4}.
        Compute $f_k = f(x_k)$
    end while
    Set $x_{ac} \leftarrow x_k$ and $f_{ac} \leftarrow f(x_k)$.
    Compute $g_{ac} = g(x_{ac})$ and d_k.
end while
return x_{ac} the local minimizer and its function value f_{ac}

For more details about the gradient method and the backtracking line-search approach see [23]. The second approach of the GLMSA algorithm is presented in the following subsection.

2.2. The Second Approach (Simulated Annealing SA)

It must be noted that the modified simulated annealing algorithm in [23] contains three alternatives to generate a new point, but in this paper, the first alternative is considered to generate a new point. This procedure is very important for reducing the function evaluations from three times at each iteration to one function evaluation for every iteration, because we need to allow for more inner iterations when solving constrained optimization problems. This procedure guarantees that the parameters of the penalty function are increasing enough because it is a necessary condition for non-stationary penalty functions [28], i.e., when $k \to \infty$, parameters must also go to infinity.

The second point is generated by

$$x_{k'+1} = x_{ac} + \psi_{k'}, \tag{5}$$

where x_{ac} is the best point which is accepted so far and $\psi_{k'}$ is the step of the second approach and computed by Algorithm 2 below.

The gradient line-search algorithm (GL) has been listed in Algorithm 1 and a modified simulated annealing algorithm (MSA) is illustrated by Algorithm 3.

Algorithm 2 The second approach to generate the step $\psi_{k'}$.

Step 1: Set $k' = 0$.
Step 2: Compute $\omega_{k'} = 10^{(0.1*k')}$.
Step 3: Generate a random vector $X'_k \in [-1, 1]^n$.
Step 4: Compute $D^i_{k'} = \frac{-1+(1+\omega_{k'})^{|X^i_{k'}|}}{\omega_{k'}}$, $i = 1, 2, \ldots, n$. ▷ n is the number of variables.
Step 5: Set $DX^i_{k'} = sign(X^i_{k'})$.
Step 6: Compute $DE^j_{k'} = D^i_{k'} * DX^j_i$.
Step 7: Compute $\psi^i_{k'} = b_i * DE^i_{k'}$. ▷ b_i is the upper bound of the feasible search space.
Step 8: $k' \leftarrow k' + 1$.
Step 9: Repeat steps 2–8 until $k' = N$. ▷ N is the number of iterations and it is given in advance.

Algorithm 3 Modified Simulated-Annealing "MSA".

Input: x_{ac}, f_{ac}, N and T. ▷ T control parameter (Temperature)
Output: x_{best} is the best point of N points and it value f_{best}

for $k' = 0 \rightarrow N$ **do**
  $x_{k'} = x_{ac} + \psi_{k'}$, using Equation (5).
  Compute $\Delta f = f(x_{k'}) - f_{ac}$.
  if $\Delta f < 0$ **then**
    Set $x_{ac_{k'}} \leftarrow x_{k'}$, $f_{ac_{k'}} \leftarrow f(x_{k'})$.
  else
    Generate a random number $\beta \in (0, 1)$
    if $\beta < e^{-\frac{\Delta f}{T}}$ **then**
      Set $x_{ac_{k'}} \leftarrow x_{k'}$, $f_{ac_k} \leftarrow f(x_{k'})$.
    end if
  end if
end for
return x_{ac} and its function value f_{ac}. ▷ $f_{ac} = f(x_{ac})$.

where N is the maximum number of possible trials (Length Markov Chains of MSA) and T is the control parameter (temperature). For more details about the MSA algorithm, please, see [23].

For a detailed description of the simulated annealing algorithm SA see for example [18,29–31].

As we have mentioned above, Algorithm 1 (gradient line-search algorithm (GL)) is hybridized with Algorithm 3 (a modified simulated annealing algorithm (MSA)) to get the LGMSA algorithm that solves the unconstrained optimization problem.

In the next section, the LGMSA algorithm is guided to solve Problem (1) by using the penalty function method. There are many methods for handling the existence of the constraints in the constrained problem.

3. Constraints Handling

The algorithms which have been proposed to solve unconstrained optimization problems are unable to deal directly with constrained optimization problems. There are several approaches proposed to handle the existence of the constraints, see for example [27,32,33]. The most popular of them is the penalty function method.

The penalty function method is a successful technique for handling constraints [27,34,35].

3.1. Penalty Function Methods

The penalty methods have been most widely studied and used due to their simplicity in implementation. The major definition of the penalty function methods is the degree to which each constraint is penalized [28]. There are several types of penalty methods that are used to penalize the constraints in constrained optimization problems.

Three groups of penalty function methods are most popular; the first one is a group of methods of static penalties. In these methods, the penalty parameter does not depend on the current iteration, i.e., parameters remain constant through the evolutionary process [24,36].

The second one is a set of methods of dynamic penalties. In these methods the penalty parameters are usually dependent on the current iteration, in other words, the penalty parameters are functions in the iteration k, i.e., they are non-stationary. See [24,37,38].

The third is a set of methods of adaptive penalties; in this group penalty parameters are updated for every iteration [24].

The next section presents a suggested penalty function method with dynamic and adaptive parameters.

Proposed Penalty Function Method

This section shows how Problem (1) is transformed to an unconstrained optimization problem which is simple bounded as follows:

$$\begin{array}{ll} \min\limits_{x\in\mathbb{R}^n} & \theta(x,r) = f(x) + rp(x), \\ s.t & a_i \leq \ x_i \ \leq b_i, \quad i = 1,2,\ldots,n, \end{array} \tag{6}$$

where $f(x)$ is the original objective function in Problem (1), r is a penalty parameter. The penalty term $p(x)$ is defined by:

$$p(x) = \sum_{l=1}^{q} \left(max\{0, g_l(x)\}\right)^2 + \sum_{j=1}^{m} |h_j(x)|^2. \tag{7}$$

The difference between the penalty function methods is in the way of defining the penalty term and its parameter r [24].

The penalty function methods force infeasible points toward the feasible region by step-wise increasing the penalty; r is used in the penalizing function $p(x)$.

Therefore, the solution x^* minimizes the objective function of Problem (6) and also minimizes the objective function of Problem (1), i.e., as long as $k \to \infty$ and $r_k \to \infty$, x^* approaches the feasible region and $r_k p(x) \to 0$ [28].

In this paper, the penalty function method has two parameters—the first one is r which penalizes the inequality constraint that is violated , i.e., when $g_l(x) > 0$. The second parameter is t which penalizes the equality constraint $h_j(x)$ whose value is not equal to zero.

Accordingly, the $\theta(x,r)$ function is defined by:

$$\theta(x,r) = f(x) + \frac{r}{2}p_1(x) + \frac{t}{2}p_2(x), \tag{8}$$

where $p_1(x) = \sum\limits_{l=1}^{q} \left(max\{0, g_l(x)\}\right)^2$, $p_2(x) = \sum\limits_{j=1}^{m} |h_j(x)|^2$ and r and t are the parameters for inequality and equality constraints respectively.

The parameters r and t are updated at each iteration k as follows.

$$\begin{array}{l} r_{k+1} = r_k + \varphi_k * \Phi_k, \\ t_{k+1} = t_k + 1, \end{array} \tag{9}$$

where the parameter φ_k is updated by:

$$\varphi_k = \begin{cases} 0 & \text{if } g_l(x) \leq 0, \\ 2 & \text{otherwise.} \end{cases} \tag{10}$$

The parameter φ_k is an adaptive parameter, i.e., when the candidate solutions are out of the feasible region then φ_k penalizes a violated constraint by multiplying the term Φ_k by 2, where $r_0 = 1$ is the initial value of r. The parameter Φ is updated as follows: $\Phi_{k+1} = \Phi_k + 1$, $t_0 = 1$.

Note: The equality constraint is more difficult than the inequality constraint because the size of the feasible region of the equality constraint is smaller than the size of the feasible region of the inequality constraint. For example, $f(x,y) = xy$ s.t $h(x,y) = x^2 + y^2 - 1 = 0$ and $f(x,y) = xy$ s.t $g(x,y) = x^2 + y^2 - 1 <= 0$. The first problem is much harder than the second because in the first problem the size of the feasible region is the circumference of the circle while in the second problem, the feasible region is the whole disk. So, the parameter $t(k)$ must be taken carefully.

3.2. Mechanism of Working of the Penalty Function Method

The penalty method solves the general Problem (1), during a succession of unconstrained optimization problems.

Let us discuss two examples in order to illustrate how the parameters of the penalty function are run.

The first example is very easy (one dimension); minimize $f(x) = x^2 - 3$ subject to $g(x) = 0.5 - 0.5x \leq 0$, where $S = [-6, 6]$ is the search domain.

If we want to find the optimal solution of the objective function $f(x) = x^2 - 3$ as an unconstrained problem, it is clear that the global solution to this problem is the point $x^* = 0$, such that $f(x^*) = -3$, for $x \in \mathbb{R}$, but when we want to find the optimal solution of the objective function $f(x) = x^2 - 3$ subject to $g(x) \leq 0$, in this case, the problem is very difficult because we have to find the point x^* that minimizes $f(x)$ and at the same time it must satisfy the condition of the constraint $g(x) \leq 0$, which is why we need to apply the penalty function.

Hence, the problem $f(x) = x^2 - 3$ subject to $g(x) = 0.5 - 0.5x \leq 0$ is transformed into $\theta(x,r) = x + \frac{r}{2}(\max\{0, (\frac{1}{2} - 0.5x)\}^2)$, if $g(x) > 0$; ($g(x)$ is violated), the first derivative is computed by the function $\theta(x,r)$; $\frac{d\theta(x,r)}{dx} = 1 - \frac{r}{2}(\frac{1}{2} - 0.5x)$, then $1 - \frac{r}{2}(\frac{1}{2} - 0.5x) = 0$; $x^* = 1 - \frac{4}{r}$, when $r = \{1, 2, 3, \ldots, \infty\}$, then $x^* = \{-3, -1, \frac{-1}{3}, \ldots, 1\}$, $f(x^*) = \{6, -2, \frac{-26}{9} \ldots, -2\}$ and $g(x^*) = \{2, 1, \frac{2}{3}, \ldots, 0\}$, i.e., when $r \to \infty$, $x^* \to 1$, $g(x^*) \to 0$, $rp(x^*) \to 0$, $f(x^*) \to -2$, and $\theta(x^*, r) \to -2$.

Hence, the optimal point is $x^* = 1$, such that $f(x^*) = -2$ and the constraint $g(x^*) = 0$ is satisfied.

Figure 1 illustrates the behavior of the penalty functions; $rp_1(x)$, $rp_2(x)$ and $rp_3(x)$ and the objective function $f(x)$ of the original problem (constrained problem) and the objective function $\theta(x,r)$ of the transformed problem (unconstrained problem) for all $x \in S = [-6,\ 6]$.

Example 2: minimize $-xy$ s.t $g(x,y) = x + 2y - 4 \leq 0$; $\theta(x,y,r) = -xy + \frac{r}{2}(\max\{0, (x + 2y - 4)\}^2)$, if $g(x,y) > 0$; ($g(x,y)$ is violated), the gradient vector is computed by the function $\theta(x,y)$; $g(x,y) = [-y + r(x + 2y - 4), -x + 2r(x + 2y - 4)]$, hence, $(x^*, y^*) = (\frac{2}{1 - \frac{1}{4r}}, \frac{1}{1 - \frac{1}{4r}})$, then $(x^*, y^*) \to (2, 1)$ as $r \to \infty$; this is why it must allow for the parameters r_k and t_k to increase as long as there exists a violated constraint, i.e., when a process of searching for a solution is an infeasible region.

