


Article

A Hybrid Stochastic Deterministic Algorithm for Solving Unconstrained Optimization Problems

Ahmad M. Alshamrani ¹, Adel Fahad Alrasheedi ¹ , Khalid Abdulaziz Alnowibet ¹ , Salem Mahdi ^{2,3} 
and Ali Wagdy Mohamed ^{4,*}

¹ Statistics and Operations Research Department, College of Science, King Saud University, P.O. Box 2455, Riyadh 11451, Saudi Arabia

² Department of Mathematics & Computer Science, Faculty of Science, Alexandria University, Alexandria 21545, Egypt

³ Educational Research and Development Center Sanaa, Sanaa 00967, Yemen

⁴ Operations Research Department, Faculty of Graduate Studies for Statistical Research, Cairo University, Giza 12613, Egypt

* Correspondence: aliwagdy@staff.cu.edu.eg

Abstract: In this paper, a new deterministic method is proposed. This method depends on presenting (suggesting) some modifications to existing parameters of some conjugate gradient methods. The parameters of our suggested method contain a mix of deterministic and stochastic parameters. The proposed method is added to a line search algorithm to make it a globally convergent method. The convergence analysis of the method is established. The gradient vector is estimated by a finite difference approximation approach, and a new step-size h of this approach is generated randomly. In addition, a set of stochastic parameter formulas is constructed from which some solutions are generated randomly for an unconstrained problem. This stochastic technique is hybridized with the new deterministic method to obtain a new hybrid algorithm that finds an approximate solution for the global minimization problem. The performance of the suggested hybrid algorithm is tested in two sets of benchmark optimization test problems containing convex and non-convex functions. Comprehensive comparisons versus four other hybrid algorithms are listed in this study. The performance profiles are utilized to evaluate and compare the performance of the five hybrid algorithms. The numerical results show that our proposed hybrid algorithm is promising and competitive for finding the global optimum point. The comparison results between the performance of our suggested hybrid algorithm and the other four hybrid algorithms indicate that the proposed algorithm is competitive with, and in all cases superior to, the four algorithms in terms of the efficiency, reliability, and effectiveness for finding the global minimizers of non-convex functions.

Keywords: global optimization; unconstrained minimization; derivative-free optimization; meta-heuristics; stochastic methods; conjugate gradient methods; efficient algorithm; performance; software and testing

MSC: 65D05



Citation: Alshamrani, A.M.; Alrasheedi, A.F.; Alnowibet, K.A.; Mahdi, S.; Mohamed, A.W. A Hybrid Stochastic Deterministic Algorithm for Solving Unconstrained Optimization Problems. *Mathematics* **2022**, *10*, 3032. <https://doi.org/10.3390/math10173032>

Academic Editors: Yi-Kuei Lin, Lance Fiondella, Cheng-Fu Huang and Ping-Chen Chang

Received: 21 July 2022

Accepted: 17 August 2022

Published: 23 August 2022

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



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The main aim of this paper is to discuss the global minimization problem, which can be defined mathematically as follows.

Definition 1. Global minimum

The global minimum $x_{gl} \in \mathbb{S}$ of objective function f ; $f : \mathbb{S} \rightarrow \mathbb{R}$ is an input element with $f(x_{gl}) \leq f(x) \forall x \in \mathbb{S}$. If $S \subseteq \mathbb{R}^n$, it can be written as follows:

$$\min_{x \in \mathbb{S}} f(x) : \mathbb{S} \rightarrow \mathbb{R}. \quad (1)$$

In Problem (1), the $S \subseteq \mathbb{R}^n$ is the research domain of the function $f(x)$, and the following condition must be satisfied: $\|g(x_{gl})\| \leq \varepsilon$ for $\varepsilon > 0$ is small enough, where $\|g(x_{gl})\|$ is the norm of the gradient vector at the point x_{gl} .

We assume that $f(x) \in C^1$ (continuously differentiable).

Global optimization (GO) seeks to find the global minimum or maximum of an objective function of the optimization problem.