To ensure that the process of searching for the optimal solution remains within the search domain, the interior-point algorithm is used. Therefore, the next section presents a brief description of this technique.

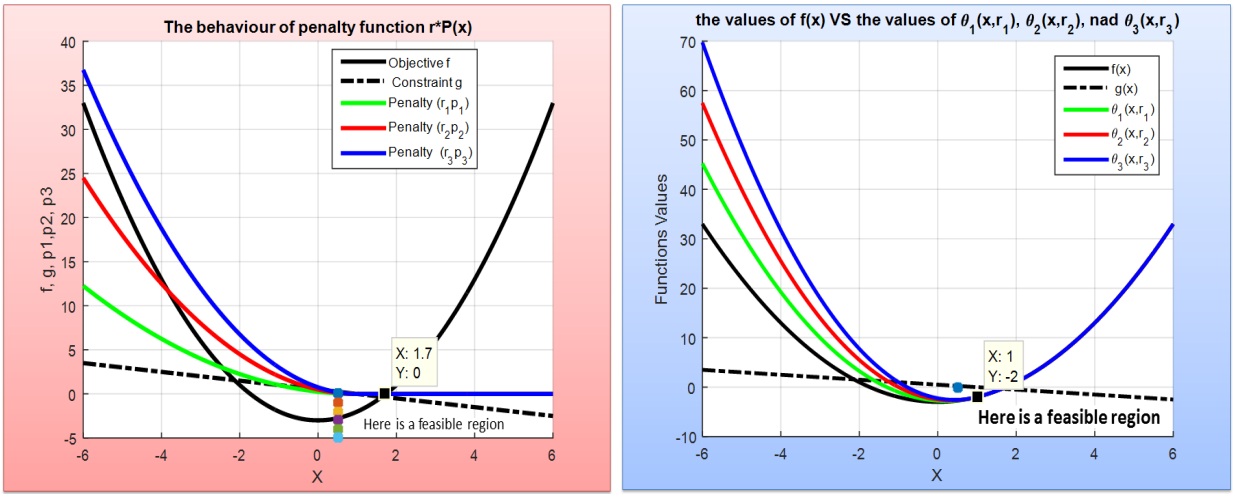

Figure 1. Penalty function $rp(x)$ converges to zero VS $f(x) \to -2$ and $\theta(r, x) \to -2$ that is the optimal solution of the constrained problem.

3.3. Interior-Point Method

The interior-point method is used in this paper, when a simple bounded exists in the test problem. Therefore, the interior point technique is used to ensure that the candidate solution lies inside a feasible region. This technique is used as follows at each iteration k, a damping parameter τ_k is applied to insure that x_{k+1} is feasible with respect to the limits $a_i \leq x_i \leq b_i$, $i = 1, 2, \ldots n$ and $k = 1, 2, \ldots M$ as the inner loop of Algorithm 4, ref. [39].

Algorithm 4 Guided Hybrid Modified Simulated-Annealing Algorithm (GHMSA).

Input: $f(x)$, $g_l(x)$ and $h_d(x) : \mathbb{R}^n \longrightarrow \mathbb{R}$, $x_0 \in \mathbb{R}^n$, M, T, T_f, T_{out}, ε, r_0, Φ_0 and t_0.

set $x_{ac} = x_0$ ▷ at the beginning we accept the initial point x_0 as an optimal solution.
compute $\theta(x_{ac}) = f(x_{ac}) + \frac{r_k}{2} p_1(x_{ac}) + \frac{t_k}{2} p_2(x_{ac})$ ▷ Using Formula (8).
set $\theta_b = \theta(x_{ac})$ and $\theta_\delta = 1$. ▷ The values of θ_b and $\theta_\delta = 1$ are updated after M iterations.
while $\Big(T > T_f$ and $\theta_\delta > \varepsilon\Big)or\Big(T > T_{out}\Big)$ **do** ▷ $T_{out} < T_f \leq 10^{-4}$ are as stopping criteria.
    for $k = 0$ to M **do**
        compute $\theta(x_{ac}) = f(x_{ac}) + \frac{r_k}{2} p_1(x_{ac}) + \frac{t_k}{2} p_2(x_{ac})$.
        set $\theta_{ac} = \theta(x_{ac})$.
        compute $x_1 = x_{ac} + d_k$. ▷ d_k is competed by (16).
        go to Formula (8) to ensure that the point x_1 lies inside $[a, b]^n$▷ by Formula (14).
        compute $\Delta\theta = \theta(x_1) - \theta_{ac}$
        if $\Delta\theta < 0$ **then**
            go to Algorithm 1.
        else
            go to Formula (5) to generate other point.
        end if
    end for
    compute $\Phi_{k+1} = \Phi_k + 1$ ▷ here update penalty parameters.
    $T = r_T * T$ ▷ decrease temperature, where $r_T = 0.8$.
    compute $\theta_\delta = |\theta_b - \theta_{ac}|$ and $\theta_b \leftarrow \theta_{ac}$.▷ θ_δ is a stopping criterion when the solutions converge in the accumulation point for all iterations.
end while
Set $x_g \leftarrow x_{ac}$, $\theta_g \leftarrow \theta_{ac}$
return x_g the global minimizer and the value of the objective function $\theta(x_g)$ at x_g.

The damping parameter τ_k is defined to be:

$$\tau_k = \min\{1, \min_i\{u_k^i, v_k^i\}\}, \tag{11}$$

where

$$u_k^i = \begin{cases} \frac{\left[a^i - x_k^i\right]}{\Delta x_k^i} & \text{if } a^i > -\infty \text{ and } \Delta x_k^i < 0, \\ 1, & \text{otherwise,} \end{cases} \tag{12}$$

$$v_k^i = \begin{cases} \frac{\left[b^i - x_k^i\right]}{\Delta x_k^i} & \text{if } b^i < \infty \text{ and } \Delta x_k^i > 0, \\ 1 & \text{otherwise,} \end{cases} \tag{13}$$

where a^i and b^i are the lower and upper bounds of the domain of the problem respectively, $i = 1, 2, \ldots n$, n is the number of variables of function in problem, x_k^i is the component ith of variable x at iteration k and Δx_k denotes the steps which are obtained by either Formula (2) or by Formula (5).

Since the $\{x_k\}$ is always required to satisfy, for all k, $a < x_k < b$, and then the point x_{k+1} is computed by:

$$x_{k+1} = x_k + 0.99\tau_k \Delta x_k, \tag{14}$$

where the constant 0.99 is a damping parameter to ensure that x_k is feasible with respect to the domain of function in the problem.

4. The Proposed Algorithm for Solving Constrained Optimization Problems (GHMSA)

According to the above procedures the GLMSA Algorithm is capable of solving Problem (1) as a constrained optimization problem during the solving of Problem (6) as an unconstrained optimization problem, hence there are some changes to the objective function $\theta(x, r)$ in Problem (6) to fit with the first step of the GLMSA Algorithm as follows.

- the function $f(x)$ is replaced by the function $\theta(x, r)$ defined in Equation (8), and then calculate

$$\alpha_k = \frac{\theta(x_{ac})}{\| g(x_{ac}, r_k) \|_2^2}, \tag{15}$$

where x_{ac} is the accepted solution at iteration k,

$$d_k = -|\alpha_k| g(x_{ac}, r_k), \tag{16}$$

where the parameter r_k might denote r only or t only or both together according to a type of constrained optimization problem, for example, if the constraints contain mixed constraints inequality and equality, then $r_k = (r_k, t_k)$.

- if the constrained problem contains simple bounded, we use Formula (14) to limit the new point inside this simple bounded.

In light of the above procedures, we rename the GLMSA Algorithm the "Guided Hybrid Modified Simulated-Annealing Algorithm" with the abbreviation "GHMSA".

Setting Parameters of GHMSA Algorithm

The choice of a cooling schedule has an important impact on the performance of the simulated-annealing algorithm. The cooling schedule includes two terms: the initial value of the temperature T and the cooling coefficient r_T which is used to reduce T. Many suggestions have been proposed in the literature for determining the initial value of the temperature T and the cooling coefficient r_T, see for example [4,18,40–42].

In general, it is a unanimous fact that the initial temperature T must be sufficiently high (to ensure escape from local points) and $r_T \in (0.1, 1)$ [7,43,44]. In this section, we suggest that the initial value of T be related to the number of variables and the value of $f(x)$ at

the starting point x_0. The cooling coefficient is taken to be $r_T \in [0.8, 1)$ to decrease the temperature T slowly.

Therefore, the parameters used in Algorithm 4 are presented as follows. M is the inner loop maximum number of iterations, T is the control parameter (Temperature), $Tout$ is a final value of T, r_T is the cooling coefficient and T_f is a final value of T if it is sufficiently small.

The setting of parameters is as follows: $T = 10^4$, $\varepsilon = 10^{-6}$, $T_f = 10^{-14}$, $T_{out} = 10^{-20}$, $r_T = 0.8$, and $M = 10\ n$.

5. Numerical Result

To test the effectiveness and efficiency of the proposed algorithm, the algorithm is run on some test problems. The test problems are divided into two sets. The first set of test problems are taken from [45]. They are 24 well-known constrained real-parameter optimization problems. The objective functions in these problems take different shapes and the number of variables is between 2 and 24. These test problems also contain four types of constraints as follows: (LI) denotes a linear inequality, (LE) is a linear equality, (NI) refers to a nonlinear inequality, and (NE) denotes a nonlinear equality. They are listed in Table 1, where $f(x^*)$ is the best known optimal function value and a denotes the active constraint number at the known optimal solution. "The information mentioned in Table 1 is taken from [46]".

Table 1. List of first and second types of test problems and their exact solutions.

pr	n	$f(x^*)$	Kind of Function	LI	NI	LE	NE	a
G1	13	−15	quadratic	9	0	0	0	6
G3	10	−1.0005001000	polynomial	0	0	0	1	1
G4	5	−30,665.5386717834	quadratic	0	6	0	0	2
G5	4	5126.4967140071	cubic	2	0	0	3	3
G6	2	−6961.8138755802	cubic	0	2	0	0	2
G7	10	24.3062090681	quadratic	3	5	0	0	6
G8	2	−0.0958250415	nonlinear	0	2	0	0	0
G9	7	680.6300573745	polynomial	0	4	0	0	2
G10	8	7049.2480205286	linear	3	3	0	0	6
G11	2	0.7499000000	quadratic	0	0	0	1	1
G12	3	−1.0000000000	quadratic	0	1	0	0	0
G13	5	0.0539415140	nonlinear	0	0	0	3	3
G14	10	−47.7648884595	nonlinear	0	0	3	0	3
G15	3	961.7150222899	quadratic	0	0	1	1	2
G16	5	−1.9051552586	nonlinear	4	34	0	0	4
G18	9	−0.8660254038	quadratic	0	13	0	0	6
G19	15	32.6555929502	nonlinear	0	5	0	0	0
G24	10	−5.5080132716	polynomial	0	0	0	1	1

The GHMSA Algorithm solved 18 test problems out of the 24 because the other problems are either not continuous or not differentiable. The second set of test problems contains four known non-linear engineering design optimization problems. These test problems do not have known exact solutions.

5.1. Results of "GHMSA" Algorithm

The GHMSA algorithm is programmed using MATLAB version 8.5.0.197613 (R2015a) and it is run on a personal laptop and the machine epsilon about 1×10^{-16}.

The results of our algorithm are compared against the results of the CB-ABC Algorithm in [47], the CCiALF Algorithm in [48], the NDE Algorithm in [49] and the CAMDE Algorithm in [50].

Liang et al. [45] suggested that the achieved function error values of the obtained optimal solution x after 5×10^3, 5×10^4 and 5×10^5 function evaluations (FES) are summarized

in terms of {Best, Median, Worst, c, $\overline{v}$ ($\overline{v} = \frac{p(x)}{q+m}$, $p(x)$ is a penalty term in Equation (7)), Mean, s.d}.

The results are listed in Tables 2–4; where c is a concatenation of three numbers indicating the violated constraint number at the median solution by more than 1.0, between 0.01 and 1.0, and between 0.0001 and 0.1, respectively. $\overline{v}$ is the mean value of the violations of all constraints at the median solution. The numbers in the parenthesis after the error value of the Best, Median, Worst solution are the constraint numbers not satisfying the feasible condition of the Best, Median, and Worst solutions, respectively. Tables 2–4 denote that the GHMSA can determine feasible solutions at each run utilizing 5×10^3 FES for 12 test problems {G01, G03, G04, G06, G08, G09, G10, G12, G13, G16, G18, G24}. As for problems G11, G14 and G15, the GHMSA Algorithm finds feasible solutions by using 5×10^4 FES. For the other three test problems, {G05, G07, G19}, the GHMSA Algorithm is able to reach feasible solutions by using 5×10^5 FES.

Assume that if the result x is a feasible one satisfying $(f(x) - f(x^*) \leq 0.0001$, then x is in a neighborhood (near-optimal) of the optimal point $x^* = x_g$. Tables 2–4 indicate that the GHMSA Algorithm can get near-optimal points for six problems, { G01, G04, G06, G08, G12, G24,} by using only 5×10^3 FES, { G03, G11, G13, G14, G15, G16,G18} by using only 5×10^4 FES and { G07, G09, G13, G19} by using only 5×10^5 FES. However, the GHMSA Algorithm failed to satisfy $(f(x) - f(x^*) \leq 0.0001$, for two problems {G05, G10}. As suggested by [45], Table 5 presents the Best, Median, Worst, Mean, and s.d values of successful run, feasible rate, success rate, and success performance over 40 runs. Let us define the following:

Feasible run: A run through which at least one feasible solution is found in Max FES.
Successful run: A run during which the algorithm finds a feasible solution x satisfying $(f(x) - f(x^*) \leq 0.0001$.
Feasible rate (f.r) = (# of feasible runs)/ total runs.
Successrate(s.r) = (# of successful runs) / total runs.
Successperformance(s.p) = mean (FES for successful runs) × (# of total runs)/(# of successful runs).

Table 2. Error values achieved if FES $= 5 \times 10^3$, FES $= 5 \times 10^4$, FES $= 5 \times 10^5$ for G1, G3, G4, G5, G6 and G7.

FES		G1	G3	G4	G5	G6	G7
	Best	1.93×10^{-05} (0)	2.70×10^{-04} (0)	2.68×10^{-09} (0)	0.02 (3)	8.00×10^{-08} (0)	−1.26 (8)
	Median	8.44×10^{-05} (0)	9.34×10^{-04} (0)	1.5×10^{-05} (0)	0.41 (3)	3.7×10^{-06} (0)	0.206 (8)
	Worst	9.99×10^{-05} (0)	1 (0)	4.25×10^{-05} (0)	13.18 (3)	2.89×10^{-04} (0)	28.16 (8)
5×10^3	c	0, 0, 0	0	0, 0, 0	0, 3, 3	0, 0, 0	0, 8, 8
	$\overline{v}$	0	4.22×10^{-04}	0	0.02	0	0.016
	Mean	8.15×10^{-05}	1.77×10^{-01}	1.9×10^{-05}	1.97	1.7×10^{-05}	6.644
	s.d	1.70×10^{-05}	0.380881	1.48×10^{-05}	3.44	5.5×10^{-05}	9.222
	Best	0 (0)	3.62×10^{-06} (0)	1.09×10^{-11} (0)	0.0016 (3)	8×10^{-08} (0)	−0.07 (4)
	Median	0 (0)	3.64×10^{-06} (0)	8.00×10^{-11} (0)	0.02 (3)	1×10^{-06} (0)	-4.55×10^{-03} (4)
	Worst	0 (0)	3.99×10^{-06} (0)	9.82×10^{-11} (0)	1.32 (3)	4×10^{-05} (0)	0.819 (4)
5×10^4	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 3	0, 0, 0	0, 0, 0, 4
	$\overline{v}$	0	1.51×10^{-06}	0	3.45×10^{-04}	0	2×10^{-04}
	Mean	0	3.71×10^{-06}	6.90×10^{-11}	0.166	5×10^{-06}	−0.04
	s.d	0	1.24×10^{-07}	2.86×10^{-11}	0.3237	8×10^{-06}	0.004
	Best	0 (0)	9.99×10^{-07} (0)	1.09×10^{-11} (0)	0.0016 (0)	8×10^{-08} (0)	-1×10^{-04} (0)
	Median	0 (0)	2.58×10^{-06} (0)	8.00×10^{-11} (0)	0.0233 (0)	1×10^{-06} (0)	-3×10^{-05} (0)
	Worst	0 (0)	8.50×10^{-06} (0)	9.82×10^{-11} (0)	1.3182 (0)	4×10^{-05} (0)	410^{-05} (0)
5×10^5	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	$\overline{v}$	0	4.05×10^{-07}	0	3.45×10^{-05}	0	2.3×10^{-05}
	Mean	0	2.37×10^{-06}	6.90×10^{-11}	0.166	5×10^{-06}	-4×10^{-05}
	s.d	0	1.92×10^{-06}	2.86×10^{-11}	0.03237	8×10^{-06}	4×10^{-05}

Table 3. Error values achieved when FES $= 5 \times 10^3$, FES $= 5 \times 10^4$, FES $= 5 \times 10^5$ for Problems G8, G9, G10, G11, G12 and G13.

FES		G8	G9	G10	G11	G12	G13
	Best	1.05×10^{-10} (0)	1.0467 (0)	4.77 (0)	1.9×10^{-04} (1)	0 (0)	1.25×10^{-04} (0)
	Median	6.52×10^{-09} (0)	1.494 (0)	17.67 (0)	1.04×10^{-03} (1)	0 (0)	3.83×10^{-03} (0)
	Worst	4.13×10^{-08} (0)	3.42 (0)	300.41 (0)	5.66×10^{-03} (1)	0 (0)	9.83×10^{-02} (0)
5×10^3	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 1	0, 0, 0	0, 0, 0
	$\overline{v}$	0	0	0	0.00173	0	7.27×10^{-05}
	Mean	9.24×10^{-09}	1.863	49.86	0.00161	0	0.01171
	Std	1.20×10^{-08}	0.9631	75.17	0.00151	0	0.02013
	Best	1.05×10^{-10} (0)	0.3489 (0)	0.10 (0)	6.67×10^{-05} (0)	0 (0)	8.96×10^{-06} (0)
	Median	6.52×10^{-09} (0)	4.98×10^{-01} (0)	0.35 (0)	9.60×10^{-05} (0)	0 (0)	6.93×10^{-05} (0)
	Worst	4.13×10^{-08} (0)	1.14 (0)	6.01 (0)	9.96×10^{-05} (0)	0 (0)	2.94×10^{-04} (0)
5×10^4	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	$\overline{v}$	0	0	0	4.72×10^{-06}	0	2.02×10^{-06}
	Mean	9.24×10^{-09}	6.21×10^{-01}	1.00	9.60×10^{-05}	0	4.39×10^{-05}
	Std	1.20×10^{-08}	0.321034	1.50	1.70×10^{-06}	0	6.53×10^{-05}
	Best	1.05×10^{-10} (0)	6.58×10^{-05} (0)	0.02 (0)	6.67×10^{-05} (0)	0 (0)	8.50×10^{-06} (0)
	Median	6.52×10^{-09} (0)	8.53×10^{-05} (0)	0.09 (0)	9.60×10^{-05} (0)	0 (0)	5.70×10^{-05} (0)
	Worst	4.13×10^{-08} (0)	9.88×10^{-05} (0)	1.50 (0)	9.96×10^{-05} (0)	0 (0)	9.90×10^{-05} (0)
5×10^5	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	$\overline{v}$	0	0	0	4.72×10^{-06}	0	4.90×10^{-07}
	Mean	9.24×10^{-09}	8.38×10^{-05}	0.25	9.60×10^{-05}	0	5.50×10^{-05}
	Std	1.20×10^{-08}	1.52×10^{-05}	0.38	1.70×10^{-06}	0	2.70×10^{-05}

Table 4. Error values achieved when FES $= 5 \times 10^3$, FES $= 5 \times 10^4$, FES $= 5 \times 10^5$ for Problems G14, G15, G16, G18, G19 and G24.

FES		G14	G15	G16	G18	G19	G24
	Best	4.25×10^{-01} (3)	-5.83×10^{-02} (2)	4.31×10^{-04} (0)	0.01 (0)	-5.58×10^{-02} (3)	5.10×10^{-12} (0)
	Median	1.4 (3)	0.18 (2)	0.0064 (0)	0.21 (0)	4.28×10^{-01} (3)	9.05×10^{-12} (0)
	Worst	1.53 (3)	55.40 (2)	0.0181 (0)	0.79 (0)	27.6 (3)	9.99×10^{-12} (0)
5×10^3	c	0, 3, 3	0, 1, 2	0, 0, 0	0, 0, 0	0, 0, 3	0, 0, 0
	$\overline{v}$	2.62×10^{-02}	1.16×10^{-03}	0	0	9.80×10^{-03}	0
	Mean	1.23	2.60	0.0076	0.27	4.07	8.57×10^{-12}
	s.d	0.40	10.79	0.0046	0.21	7.69	1.32×10^{-12}
	Best	2.85×10^{-07} (0)	1.12×10^{-07} (0)	7.70×10^{-11} (0)	4.51×10^{-06} (0)	-5.58×10^{-03} (3)	5.10×10^{-12} (0)
	Median	4.68×10^{-05} (0)	6.96×10^{-06} (0)	8.40×10^{-11} (0)	7.97×10^{-05} (0)	4.28×10^{-02} (3)	9.05×10^{-12} (0)
	Worst	9.11×10^{-05} (0)	4.75×10^{-04} (0)	8.90×10^{-11} (0)	9.88×10^{-05} (0)	2.76 (3)	9.99×10^{-12} (0)
5×10^4	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 3	0, 0, 0
	$\overline{v}$	3.62×10^{-05}	8.10×10^{-06}	0	0	9.80×10^{-04}	0
	Mean	5.17×10^{-05}	5.08×10^{-05}	8.40×10^{-11}	6.83×10^{-05}	4.07×10^{-01}	8.57×10^{-12}
	s.d	2.42×10^{-05}	1.20×10^{-04}	3.40×10^{-12}	2.65×10^{-05}	0.76855	1.32×10^{-12}
	Best	2.85×10^{-07} (0)	1.12×10^{-07} (0)	7.70×10^{-11} (0)	4.51×10^{-06} (0)	-9.94×10^{-05} (0)	5.10×10^{-12} (0)
	Median	4.68×10^{-05} (0)	6.96×10^{-06} (0)	8.40×10^{-11} (0)	7.97×10^{-05} (0)	-3.21×10^{-05} (0)	9.05×10^{-12} (0)
	Worst	9.11×10^{-05} (0)	4.75×10^{-04} (0)	8.90×10^{-11} (0)	9.88×10^{-05} (0)	8.70×10^{-05} (0)	9.99×10^{-12} (0)
5×10^5	c	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	$\overline{v}$	3.62×10^{-05}	8.10×10^{-06}	0	0	8.09×10^{-05}	0
	Mean	5.17×10^{-05}	5.08×10^{-05}	8.40×10^{-11}	6.83×10^{-05}	-1.86×10^{-05}	8.57×10^{-12}
	s.d	2.42×10^{-05}	1.20×10^{-04}	3.40×10^{-12}	2.65×10^{-05}	6.76×10^{-05}	1.32×10^{-12}

Table 5 shows that the GHMSA Algorithm obtains a 100% feasible rate and success rate for all 18 problems with the exception of problems G05 and G10.