In the global optimization algorithms denoted by GOA, a problem is usually defined such as Formula (1), and in a case in which we seek to find the maximization of the real-valued function $f(x)$, we use the same algorithm with set $f(x) = -f(x)$.

Finding the global minimizer of Problem (1) can be difficult because our knowledge of f is usually only local.

The deterministic optimization methods desire to find only a local solution, a point at which the objective function is smaller than other feasible points in its neighborhood. Therefore, these methods are the speediest optimization algorithms in finding a local minimizer of a function.

Consequently, the GOA focuses on finding the minimum or maximum of the f over the given set, while the task of the local optimization algorithms, LOAs, seeks to find local minima or maxima. Hence, the task of the GOA is very difficult when the f is non-convex because the function f contains several local optimum points with one global optimal point.

Therefore, finding an arbitrary local minimum of the function f is relatively simple by using the LOA. On the other hand, finding the global minimizer of the function f is far more difficult, and the use of numerical solution algorithms often leads to a very difficult challenge.

Recently, (GOA) algorithms have been proposed that are designed to deal with the global minimization problem.

The ideas of these proposed GOAs depend on the principle of stochastic parameters. On the contrary, in the deterministic (classical) methods, no probabilistic information is used. Therefore, for finding the global minimizer of Problem (1) by using classical methods, it needs an exhaustive search over S .

Finding the global minimum for the unconstrained problems by using stochastic methods, the asymptotic convergence in probability can be proved, i.e., these methods are asymptotically successful with probability 1 [1–3].

The conjugate gradient method is one of the popular deterministic methods [4], which are widely used for finding a local unconstrained optimization problem [5]. The search direction of the conjugate gradient method is computed by

$$d_{k+1} = -g_{k+1} + \beta_k d_k, d_0 = -g_0, \tag{2}$$

where $g_{k+1} = g(x_{k+1})$ is the gradient vector, and the parameter β_k is the core difference between the different versions of the conjugate gradient methods which have been proposed to solve an unconstrained optimization problem (see, for example, [4,6,7]) or nonlinear equations (see, for example, [6,8–12]).

The history of the conjugate gradient method has been surveyed by many authors, and it began early on. See, for example, [13,14].

In the 1950s, the conjugate gradient methods were used to solve linear systems of equations and eigenproblems (nonlinear eigenvalue problem).

Since that date, the conjugate gradient method has attracted the attention of authors to propose many versions of conjugate gradient methods. Those proposed methods contain either one term, which represents the fundamentals of the conjugate gradient method, or two or three terms. For example, the conjugate gradient methods that have one term are

$$\beta_k^{HS} = \frac{y_k^T g_{k+1}}{d_k^T y_k}. \tag{3}$$

where $y_k = g_{k+1} - g_k$. Formula (3) was suggested by Hestenes and Stiefel [15]

$$\beta_k^{FR} = \frac{\|g_{k+1}\|^2}{\|g_k\|^2}. \tag{4}$$

Formula (4) was proposed by [16].

$$\beta_k^{PRP} = \frac{y_k^T g_{k+1}}{\|g_k\|^2}. \tag{5}$$

Formula (5) was presented by [17,18].

$$\beta_k^{LS} = \frac{F_{k+1}^T y_k}{-d_k^T g_k}. \tag{6}$$

Formula (6) was suggested by Liu and Storey [19].

$$\beta_k^{DY} = \frac{\|g_{k+1}\|^2}{y_k^T d_k}. \tag{7}$$

Formula (7) was suggested by Dai and Yuan [20].

Other formula versions of the parameter β_k in two and three terms can be found in [9,10,14,21–27].

The conjugate gradient methods have a numerical drawback: their sub-successive steps may be short if a small step is generated away from the optimal solution [26].

Therefore, to overcome this drawback, the line-search method is added to the CG method to make it a globally convergent method [28,29].

In previous studies, several conjugate gradient methods with the global convergence analysis of these methods are proposed (see, for example, [30–32]).

In practice, the CG method with a line-search method demonstrated superiority in obtaining the local minimizer of an unconstrained optimization problem.

A pure conjugate gradient line-search method is not capable of finding the global minimizer of the objective function of Problem (1) at each run, due to the fact that its stopping criterion guides it to a near local point of the previous point and other reasons that have been mentioned.