Table 5. Number of FES to achieve the fixed accuracy level $((f(x) - f(x^*)) \leq 0.0001)$, success rate, feasible rate and success performance.

pr	Best	Median	Worst	Mean	s.d	f.r (%)	s.r (%)	s.p
G1	1964	2360	2748	2386.68	172.3774278	100	100	2386.68
G3	7681	11,558	13,545	11,566.82353	1167.105632	100	100	11,566.82353
G4	1906	4417	4924	4295.6	563.9451037	100	100	4295.6
G5	-	-	-	-	-	0	0	-
G6	3628	4455	5409	4388.851852	390.8762903	100	100	4388.851852
G7	34,773	210,502	500,559	259,738.33	224,164.15	100	100	259,738.33
G8	794	1108	1350	1109.9615	133.37526	100	100	1109.9615
G9	417,565	4.33×10^{05}	495,232	444,552.9	31,764.59	100	100	444,552.9
G10	-	-	-	-	-	0	0	-
G11	6248	8146	9877	8233.92	917.8058	100	100	8233.92
G12	117	230	339	226.6	57.190209	100	100	226.6
G13	12,800	37,261	80,814	42,242.04	17,190.94051	100	100	42242.04
G14	28,366	54,687	71,293	52,486.30769	16,047.27821	100	100	52,486.30769
G15	9258	25,435	90,720	30,647.44	19,355.55703	100	100	30,647.44
G16	5758	9199	11,398	8970.76	1060.014463	100	100	8970.76
G18	10,300	36,198	85,882	42,434.56	20,906.48809	100	100	42,434.56
G19	73,800	193,000	499,000	247,000	187,852.1	100	100	247,000
G24	537	755.5	999	744.846154	103.587456	100	100	744.846154

For achieving the success condition during the view of success performance in Table 5, the GHMSA Algorithm needs:

(1) $117 \leq \text{FES} \leq 4.924 \times 10^3$ for 5 problems i.e., {G01, G04, G08, G12, G24}.

(2) $3628 \leq \text{FES} \leq 1.4 \times 10^4$ for 4 problems i.e., {G03, G06, G11, G16}.

(3) $9258 \leq \text{FES} \leq 90{,}720$ for 4 problems i.e., {G13, G14, G15, G18}.

(4) $34{,}773 \leq \text{FES} \leq 500{,}559$ for 3 problems i.e., {G07, G09, G19}.

The GHMSA Algorithm failed to achieve the success condition for two problems, i.e., {G05, G10}. More information about the performance of the GHMSA Algorithm for solving these problems is given in Figures 2–4. We have plotted the relationship between $log_{10}(f(x) - f(x^*))$ and FES for showing the convergence of the GHMSA at the median run over 40 independent runs. So the convergence graphs of these problems in Figures 2–4 show that the error values decrease dramatically with increasing FES for all test problems.

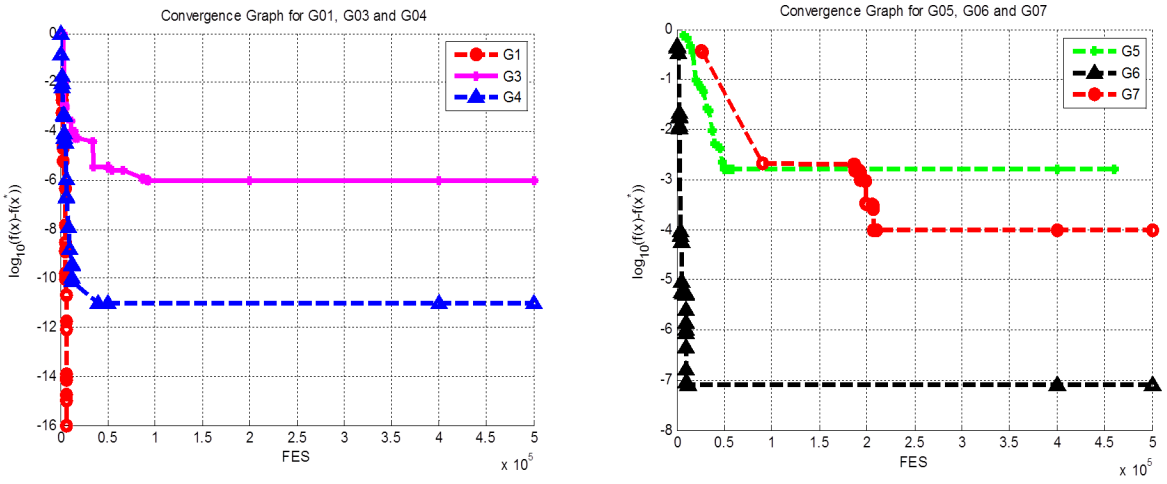

Figure 2. Convergence graph for G01 to G07.

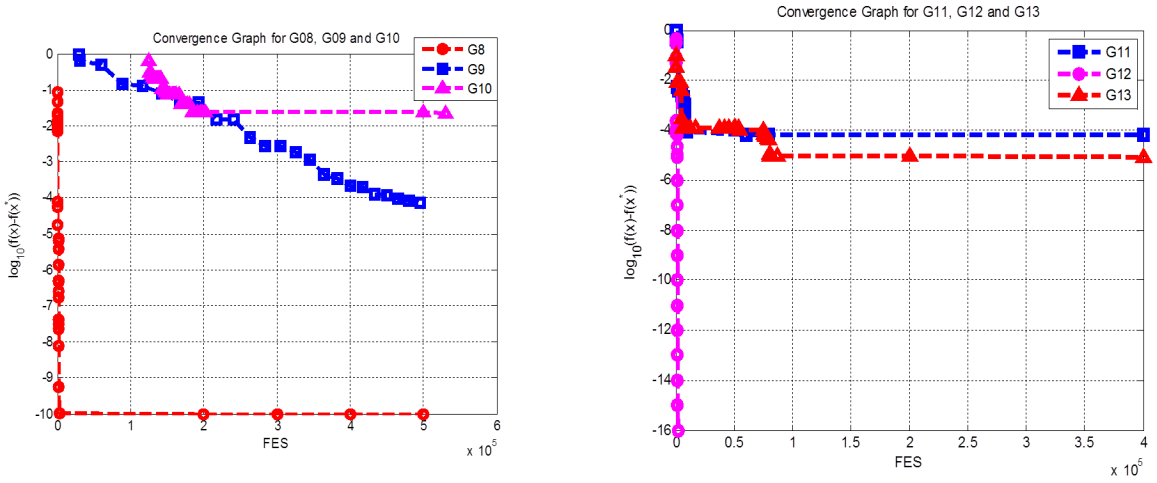

Figure 3. Convergence graph for G08 to G13.

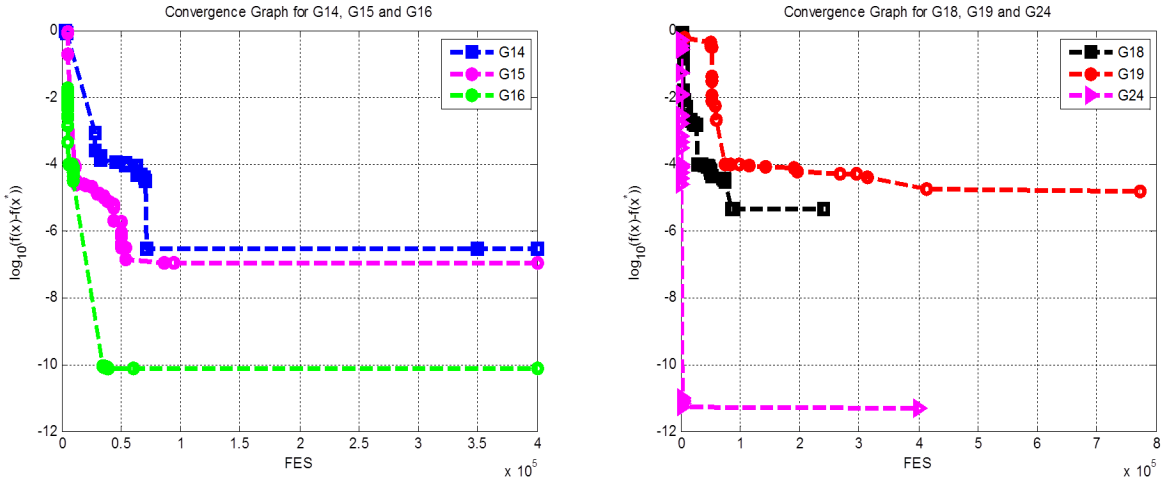

Figure 4. Convergence graph for G14 to G24.

5.2. Performance of GHMSA Algorithm Using Statistical Hypothesis Testing

In this section, we use statistical hypothesis testing to evaluate the efficiency of the GHMSA Algorithm versus the efficiency of the CB-ABC, the CCiALF, the NDE and the CAMDE Algorithms.

A statistical hypothesis is a surmise about a population parameter. This expectation might be true or false. The null hypothesis is denoted by H_0, and it is a statistical hypothesis that announces that there is no difference between a parameter and a specific value or that there is no difference between two parameters. The alternative hypothesis is indicated by H_a, and it is a statistical hypothesis that declares a specific difference between a parameter and a specific value or states that there is a difference between two parameters. Hypothesis testing is a form of inferential statistic which authorizes us to draw conclusions on a whole population based on a representative sample [51]. Parametric tests can provide trustworthy results with distributions that are skewed and non normal. Parametric analysis can produce reliable results even if the continuous data are non normally distributed. We just have to be sure that the sample size is greater than 30. A one sample *t*-test is one of the parametric tests that is used to compare the mean (Average) of a sample with a mean of the population. The important conditions for using the one-sample *t*-test are independence and normality (or sample size > 30). In our study the sample size is 50, i.e., the number of runs is 50 randomly (from any starting point) run; this criterion is suggested by [45]. The significance level in this study is 95%, i.e., $\alpha = 0.05$. Our hypotheses are formulated in the following:

H_0: the mean (average) of the results of the GHMSA Algorithm and the mean (average) of the results of other algorithms are equal.

H_a: the mean (average) of the results of the GHMSA Algorithm and the mean of the results of other algorithms are different.

The above hypotheses can be formulated in Equation (17).

$$\begin{aligned} H_0 &: Me_{GHMSA} = Me_{Algorithm_l}, \\ H_a &: Me_{GHMSA} \neq Me_{Algorithm_l}, \end{aligned} \tag{17}$$

where l denotes one of the algorithms, CB-ABC, CCiALF, NDE and CAMDE, and Me denotes the average results of the algorithms.