Hence the stochastic methods (random parameters) are used to guide the search process [33].

In practice, the numerical results showed that the hybridizing between deterministic algorithms and random algorithms is very successful. See, for example, [34–38].

The main contributions of this paper are presented in the following points.

- Some modifications and improvements to some previous conjugate gradient methods are proposed. The result is a new local optimization algorithm, the suggested algorithm contains a mix of deterministic parameters and random parameters.
- The proposed conjugate gradient method is combined with a line-search algorithm to obtain a globally convergent method.
- The convergence analysis of the suggested algorithm is established.
- The approximate gradient vector is computed by using a numerical approximation method, and it is provided with a new approach for estimating an appropriate step-size h randomly.
- Three stochastic parameter formulas are constructed, from which some solutions are generated randomly for an unconstrained optimization problem.
- The proposed conjugate gradient algorithm is hybridized with the stochastic technique; the result is a hybrid algorithm that solves Problem (1).

- Numerical experiments are presented by solving a set of test problems containing convex and non-convex functions.

The rest of paper is organized as follows. Section 2 briefly discusses finding a local minimizer by conjugate gradient methods. Section 3 presents a new modified conjugate gradient method and the convergence analysis of the proposed method. Section 4 presents a numerical approximation method for estimating the gradient vector. Section 5 briefly discusses the global minimization problem. Section 6 gives the numerical experiments of the global minimization problem. Section 7 contains some concluding remarks.

2. Finding a Local Minimizer

A high number of previous studies have been presented for finding a local minimizer x^* of the function $f(x)$. See, for example, [39–45]. One of the efficient and inexpensive methods is the conjugate gradient method (CG). In the following subsection, the conjugate gradient method (CG) is briefly discussed.

Conjugate Gradient Methods (CG)

The conjugate gradient method (CG) is a popular deterministic method that concerns finding a local minimizer of an objective function for an unconstrained optimization problem.

A point x^* is a local minimizer if there is a neighborhood N of x^* , such that $f(x^*) \leq f(x)$ for $x \in N$ and $\|g(x^*)\| < \varepsilon$.

The conjugate gradient method (CG) is iterative. At each iteration, a step in the direction d_k with a step-size α_k is computed and added to the current point as follows:

$$x_{k+1} = x_k + \alpha_k d_k, \tag{8}$$

where the step-size α_k is positive, and the d_k is a research direction that defined by (2). Conjugate gradient (CG) methods contain a group of optimization algorithms that solve optimization problems; additionally, conjugate gradient (CG) methods have some features, such as low memory requirements and strong local and global convergence properties [46].

If α_k is obtained by an exact line search, then by (2) and the orthogonality condition $g_{k+1}^T y_k = 0$, we have $y_k^T d_k = (g_{k+1} - g_k)^T d_k = -g_k^T d_k = \|g_k\|^2$.

Hence, $\beta_k^{HS} = \beta_k^{PRP}$ when α_k is computed by an exact line search.

Therefore, if the objective function f is quadratic and α_k is picked to minimize the objective function in the search direction d_k , the above (3)–(7) choices of the parameter β_k are equivalent, but for a general nonlinear function, different choices have quite different convergence properties [4].

For non-quadratic cost functions, each choice for the update parameter leads to different performance [14].

Under the various line search strategies, the CG methods satisfy the following sufficient descent condition:

$$g_k^T d_k \leq -C \|g_k\|^2, \tag{9}$$

where $C > 0$ is a constant. Furthermore, the sufficient descent condition plays a crucial role in the global convergence analysis of the CG methods. See, for example, [13,14,20,26,42,46–49].

Recently, Yuan and Zhang [25] and Yuan et al. [26] showed that the CG method has strong convergence advantages if it satisfies the trust-region merit, which is defined by

$$\|d_k\| < C_v \|g_k\|, \tag{10}$$

where $C_v > 0$ is a constant. It is shown, therefore, that the trust-region feature can enable the search direction d_k being limited in trust radius [26]. Moreover, many authors have suggested that other CG methods have good numerical performance and strong convergence features. See, for example, [25,26,46,48].