In order to compare the performance of the GHMSA Algorithm with the CB-ABC, the CCiALF, the NDE and the CAMDE Algorithms, the t-test with a significance level of $\alpha = 0.05$ is performed. To perform the t-test, the hypotheses in Equation (17) are considered.

Statistical processes are performed by using the SPSS Program. Rejecting or accepting H_0 is based on the value of the p-value (Sig. (2-tailed)) according to Column 1 of Table 6. While the performance of the algorithm based on the value of the t-test is in Column 3 of Table 6. So, Column 4 of Table 6 takes three values according to the probabilities in (18).

$$\text{Decision} = \begin{cases} 1 & \text{then } Me_{GHMSA} < Me_{Algorithm_{al}}, \\ -1 & \text{then } Me_{GHMSA} > Me_{Algorithm_l}, \\ 0 & \text{then } Me_{GHMSA} = Me_{Algorithm_l}. \end{cases} \tag{18}$$

The results of the GHMSA are compared to the results of the CB-ABC, the CCiALF, the NDE and the CAMDE Algorithms. The statistical hypotheses in Equation (17) are tested by using the t-test. Tables 7–10 present these results.

The results of the GHMSA are compared versus the four meta-heuristic algorithms in the literature. The results of statistical tests are presented in Tables 7, 9 and 10. In Table 7, Column 1 presents the abbreviation of the test problems denoted by pr. Column 2 presents the results of the s.t which include {b.s, mean, s.d, Decision }, where Decision denotes wins, losses and draws of the GHMSA compared with the other algorithms. Columns 3–7 give the results of the five algorithms. Tables 9 and 10 are similar to Table 7.

After executing the pairwise t-test for all algorithms, if the GHMSA Algorithm is superior, inferior or equal to the compared algorithm denoted by $algorithm_l$, then the decision is set to 1, –1 and 0 respectively, as we have shown in Table 6. The left of Figure 5 summarizes the results that are presented in Tables 7–10 regarding the decision. The left of Figure 5 shows that the GHMSA Algorithm was superior at {7, 6, 9, 5 } problems, equal at {6, 6, 3, 5 } problems and inferior at {4, 5, 5, 7 } problems compared to the CB-ABC, the CALF, the NDE and the CAMDE Algorithms, respectively. However, the GHMSA is inferior at seven problems compared to the CAMDE, but the GHMSA needs 1,590,905 as a total FES versus the CAMDE needing 4,320,000, as shown in Figure 6. To gain the success condition from the point of view of successful execution, the GHMSA needs less than 5×10^3 FES for five problems, i.e., {G01, G04, G08, G12, G24} versus the CAMDE needing at least 5×10^3 FES for two problems, i.e., {G08, G12}. We can say that the percentage of superior, equal and inferior of the GHMSA are 40%, 30%, 30% respectively.

Table 6. How the null hypothesis is rejected (or accepted) and the decision is made.

p-Value	H_0	t	Decision
$<\alpha$	reject	<0	1
$<\alpha$	reject	>0	−1
$>\alpha$	accept	-	0

Table 7. Comparison of results for test problems G01 to G08.

pr	s.t	CB-ABC	CCiALF	NDE	CAMDE	GHMSA
G1	b.s	−15	−15	−15	−15	−15
	mean	−15	−15	−15	−15	−15
	s.d	5.03×10^{-15}	2.39×10^{-08}	0	0	0
	decision	0	0	0	0	0
	FES	135,180	30,819	240,000	240,000	5773.84
G3	b.s	−1.0005	−1.000501	−1.0005001	−1.000500	−1.000009
	mean	−1.0005	−1.000501	−1.0005001	−1.000500	−1.000002
	s.d	3.64×10^{-07}	1.69×10^{-08}	0	6.80×10^{-16}	1.92×10^{-06}
	decision	‡	‡	‡	‡	
	FES	90,090	87,860	240,000	240,000	62,546.38462
G4	b.s	−30,665.54	−30,665.539	−30,665.539	−30,665.53867	−30,665.53867
	mean	−30,665.54	−30,665.539	−30,665.539	−30,665.53867	−30,665.53867
	s.d	8.72×10^{-11}	9.80×10^{-06}	0	3.71×10^{-12}	3.49×10^{-07}
	decision	0	0	0	0	
	FES	45,045	26,268	240,000	240,000	9671.32
G5	b.s	5126.50	5126.4967	5126.49671	5126.496710	5126.49833
	mean	5126.50	5126.497	5126.49671	5126.496710	5126.662712
	s.d	1.07×10^{-10}	9.17×10^{-08}	0	2.78×10^{-12}	0.03442
	decision	−1	−1	−1	−1	
	FES	135,180	156,248	240,000	240,000	33,917.7702
G6	b.s	−6961.81	−6961.814	−6961.813875	−6961.81388	−6961.813826
	mean	−6961.81	−6961.814	−6961.813875	−6961.81388	−6961.813811
	s.d	1.82×10^{-12}	5.19×10^{-11}	0	0	9.20×10^{-06}
	decision	1	0	1	1	
	FES	45,045	17,573	240,000	240,000	8921.518519
G7	b.s	24.3062	24.3062	24.306209	24.30621	24.30610911
	mean	24.3062	24.3062	24.306209	24.30621	24.30617
	s.d	4.16×10^{-07}	6.82×10^{-07}	1.35×10^{-14}	8.55×10^{-15}	4.34×10^{-05}
	decision	1	1	1	1	
	FES	135,180	8745	240,000	240,000	259,738.33
G8	b.s	−0.095825	−0.095825	−0.095825	−0.09583	−0.0958141
	mean	−0.095825	−0.095825	−0.095825	−0.09583	−0.0957819
	s.d	2.87×10^{-17}	1.07×10^{-15}	0	1.42×10^{-17}	2.58×10^{-05}
	decision	1	1	1	−1	
	FES	8000	4812	240,000	240, 000	2394.577

The mark ‡ means that we do not use G03 to compare the result of the GHMSA with results of the four algorithms because the $h(x^*) = 0.0001$, i.e., $\overline{v} = 0.0001$ in [45], but $\overline{v}$ for the GHMSA is 4.05×10^{-07}, see Tables 1, 2 and 8.

Table 8. Statistical results of "GHMSA" Algorithm for first set of test problems and four mechanical engineering problems.

pr	Best	Median	Worst	Mean	s.d	FES
G1	−15	−15	−15	−15	0	5773.84
G3	−1.000009	−1.000003	−1.000001	−1.000002	1.91679×10^{-06}	62,546.38462
G4	−30,665.538672	−30,665.538672	−30,665.53867	−30,665.538672	3.49×10^{-07}	9671.32
G5	5126.49833	5126.520053	5127.81491	5126.662712	0.03442	33,917.7702
G6	−6961.813826	−6961.813811	−6961.81377	−6961.813811	9.19642×10^{-06}	8921.518519
G7	24.30610911	24.30618042	24.30625377	24.30617	4.34×10^{-05}	259,738.33
G8	−0.0958141	−0.095824999	−0.09582499	−0.0957819	2.58×10^{-05}	2394.577
G9	680.6301232	680.6301426	680.6301562	680.6301412	1.52×10^{-05}	444,552.9
G10	7049.271862	7049.689888	7049.460323	7049.336552	2.69×10^{-02}	290,146
G11	0.74999176	0.749996	0.75	0.7499961	0.0000001	8233.92
G12	−1	−1	−1	−1	0	1515
G13	0.053950002	0.053998358	0.054040318	0.053996327	2.73×10^{-05}	53,754
G14	−47.76497953	−47.76493525	−47.76488874	−47.76494056	2.51×10^{-05}	52,486.30769
G15	961.71502	961.71502	961.715107	961.7149837	1.31×10^{-04}	38,609.24

Table 8. *Cont.*

pr	Best	Median	Worst	Mean	s.d	FES
G16	−1.905155259	−1.905155259	−1.905155259	−1.905155259	2.06×10^{-10}	36,346.76
G18	−0.866025404	−0.865945746	−0.865926597	−0.865958115	2.85×10^{-05}	42,434.56
G19	32.65549	32.65556	32.65568	32.6555744	6.76×10^{-05}	247,295.25
G24	−5.508013272	−5.508013272	−5.508013272	−5.508013272	1.09×10^{-10}	2460.038
Enp1	5885.332773	5885.332773	5885.332773	5885.332773	2.2×10^{-12}	32,129
Enp2	0.012665233	0.012665268	0.012665243	0.012665334	1.54×10^{-09}	9970
Enp3	1.724852306	1.724852306	1.724852306	1.724852306	1.33×10^{-16}	24,270
Enp4	2994.471066	2994.471066	2994.471066	2994.471066	4.27×10^{-15}	16,764

Table 9. Comparison of results for test problems G09 to G15.

pr	s.t	CB-ABC	CCiALF	NDE	CAMDE	GHMSA
G9	b.s	680.63	680.63	680.630057	680.63006	680.6301232
	mean	680.63	680.63	680.630057	680.63006	680.6301412
	s.d	2.77×10^{-09}	5.43×10^{-08}	0	2.32×10^{-13}	1.52×10^{-05}
	decision	0	0	−1	−1	
	FES	45,045	12,801	240,000	240,000	444,552.9
G10	b.s	7049.25	7049.248	7049.24802	7049.24802	7049.271862
	mean	7049.25	7049.248	7049.24802	7049.24802	7049.336552
	s.d	3.98×10^{-05}	6.04×10^{-07}	3.41×10^{-09}	4.39×10^{-12}	2.69×10^{-02}
	decision	−1	−1	−1	−1	
	FES	135,180	2858	240,000	240,000	240,146
G11	b.s	0.7499	0.749896	0.749999	0.749900	0.74999176
	mean	0.7499	0.749898	0.749999	0.749900	0.7499961
	s.d	1.29×10^{-10}	2.05×10^{-16}	0	1.13×10^{-16}	0.0000001
	decision	−1	−1	1	−1	
	FES	90,090	168,448	240,000	240,000	8233.92
G12	b.s	−1	−1	−1	−1	−1
	mean	−1	−1	−1	−1	−1
	s.d	0	7.76×10^{-11}	0	0	0
	decision	0	0	0	0	
	FES	13,500	17,892	240,000	240,000	1515
G13	b.s	0.053942	0.053942	0.0539415	0.05394	0.053950002
	mean	0.06677	0.053943	0.0539415	0.05394	0.053996327
	s.d	6.91×10^{-02}	4.03×10^{-06}	0	2.32×10^{-17}	2.73×10^{-05}
	decision	1	−1	−1	−1	
	FES	198,270	19,883	240,000	240,000	53,754
G14	b.s	−47.7649	−47.764900	−47.7648885	−47.764890	−47.76497953
	mean	−47.7649	−47.764900	−47.7648885	−47.764890	−47.76494056
	s.d	1.02×10^{-05}	4.04×10^{-08}	5.14×10^{-15}	2.21×10^{-14}	2.51×10^{-05}
	decision	1	1	1	1	
	FES	239,715	152,697	240,000	240,000	52,486.30769
G15	b.s	961.715	961.715	961.7150223	961.715020	961.71502
	mean	961.715	961.715	961.7150223	961.715020	961.7149837
	s.d	2.81×10^{-11}	1.86×10^{-08}	0	5.80×10^{-13}	1.31×10^{-04}
	decision	0	0	1	1	
	FES	135, 180	77, 910	240, 000	240, 000	38, 609.24

For the four engineering problems, we give a brief description. The pressure vessel problem is a practical problem that is often used as a benchmark problem for testing optimization algorithms [52]. The left of Figure 7 shows the structure of this issue, where a cylindrical pressure vessel is capped at both ends by hemispherical heads. The aim of the problem is to find the minimum total cost of fabrication, including costs from a combination of welding, material and forming. The thickness of the cylindrical skin,$x_1(Ts)$, thickness of the spherical head, $x_2(T_h)$, the inner radius, $x_3(R)$, and the length of the cylindrical segment of the vessel, $x_4(L)$, were included as the optimization design variables of the problem. The GHMSA Algorithm obtains these results: x_{GHMSA} = {0.778168641375105, 0.384649162627902, 40.3196187240987, 200}, i.e.,

$f(x_{GHMSA}) = 5885.332774$, c = {0, –3.8858 × 10^{-16}, 1.1642 × 10^{-97}, −40}, i.e., $\overline{v} = 0$; the left of Figure 8 shows a convergence graph of the GHMSA to the best solution for this problem.

Table 10. Comparison of results for test problems G16, G18, G19, G24 and Enp1-Enp4.

pr	s.t	CB-ABC	CCiALF	NDE	CAMDE	GHMSA
G16	b.s	−1.905	−1.905155	−1.90515525	−1.905160	−1.905155259
	mean	−1.905	−1.905155	−1.90515525	−1.905160	−1.905155259
	s.d	7.90×10^{-11}	9.77×10^{-09}	0	4.53×10^{-16}	2.06×10^{-10}
	decision	1	1	1	0	
	FES	45,045	196,196	240,000	240,000	36,346.76
G18	b.s	−0.866025	−0.866026	−0.8660254	−0.86603	−0.866025404
	mean	−0.866025	−0.866026	−0.8660254	−0.86603	−0.865958115
	s.d	1.72×10^{-08}	3.58×10^{-07}	0	4.53×10^{-17}	2.85×10^{-05}
	decision	−1	−1	−1	−1	
	FES	135,180	8742	240,000	240,000	42,434.56
G19	b.s	32.6556	32.655610	32.65559377	32.655590	32.65549
	mean	32.6556	32.660770	32.65562603	32.655590	32.6555744
	s.d	1.88×10^{-05}	2.35×10^{-04}	3.73×10^{-05}	7.11×10^{-15}	6.76×10^{-05}
	decision	0	1	1	0	
	FES	198,270	240,000	240,000	240,000	247,295.25
G24	b.s	−5.508	−5.508013	−5.50801327	−5.508010	−5.508013272
	mean	−5.508	−5.508013	−5.50801327	−5.508010	−5.508013272
	s.d	7.15×10^{-15}	1.30×10^{-08}	0	9.06×10^{-16}	1.09×10^{-10}
	decision	1	1	1	1	
	FES	27,000	6450	240,000	240,000	2460.038
Enp1	b.s	6059.71	6059.714335	6059.714335	6059.714335	5885.332773
	mean	6126.62	6059.714335	6059.714335	6059.714335	5885.332773
	s.d	1.14×10^{02}	1.01×10^{-11}	4.56×10^{-07}	1.22×10^{-06}	2.2×10^{-12}
	Decision	1	1	1	1	
	FES	15,000	12,000	20,000	10,000	32,1290
Enp2	b.s	0.012665	0.012665233	0.012665232	0.012665233	0.01266523
	mean	0.012671	0.012665251	0.012668899	0.012666981	0.01266533
	s.d	1.42×10^{-05}	9.87×10^{-08}	5.38×10^{-06}	3.65×10^{-06}	1.54×10^{-09}
	Decision	1	0	1	1	
	FES	15,000	5000	24,000	10,000	9970
Enp3	b.s	1.724852	1.724852	1.724852309	1.724852	1.724852
	mean	1.724852	1.724852	1.724852309	1.724852	1.724852
	s.d	0	5.11×10^{-07}	3.73×10^{-12}	2.32×10^{-13}	1.33×10^{-16}
	Decision	0	0	1	0	
	FES	15,000	10,000	8000	10,000	24,270
Enp4	b.s	2994.471066	2994.471066	2994.471066	2994.471066	2994.471065
	mean	2994.471066	2994.4710660	2994.47106610	2994.471066	2994.471065
	s.d	2.48×10^{-07}	2.31×10^{-12}	4.17×10^{-12}	2.20×10^{-12}	4.27×10^{-15}
	Decision	0	0	1	0	
	FES	15,000	10,000	18,000	10,000	16,764

Another well-known engineering optimization task is the design of a tension (compression spring) for a minimum weight. This problem has been studied by several authors. For example, [52]. The right of Figure 7 shows a tension (compression spring) with three design variables. It needs to minimize the weight of a tension (compression string) subject to constraints on minimum deflection, shear stress, surge frequency, limits on outside diameter and on design variables. The design variables are the wire diameter, $d(x_1)$, the mean coil diameter, $D(x_2)$, and the number of active coils, $P(x_3)$. The GHMSA obtains these results: x_{GHMSA} = {0.0516890825110813, 0.356718255308635, 11.2889355307237}, i.e., $f(x_{GHMSA}) = 0.01266523279$, c = {-1.55×10^{-10}, 4.44×10^{-16}, –4.05379, –0.72773}, i.e., $\overline{v} = 1.11 \times 10^{-16}$. The convergence graph for Engp2 is presented on the right of Figure 8. The welded beam design optimization problem has been solved by many researchers [52]. The left of Figure 9 shows the welded beam structure which consists of a beam A and the weld required to hold it to member B. The goal of this problem is to minimize the overall cost of fabrication, subject to some constraints. This problem has four design variables—x_1,

x_2, x_3 and x_4—with constraints of shear stress τ, bending stress in the beam σ, buckling load on the bar P_c, and end deflection on the beam δ. The GHMSA obtains these results: $x_{GHMSA} = \{0.205729642092758, 3.4704886133955, 9.03662391715327, 0.205729639752274\}$, i.e., $f(x_{GHMSA}) = 1.7248523060$, $c = \{-9.03 \times 10^{-08}, -4.02 \times 10^{-05}, 2.34 \times 10^{-09}, -3.43298, -0.08073, -0.23554, -8.73 \times 10^{-09}\}$, i.e., $\overline{v} = 3.3429 \times 10^{-10}$. The convergence graph for Engp3 is presented by the left of Figure 10.

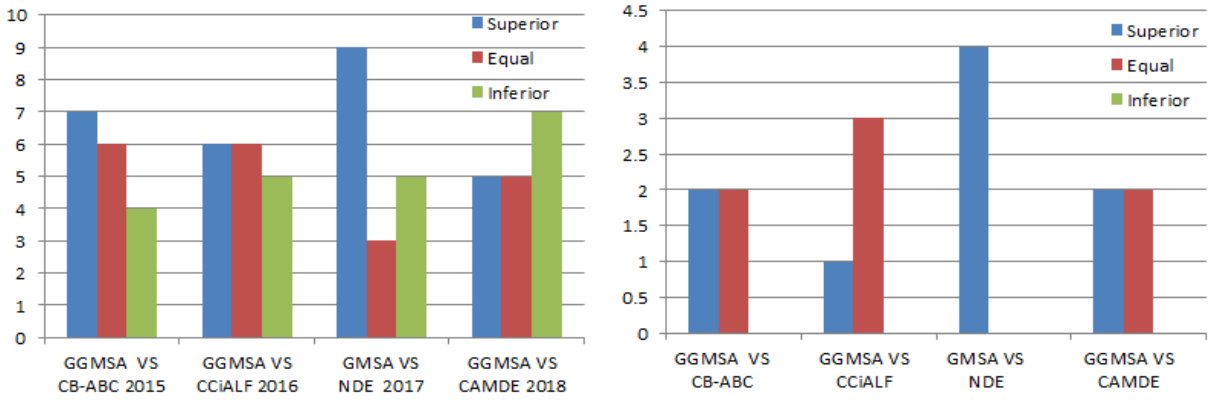

Figure 5. The number of "wins-draws-losses" of GHMSA compared with other algorithms for G01 to G24 and Enp1 to Enp4.

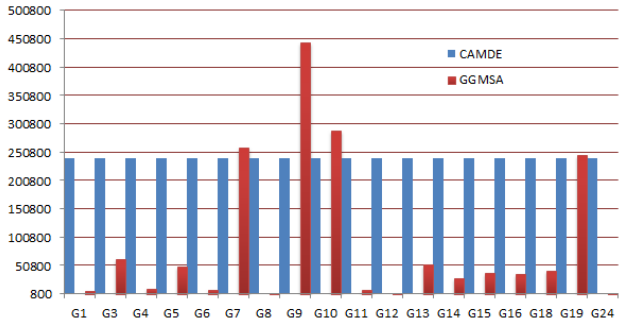

Figure 6. Comparison Between GHMSA With CAMDE Regarding FES.

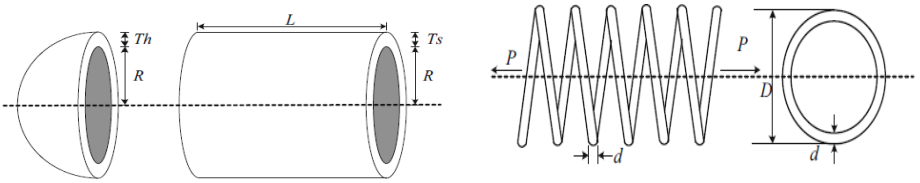

Figure 7. Design engineering problems (Engp1 and Engp2).

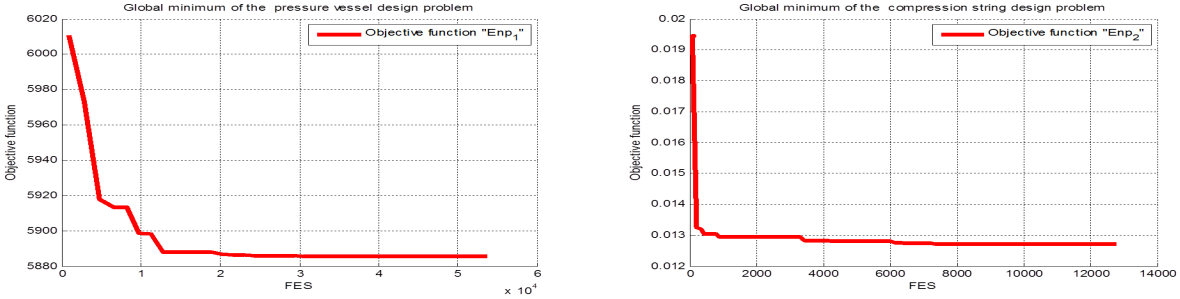

Figure 8. Convergence graph for engineering problems (Engp1 and Engp2).

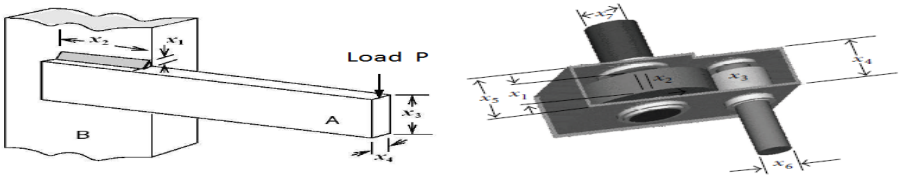

Figure 9. Design engineering problems (Engp3 and Engp4).