The choice of a suitable step-size α_k can enable these methods to succeed in global convergence. The exact line search is defined by

$$f(x_k + \alpha_k d_k) = \min_{\alpha \geq 0} \theta(\alpha) = f(x_k + \alpha d_k). \tag{11}$$

However, the exact line search is very impossible in large-scale problems. Hence, there are many inexact line-search methods to achieve this purpose. Therefore, the weak Wolfe–Powell method (WWP) is a popular method, and it is widely used as an inexact line-search technique. The WWP method is designed to find the step-size α_k satisfying the following conditions:

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \delta \alpha_k g_k^T d_k, \tag{12}$$

and

$$g(x_k + \alpha_k d_k)^T d_k \geq \sigma g_k^T d_k, \tag{13}$$

where $\delta \in (0, 0.5)$ and $\sigma \in (\delta, 1)$ are constants.

The first condition is called the Armijo condition, and the WWP line search reduces to the strong Wolfe–Powell (SWP) by substituting the second condition with the following inequality:

$$|g(x_k + \alpha_k d_k)^T d_k| \leq -\sigma g_k^T d_k, \tag{14}$$

Under the WWP line search, we usually assume that the gradient $g(x)$ of the function $f(x)$ is Lipschitz continuous in the convergence analysis. This means that the following is met:

$$\|g(x) - g(y)\| \leq L \|x - y\|, \tag{15}$$

where L is a constant for $x, y \in R^n$.

Recently, Yuan et al. [26] proposed a new CG method based on the work presented by Hager and Zhang [46]. The proposed method that was suggested (MHZ) by Yuan et al. [26] has the sufficient descent property and the trust region feature independent of line-search technique.

3. Suggested CG Method

The idea of this paper begins with the idea which is presented by Hager and Zhang [46]. They suggested a new CG method, and its parameter β_k^{HZ} is defined by

$$\beta_k^{HZ} = \frac{(y_k^T g_k)(d_{k-1}^T y_k) - 2\|y_k\|^2(d_{k-1}^T g_k)}{(d_{k-1}^T y_k)^2}. \tag{16}$$

The parameter β_k^{HZ} can ensure that d_k satisfies $d_k^T g_k \leq -\frac{7}{8}\|g_k\|^2$ [46]. If the step-size α_k is computed by the exact line search, then β_k^{HZ} reduces to β_k^{HS} , due to $d_k^T g_k = 0$ being true [26].

Hence, for obtaining the global convergence for a general function, the authors of [46] dynamically adjust the lower bound of β_k^{HZ} by the following formula:

$$d_k = -g_k + \beta_k^{HZ+} d_{k-1}, d_0 = -g_0, \tag{17}$$

where $\beta_k^{HZ+} = \max\{\beta_k^{HZ}, r_k\}$, $r_k = \frac{-1}{\|d_{k-1}\| \min\{r, \|g_{k-1}\|\}}$, $r > 0$ is a constant. In the numerical experiments, they set $r = 0.01$.

Several authors have proposed some modifications and improvements to the HZ-CG method to obtain modified and improved algorithms which solve large unconstrained optimization problems. The modern version of these HZ-CG methods is presented by Yuan et al. [26]. They have extended and modified the CG-HZ method to the CG-MHZ

method. The CG-MHZ method has a sufficient condition and the trust region merit. The parameters of the CG-MHZ method are defined as follows:

$$d_k = -g_k + \beta_k^{MHZ}d_{k-1}, d_0 = -g_0, \tag{18}$$

where

$$\beta_k^{MHZ} = \frac{(y_k^T g_k)(d_{k-1}^T y_k) - 2\|y_k\|^2(d_{k-1}^T g_k)}{\max\{\eta\|y_k\|^2\|d_{k-1}\|^2, (d_{k-1}^T y_k)^2\}}, \tag{19}$$

where $\eta > 0.5$ is a constant [26].