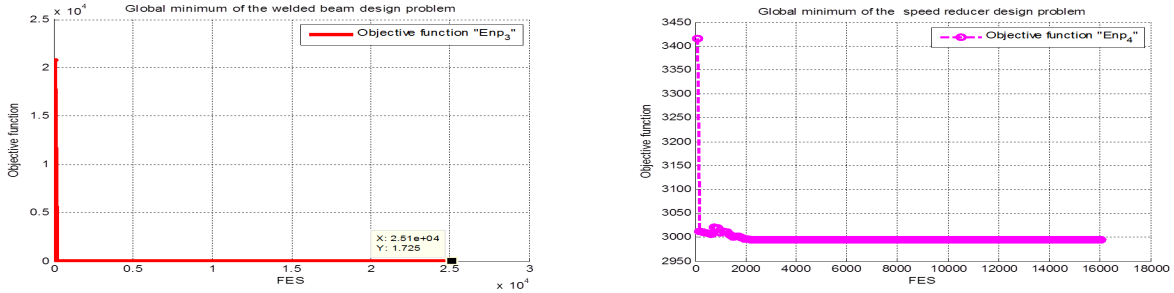

Figure 10. Convergence graph for engineering problems (Engp3 and Engp4).

The speed reducer design problem is one of the benchmark structural engineering problems [52]. It has seven design variables as described in the right of Figure 9, with the face width x_1, module of teeth x_2, number of teeth on pinion x_3, length of the first shaft between bearings x_4, length of the second shaft between bearings x_5, diameter of the first shaft x_6, and diameter of the first shaft x_7. The aim of this problem is to minimize the total weight of the decelerator. The GHMSA obtains these results: $x_{GHMSA} = \{3.499999999, 0.7, 17, 7.3, 7.715319913, 3.350214666, 5.286654465\}$, i.e., $f(x_{GHMSA}) = 2994.471066$, $c = \{-0.073915, -0.198, -0.49917, -0.90464, 8.6365 \times 10^{-11}, -1.0931 \times 10^{-11}, -0.7025, 2.86 \times 10^{-10}, -0.58333, -0.051326, -1.944210^{-10}\}$, i.e, $\overline{v} = 2.6 \times 10^{-11}$. The convergence graph for Engp4 is presented by the right of Figure 10. The four engineering problems are used to compare the performance of the GHMSA against the CB-ABC, the CCiALF, the NDE and the CAMDE Algorithms. Statistical hypotheses in Equation (17) are used to compare the mean of the GHMSA with means of the CB-ABC, the CCiALF, the NDE and the CAMDE Algorithms. Rows 22–41 of Table 10 present the statistical comparisons of the GHMSA versus the four Algorithms for engineering problems Enp1 to Enp4. The right of Figure 5 gives the number of "wins-draws-losses" of the GHMSA compared with the CB-ABC, the CCiALF, the NDE and the CAMDE for Enp1 to Enp4. Figure 11 shows the convergence graph of standard deviation for problems Enp1, Enp2, Enp3 and Enp4 for the five algorithms. The relation between the four engineering problems {Enp1, Enp2, Enp3 and Enp4} and their values for $log_{10}(s.d)$ are plotted. From the right of Figures 5 and 11, it can be said that the performance of the GHMSA algorithm is better than the other algorithms for problems Enp1 to Enp4, for the following reasons:

(1) The GHMSA obtains a minimum value of objective function (5885.332774) for engineering problem Enp1 (pressure vessel), the point minimum x^* is feasible; many of the algorithms obtained a value of objective function equal to or greater than 6059.71. see for example [48–50,52–58].

In addition to that, if $10 \leq x_4(L) < \infty$, then $f(x^*) = 5804.37621675626$, otherwise if $10 \leq x_4(L) < 208$, then $f(x^*) = 5866.99226593889$, where L is shown in the left of Figure 8.

(2) The right of Figure 5 shows that the GHMSA Algorithm does not fall at any problem versus the other algorithms.

(3) The GHMSA is superior at {2, 1, 4, 2} problems versus the CB-ABC Algorithm, the CCiALF Algorithm, the NDE and the CAMDE Algorithm, respectively.

(4) The GHMSA is equal at {2, 3, 0, 2} problems versus the CB-ABC Algorithm, the CCiALF Algorithm, the NDE and the CAMDE Algorithm, respectively.

(5) Figure 11 shows that the GHMSA Algorithm converges to zero for the standard deviation (s.d). See the green color.

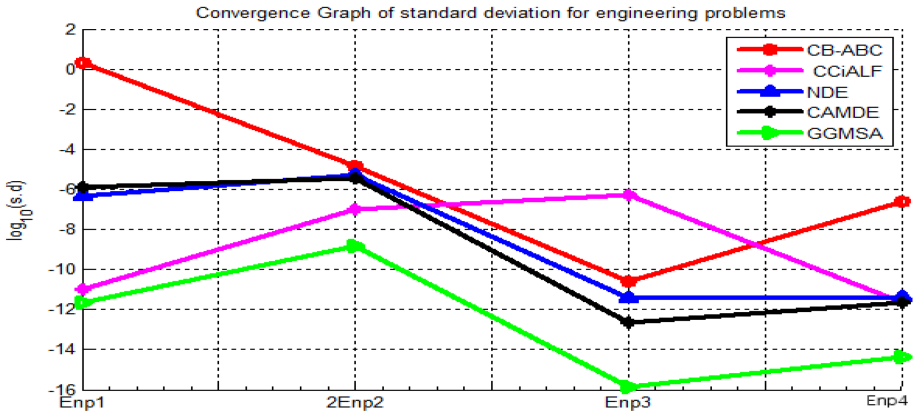

Figure 11. Convergence graph of standard deviation for Enp1 to Enp4.

6. Conclusions and Future Work

The unconstrained nonlinear optimization algorithms have been guided to find the global minimizer of the constrained optimization problem. A result, Algorithm "GHMSA", has been proposed for finding the global minimizer of the non-linear constrained optimization problem. Algorithm "GHMSA" contains a new technique that is applied to convert the constrained optimization problem into the unconstrained optimization problem. The results of the algorithm demonstrate that the proposed penalty function is a good technique to make the unconstrained algorithm able to deal with the constrained optimization problem. The interior-point algorithm keeps the candidate solutions inside the domain search. The results of some nonlinear constrained optimization problems and four nonlinear engineering optimization problems show that the GHMSA algorithm has superiority over the other four algorithms in some test problems. For the future work, the proposed algorithm can be enhanced and modified to solve the multi-objective function, and the convergence analysis of the modified simulated annealing algorithm will be performed.

Moreover, it will be considered in future work to propose a new free derivative to approximate the gradient vector that will be combined (hybridized) with a new simulated annealing algorithm to solve unconstrained optimization, constrained, or multi-objective optimization problems. Convergence analysis of the GMLSA and GHMAS algorithms will be considered in future work.

Author Contributions: M.E.-A.; Formal analysis, M.A.; Funding acquisition, K.A.A.; Investigation, M.A.; Methodology, A.W.M.; Project administration, K.A.A.; Resources, K.A.A.; Supervision, A.W.M.; Validation, M.E.-A.; Writing—original draft, S.M.; Writing—review & editing, S.M. All authors have read and agreed to the published version of the manuscript.

Funding: The Research is funded by Researchers Supporting Program at King Saud University, (Project# RSP-2021/305).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors present their appreciation to King Saud University for funding the publication of this research through Researchers Supporting Program (RSP-2021/305), King Saud University, Riyadh, Saudi Arabia.

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

Abbreviations

The following abbreviations are used in this manuscript:

CB-ABC	Crossover-Based Artificial Bee Colony Algorithm
CCiALF	Cooperative Coevolutionary Differential Evolution Algorithm
NDE	A novel Differential Evolution Algorithm
CAMDE	Adaptive Differential Evolution with Multi-Population-based Mutation Operators for Constrained Optimization
GLMSA	Gradient Line-Search Modified Simulated-Annealing Algorithm
GHMSA	Guided Hybrid Gradient Modified Simulated-Annealing Algorithm
Symbols	
T	Control Parameter (Temperature)
k	Number Iteration
n	Number of Variables
$V \in [-1,1]^n$	A random Vector of n Dimension in Interval $[-1,1]$
x_0	Starting Point
x_1	A point Computed by GHMSA Algorithm
x_2	A point Computed by GHMSA Algorithm
x_{ac}	the Best Point Accepted by Our Algorithm at Iteration k
θ_{ac}	Function Value at Point x_{ac}
θ_1	Function Value at Point x_1
θ_2	Function Value at Point x_2
$\triangle f$	the Difference Between the Value f_{ac} and f_1
M	the Inner Loop Maximum Number of Iterations
ψ	the Step Size which is Generated by First Approach in "EMSA" Algorithm
d	the Step Size which is Generated by GHMSA Algorithm
β	A random Number in $(0,1)$
r_T	the Cooling Coefficient
T_f	A final Value of T it is Sufficiently Small
T_{out}	A final Value of T; $T_{out} < T_{f_2}$
ε	A parameter has Small Value Used as A stopping Criterion
#pr	Number of Test Problems
x_g	Global Minimizer Found by GHMSA Algorithm
$\theta(x_g)$	Function Value at Global Minimum
$g(x)$	the gradient vector
$\parallel g(x_g) \parallel_2$	Norm of the gradient vector of θ at x_g
$p(x)$	Penalty Term
$g_i(x)$	Inequality Constraint
$h_j(x)$	Equality Constraint
q	A number of the Inequality Constraints
m	A number of the Equality Constraints
r	Penalty Parameter for the Inequality Constraints
t	Penalty Parameter for the Equality Constraints
U	Upper Feasible Region (Domain Search)
L	Lower Feasible Region (Domain Search)
$b.s$	the Best Solution Found by the Algorithm
$w.s$	the Worst Solution Found by the Algorithm
$s.d$	the Standard Deviation
$w.b$	Absolute Value Between the Worst Solution and the Best Denoted by $\lvert worst - best \rvert$
er	Absolute Value Between the Best Solution and the Exact Denoted by $\lvert best - exact \rvert$
$e.c$	Error Constraint where $e.c = \max\{0, g_i(x)\} + \max\{0, h_j(x)\}$
$e : v_1$	the Average of $\{s.d, w.b, er\}$
$e : v_2$	the Average of $\{s.d, w.b, e.c\}$

FES	Function Evaluation
c	A sequence of 3 Numbers Denoting the Violated Constraint Number at the Median solution
$\overline{v}$	Is the Mean Value of the Violations of All Constraints at the Median Solution
H_0	the Null Hypothesis
H_a	the Alternative Hypothesis
Me	the Average Results

References

1. Abdel-Baset, M.; Hezam, I. A Hybrid Flower Pollination Algorithm for Engineering Optimization Problems. *Int. J. Comput. Appl.* **2016**, *140*, 12. [CrossRef]
2. Ayumi, V.; Rere, L.; Fanany, M.I.; Arymurthy, A.M. Optimization of Convolutional Neural Network using Microcanonical Annealing Algorithm. *arXiv* **2016**, arXiv:1610.02306.
3. Rere, L.; Fanany, M.I.; Arymurthy, A.M. Metaheuristic Algorithms for Convolution Neural Network. *Comput. Intell. Neurosci.* **2016**, *2016*, 1537325. [CrossRef] [PubMed]
4. Rere, L.R.; Fanany, M.I.; Murni, A. Application of metaheuristic algorithms for optimal smartphone-photo enhancement. In Proceedings of the 2014 IEEE 3rd Global Conference on Consumer Electronics (GCCE), Tokyo, Japan, 7–10 October 2014; pp. 542–546.
5. Samora, I.; Franca, M.J.; Schleiss, A.J.; Ramos, H.M. Simulated annealing in optimization of energy production in a water supply network. *Water Resour. Manag.* **2016**, *30*, 1533–1547. [CrossRef]
6. Agrawal, P.; Ganesh, T.; Mohamed, A.W. A novel binary gaining–sharing knowledge-based optimization algorithm for feature selection. *Neural Comput. Appl.* **2021**, *33*, 5989–6008. [CrossRef]
7. Certa, A.; Lupo, T.; Passannanti, G. A New Innovative Cooling Law for Simulated Annealing Algorithms. *Am. J. Appl. Sci.* **2015**, *12*, 370. [CrossRef]
8. Mohamed, A.A.; Kamel, S.; Hassan, M.H.; Mosaad, M.I.; Aljohani, M. Optimal Power Flow Analysis Based on Hybrid Gradient-Based Optimizer with Moth–Flame Optimization Algorithm Considering Optimal Placement and Sizing of FACTS/Wind Power. *Mathematics* **2022**, *10*, 361. [CrossRef]
9. Nocedal, J.; Wright, S. *Numerical Optimization*; Springer Science & Business Media: Cham, Switzerland, 2006.
10. Aarts, E.; Korst, J. *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*; John Wiley & Sons, Inc.: New York, NY, USA, 1989.
11. Hillier, F.S.; Price, C.C. *International Series in Operations Research & Management Science*; Springer: Berlin/Heidelberg, Germany, 2001.
12. Laarhoven, P.J.V.; Aarts, E.H. *Simulated Annealing: Theory and Applications*; Springer-Science + Business Media, B. V.: Berlin/Heidelberg, Germany, 1987.
13. Kan, A.R.; Timmer, G. Stochastic methods for global optimization. *Am. J. Math. Manag. Sci.* **1984**, *4*, 7–40. [CrossRef]
14. Ali, M. Some Modified Stochastic Global Optimization Algorithms with Applications. Ph.D. Thesis, Loughborough University, Loughborough, UK, 1994.
15. Blum, C.; Roli, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.* **2003**, *35*, 268–308. [CrossRef]
16. Desale, S.; Rasool, A.; Andhale, S.; Rane, P. Heuristic and meta-heuristic algorithms and their relevance to the real world: A survey. *Int. J. Comput. Eng. Res. Trends* **2015**, *2*, 296–304.
17. Chakraborti, S.; Sanyal, S. An Elitist Simulated Annealing Algorithm for Solving Multi Objective Optimization Problems in Internet of Things Design. *Int. J. Adv. Netw. Appl.* **2015**, *7*, 2784.
18. Gonzales, G.V.; dos Santos, E.D.; Emmendorfer, L.R.; Isoldi, L.A.; Rocha, L.A.O.; Estrada, E.d.S.D. A Comparative Study of Simulated Annealing with different Cooling Schedules for Geometric Optimization of a Heat Transfer Problem According to Constructal Design. *Sci. Plena* **2015**, *11*. [CrossRef]
19. Poorjafari, V.; Yue, W.L.; Holyoak, N. A Comparison between Genetic Algorithms and Simulated Annealing for Minimizing Transfer Waiting Time in Transit Systems. *Int. J. Eng. Technol.* **2016**, *8*, 216. [CrossRef]
20. Armijo, L. Minimization of functions having lipschitz continuous first-partial derivatives. *Pac. J. Math.* **1966**, *16*, 187–192. [CrossRef]
21. Bertsekas, D.P. *Nonlinear Programming*; Athena Scientific Belmont: Belmont, MA, USA, 1999.
22. Dennis, J.E., Jr.; Schnabel, R.B. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 1996; Volume 16.
23. EL-Alem, M.; Aboutahoun, A.; Mahdi, S. Hybrid gradient simulated annealing algorithm for finding the global optimal of a nonlinear unconstrained optimization problem. *Soft Comput.* **2020**, *25*, 2325–2350. [CrossRef]
24. Ali, M.; Golalikhani, M.; Zhuang, J. A computational study on different penalty approaches for solving constrained global optimization problems with the electromagnetism-like method. *Optimization* **2014**, *63*, 403–419. [CrossRef]

25. Datta, R.; Deb, K. An adaptive normalization based constrained handling methodology with hybrid bi-objective and penalty function approach. In Proceedings of the 2012 IEEE Congress on Evolutionary Computation, Brisbane, QLD, Australia, 10–15 June 2012; pp. 1–8.
26. Deb, K. An efficient constraint handling method for genetic algorithms. *Comput. Methods Appl. Mech. Eng.* **2000**, *186*, 311–338. [CrossRef]
27. Jordehi, A.R. A review on constraint handling strategies in particle swarm optimization. *Neural Comput. Appl.* **2015**, *26*, 1265–1275. [CrossRef]
28. Joines, J.A.; Houck, C.R. On the Use of Non-Stationary Penalty Functions to Solve Nonlinear Constrained Optimization Problems with GA's. In Proceedings of the International Conference on Evolutionary Computation, Orlando, FL, USA, 27–29 June 1994; pp. 579–584.
29. Dekkers, A.; Aarts, E. Global Optimization and simulated-annealing algorithm. *Math. Program.* **1991**, *50*, 367–393. [CrossRef]
30. Ingber, L. Simulated Annealing: Practice versus Theory. *Mathl. Comput. Model.* **1993**, *18*, 29–57. [CrossRef]
31. Vidal, R. Applied Simulated Annealing (Lecture Notes in Economics and Mathematical Systems). In *Applied Simulated Annealing: Lecture Notes in Economics and Mathematical Systems*; Springer: Cham, Switzerland, 1993.
32. Kazarlis, S.; Petridis, V. Varying fitness functions in genetic algorithms: Studying the rate of increase of the dynamic penalty terms. In Proceedings of the International Conference on Parallel Problem Solving from Nature, Leiden, The Netherlands, 5–9 September 2020; Springer: Berlin/Heidelberg, Germany, 1998; pp. 211–220.
33. Michalewicz, Z.; Janikow, C.Z. Handling Constraints in Genetic Algorithms. In Proceedings of the 4th International Conference on Genetic Algorithms, San Diego, CA, USA, 13–16 July 1991; pp. 151–157.
34. Michalewicz, Z. A Survey of Constraint Handling Techniques in Evolutionary Computation Methods. *Evol. Program.* **1995**, *4*, 135–155.
35. Michalewicz, Z.; Schoenauer, M. Evolutionary algorithms for constrained parameter optimization problems. *Evol. Comput.* **1996**, *4*, 1–32. [CrossRef]
36. Homaifar, A.; Qi, C.X.; Lai, S.H. Constrained optimization via genetic algorithms. *Simulation* **1994**, *62*, 242–253. [CrossRef]
37. Parsopoulos, K.E.; Vrahatis, M.N. Particle swarm optimization method for constrained optimization problems. *Intell.-Technol.-Theory Appl. New Trends Intell. Technol.* **2002**, *76*, 214–220.
38. Petalas, Y.G.; Parsopoulos, K.E.; Vrahatis, M.N. Memetic particle swarm optimization. *Ann. Oper. Res.* **2007**, *156*, 99–127. [CrossRef]
39. El-Alem, M.; El-Sayed, S.; El-Sobky, B. Local convergence of the interior-point Newton method for general nonlinear programming. *J. Optim. Theory Appl.* **2004**, *120*, 487–502. [CrossRef]
40. Ali, M.M.; Gabere, M. A simulated annealing driven multi-start algorithm for bound constrained global optimization. *J. Comput. Appl. Math.* **2010**, *233*, 2661–2674. [CrossRef]
41. Kirkpatrick, S.; Gelatt, C.D.; Vecchi, M.P. Optimization by simulated annealing. *Science* **1983**, *220*, 671–680. [CrossRef]
42. Yarmohamadi, H.; Mirhosseini, S.H. A New Dynamic Simulated Annealing Algorithm for Global Optimization. *J. Math. Comput. Sci.* **2015**, *14*, 16–23. [CrossRef]
43. Corona, A.; Marchesi, M.; Martini, C.; Ridella, S. Minimizing multimodal functions of continuous variables with the simulated annealing algorithm. *ACM Trans. Math. Softw.* **1987**, *13*, 262–280. [CrossRef]
44. Metropolis, N.; Rosenbluth, A.W.; Rosenbluth, M.N.; Teller, A.H.; Teller, E. Equation of state calculations by fast computer machines. *J. Chem. Phys.* **1953**, *21*, 1087–1092. [CrossRef]
45. Liang, J.; Runarsson, T.P.; Mezura-Montes, E.; Clerc, M.; Suganthan, P.N.; Coello, C.C.; Deb, K. Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization. *J. Appl. Mech.* **2006**, *41*, 8–31.
46. Ma, H.; Simon, D. Blended biogeography-based optimization for constrained optimization. *Eng. Appl. Artif. Intell.* **2011**, *24*, 517–525. [CrossRef]
47. Brajevic, I. Crossover-based artificial bee colony algorithm for constrained optimization problems. *Neural Comput. Appl.* **2015**, *26*, 1587–1601. [CrossRef]
48. Ghasemishabankareh, B.; Li, X.; Ozlen, M. Cooperative coevolutionary differential evolution with improved augmented Lagrangian to solve constrained optimisation problems. *Inf. Sci.* **2016**, *369*, 441–456. [CrossRef]
49. Mohamed, A.W. A novel differential evolution algorithm for solving constrained engineering optimization problems. *J. Intell. Manuf.* **2018**, *29*, 659–692. [CrossRef]
50. Xu, B.; Tao, L.; Chen, X.; Cheng, W. Adaptive differential evolution with multi-population-based mutation operators for constrained optimization. *Soft Comput.* **2019**, *23*, 3423–3447. [CrossRef]
51. Sheskin, D.J. *Handbook of Parametric and Nonparametric Statistical Procedures*; CRC Press: Boca Raton, FL, USA, 2003.
52. Long, W.; Liang, X.; Cai, S.; Jiao, J.; Zhang, W. A modified augmented Lagrangian with improved grey wolf optimization to constrained optimization problems. *Neural Comput. Appl.* **2016**, *28*, 421–438. [CrossRef]
53. Lobato, F.S.; Steffen, V., Jr. Fish swarm optimization algorithm applied to engineering system design. *Lat. Am. J. Solids Struct.* **2014**, *11*, 143–156. [CrossRef]
54. Mazhoud, I.; Hadj-Hamou, K.; Bigeon, J.; Joyeux, P. Particle swarm optimization for solving engineering problems: A new constraint-handling mechanism. *Eng. Appl. Artif. Intell.* **2013**, *26*, 1263–1273. [CrossRef]

55. Mohamed, A.W.; Sabry, H.Z. Constrained optimization based on modified differential evolution algorithm. *Inf. Sci.* **2012**, *194*, 171–208. [CrossRef]
56. Rocha, A.M.A.; Fernandes, E.M.d.G. Self-Adaptive Penalties in the Electromagnetism-like Algorithm for Constrained Global Optimization Problems. In Proceedings of the 8th World Congress on Structural and Multidisciplinary Optimization, Lisbon, Portugal, 1–5 June 2009.
57. Yang, X.S.; Hossein Gandomi, A. Bat algorithm: A novel approach for global engineering optimization. *Eng. Comput.* **2012**, *29*, 464–483. [CrossRef]
58. Zhang, C.; Li, X.; Gao, L.; Wu, Q. An improved electromagnetism-like mechanism algorithm for constrained optimization. *Expert Syst. Appl.* **2013**, *40*, 5621–5634. [CrossRef]