The denominator $(d_{k-1}^T y_k)^2$ in β_k^{MHZ} has been modified to $\max\{r\|y_k\|^2\|d_k\|^2, (d_{k-1}^T y_k)^2\}$ in β_k^{MHZ} . This action can make the search direction d_k stay in a trust region automatically below every iteration [26]. In addition, in the case of $r\|y_k\|^2\|d_{k-1}\|^2 < (d_{k-1}^T y_k)^2$, β_k^{MHZ} reduces to β_k^{HZ} with α_k being computed to satisfy the inexact line search. Moreover, β_k^{MHZ} reduces to β_k^{HS} under the exact line search.

Therefore, we extend and modify the MHZ method to obtain a new CG method that has a sufficient condition and the trust region feature.

The proposed method is abbreviated by MXHZ to indicate to a mix of parameters (deterministic and random). It is defined as follows:

$$d_k = -g_k + \beta_k^{MXHZ}d_{k-1}, d_0 = -g_0, \tag{20}$$

$$\beta_k^{MXHZ} = \frac{(y_k^T g_k)(d_{k-1}^T y_k) - 2\|y_k\|^2(d_{k-1}^T g_k)}{\max\{mix_k\|y_k\|^2\|d_{k-1}\|^2, (d_{k-1}^T y_k)^2\}}, \tag{21}$$

where mix_k is a mixed parameter: it is defined as either a deterministic or random parameter.

$$mix_k = \begin{cases} \Delta f \Delta x & \text{if } \Delta f \Delta x \geq \frac{9}{10}, \\ R_p & \text{otherwise.} \end{cases} \tag{22}$$

$$\tag{23}$$

Δf and $\Delta f x$ are defined as follows:

$$\Delta f = |f_0 - f_M|, \tag{24}$$

where M is the inner loop maximum number of iterations, and after an M inner loop of iterations, f_M and Δf are computed, and then we update $f_0 = f_M$, while Δx is defined as follows:

$$\Delta x = \|x_{k+1} - x_k\|, \text{ for } k = 0, 1, \dots, M. \tag{25}$$

The parameter R_p is generated randomly from $[\frac{9}{10}, \infty)$; in this case, the search direction d_k stays in this method longer, and then the MXHZ method can reduce to the HZ method if $mix_k\|d_{k+1}\|^2\|y_k\|^2 < (d_{k-1}^T y_k)^2$, and if $mix_k = \eta$, then the MXHZ method can reduce to the MHZ method. Therefore, the MXHZ method inherits the features of the MHZ and HZ methods.

Note: The mixed parameters are working in a dynamic way; in other words, they take turns working in a dynamic way, i.e., if the deterministic parameter vanishes, the stochastic parameter will run.

This procedure with the proposed stochastic technique makes the hybrid proposed algorithm escape from the local points while researching the global minimizer of the non-convex function, where the hybrid proposed algorithm is a combination of two techniques, the first one being the new CG approach and the second one being a proposed stochastic technique. Further description of both algorithms can be found in the next sections.

According to the above discussions, Algorithm 1 is designed as follows:

Algorithm 1 A new modified CG method

Input: $f : \mathbb{R}^n \rightarrow \mathbb{R}, f \in C^1, \gamma \in (0, 1), k = 0$, a starting point $x_k \in \mathbb{R}^n$ and $\varepsilon > 0$.

Output: $x^* = x_{loc}$ the local minimizer of $f, f(x^*)$, the value of f at x^*

- 1: Set $d_0 = -g_0$ and $k := 0$.
 - 2: **while** $\|g_k\| > \varepsilon$ **do** $\triangleright g_{ac}$ is the value of the gradient vector at the accepted point x_{ac} .
 - 3: pick α_k satisfying the (WWP) line search conditions (12) and (13).
 - 4: Generate a new point $x_{k+1} = x_k + \alpha_k d_k$. \triangleright the step-size α_k is computed by (47).
 - 5: compute $f_k = f(x_{k+1}), g_k = g(x_{k+1})$.
 - 6: Set $k = k + 1$.
 - 7: calculate the search direction d_k by (20).
 - 8: **end while**
 - 9: **return** x_{ac} the local minimizer and its function value f_{ac}
-

3.1. Convergence Analysis of Algorithm 1

This section gives some properties and convergence analysis of Algorithm 1. In the following, we show that the search direction d_k , which is generated by (20), satisfies the sufficient descent condition (9) and the trust region feature (10).

Two reasonable assumptions are considered as follows.

Assumption 1. Assume that the objective function $f(x)$ is continuously differentiable.

Assumption 2. In some neighborhood N of the level set

$$\ell = \{x \in \mathbb{R}^n : f(x) \leq f(x_0)\},$$

the gradient vector $g(x)$ is Lipschitz continuous. That is, there exists a constant $L < \infty$, such that

$$\|g(x) - g(y)\| \leq L\|x - y\|,$$

for all $x, y \in N$.

Lemma 1. Assume that the sequence $\{x_k\}$ is generated by Algorithm 1. If $d_k^T y_k \neq 0$, we have

$$g_k^T d_k \leq -c\|g_k\|^2, \tag{26}$$

and

$$\|d_k\| \leq r_v \|g_k\|, \tag{27}$$

where $c = 1 - \frac{7}{8\text{mix}_k} > 0, \text{mix}_k \geq \frac{9}{10}$, and $r_v = (1 + \frac{6}{\text{mix}_k})$ is the trust-region radius.

Proof. If $k = 0, d_0 = -g_0$, then $g_0^T d_0 = -\|g_0\|^2$ and $\|d_0\| = \|g_0\|$, which implies that (26) and (27) by picking $c \in (0, 1]$ and $r_v \in [1, \infty)$.

Combining the (20) with (21), we obtain

$$g_k^T d_k = \frac{(y_k^T g_k)(d_{k-1}^T y_k)(g_k^T d_{k-1}) - 2\|y_k\|^2(g_k^T d_{k-1})^2}{\max\{\text{mix}_k \|y_k\|^2 \|d_{k-1}\|^2, (d_{k-1}^T y_k)^2\}} - \|g_k\|^2. \tag{28}$$

We apply the inequality $u^T v \leq \frac{1}{2}(\|u\|^2 + \|v\|^2)$ to the first term of the numerator of (28) with $u = d_{k-1} g_k^T y_k, v = y_k g_k^T d_{k-1}$; it is clear that $u^T v \leq \frac{7}{8}(\|u\|^2 + \|v\|^2)$ is true.

Hence, we have

$$g_k^T d_k = \frac{(y_k^T g_k)(d_{k-1}^T y_k)(g_k^T d_{k-1}) - 2\|y_k\|^2(g_k^T d_{k-1})^2}{\max\{\text{mix}_k \|y_k\|^2 \|d_{k-1}\|^2, (d_{k-1}^T y_k)^2\}} - \|g_k\|^2 \leq$$

$$\begin{aligned}
 & -\|g_k\|^2 + \frac{\frac{7}{8}\|y_k\|^2\|g_k\|^2\|d_{k-1}\|^2 + \frac{7}{8}\|y_k\|^2(g_k^T d_{k-1})^2 - 2\|y_k\|^2(g_k^T d_{k-1})^2}{\max\{mix_k\|y_k\|^2\|d_{k-1}\|^2, (d_{k-1}^T y_k)^2\}} = \\
 & -\|g_k\|^2 + \frac{\frac{7}{8}\|y_k\|^2\|g_k\|^2\|d_{k-1}\|^2 - \frac{9}{8}\|y_k\|^2(g_k^T d_{k-1})^2}{\max\{mix_k\|y_k\|^2\|d_{k-1}\|^2, (d_{k-1}^T y_k)^2\}} \leq \\
 & -\|g_k\|^2 + \frac{\frac{7}{8}\|y_k\|^2\|g_k\|^2\|d_{k-1}\|^2}{\max\{mix_k\|y_k\|^2\|d_{k-1}\|^2, (d_{k-1}^T y_k)^2\}} \leq \left(\frac{7}{8mix_k} - 1\right)\|g_k\|^2,
 \end{aligned}$$

such that

$$\max\{mix_k\|y_k\|^2\|d_{k-1}\|^2, (d_{k-1}^T y_k)^2\} \geq mix_k\|y_k\|^2\|d_{k-1}\|^2. \tag{29}$$

Since, $mix_k \geq \frac{9}{10}$ and $c = 1 - \frac{7}{8mix_k} > 0$, (26) is true.

By using (29), it is clear that

$$\begin{aligned}
 \|d_k\| &= \left\| -g_k + \frac{(y_k^T g_k)(d_{k-1}^T y_k) - 2\|y_k\|^2(d_{k-1}^T g_k)}{\max\{mix_k\|y_k\|^2\|d_{k-1}\|^2, (d_{k-1}^T y_k)^2\}} d_{k-1} \right\| \leq \\
 & \| -g_k \| + \frac{4\|y_k\|^2\|g_k\|\|d_{k-1}\|^2 + 2\|y_k\|^2\|g_k\|\|d_{k-1}\|^2}{mix_k\|y_k\|^2\|d_{k-1}\|^2} = \left(1 + \frac{6}{mix_k}\right)\|g_k\|
 \end{aligned}$$

Thus, (27) is true with $r_v \in [1 + \frac{6}{mix_k}, \infty)$. The proof is complete. \square

Corollary 1. *Following on from Inequality (27) Lemma 1, the following inequality is true.*

$$\sum_{k=0}^{\infty} \frac{\|g_k\|^4}{\|d_k\|^2} = \infty. \tag{30}$$

Proof. Since $\|d_k\| \leq r_v \|g_k\|^2$, where $1 < r_v < \infty$, then $\|d_k\|^2 \leq r_v^2 \|g_k\|^4$, therefore, $\frac{\|d_k\|^2}{\|g_k\|^4} \leq r_v^2$, hence $\frac{\|g_k\|^4}{\|d_k\|^2} \geq \frac{1}{r_v^2}$, by using summing on the final expression as $k \rightarrow \infty$, we obtain $\sum_{k=0}^{\infty} \frac{\|g_k\|^4}{\|d_k\|^2} \geq \sum_{k=0}^{\infty} \frac{1}{r_v^2} = \frac{1}{r_v^2} \sum_{k=0}^{\infty} 1 = \infty$, then (30) is true. \square

Under the assumptions, we give a useful lemma, which was essentially proved by Zoutendijk [50] and Wolfe [51,52].

Lemma 2. *Suppose that x_0 is a starting point for which Assumption 1 is met. Consider any method of the form (20), where d_k is a descent direction and α_k satisfies the standard Wolfe conditions (12) and (13). Then, we have that*

$$\sum_{k=0}^{\infty} \frac{(g_k^T d_k)^2}{\|d_k\|^2} < \infty \tag{31}$$

Proof. It follows from (13) that

$$d_k^T y_k = d_k^T (g_{k+1} - g_k) \geq (\sigma - 1)g_k^T d_k. \tag{32}$$

On the other hand, the Lipschitz condition (15) implies

$$(g_{k+1} - g_k)^T d_k \leq \alpha_k L \|d_k\|^2. \tag{33}$$

The above two inequalities give

$$\alpha_k \geq \frac{\sigma - 1}{L} \cdot \frac{g_k^T d_k}{\|d_k\|^2}, \tag{34}$$

which with (12) implies that

$$f_k - f_{k+1} \geq c \frac{(g_k^T d_k)^2}{\|d_k\|^2}, \tag{35}$$

where $c = \frac{\delta(1-\sigma)}{L}$. By summing (35) and noting that f is bounded below, we see that (31) holds, which concludes the proof. \square

Theorem 1. Assume Assumptions 1 and 2 hold, and by using the result of Corollary 1, then the sequence $\{g_k\}$ that is generated by Algorithm 1 satisfies the following.

$$\liminf_{k \rightarrow \infty} \|g_k\| = 0, \tag{36}$$

Proof. By contradiction, suppose that (36) is not true; then, for some $\epsilon > 0$, the following inequality is true:

$$\|g_k\| \geq \epsilon. \tag{37}$$

Hence, considering inequality (37) with (26), we have

$$g_k^T d_k \leq -c \|g_k\|^2 \leq -\epsilon^2. \tag{38}$$

Then, we have

$$\begin{aligned} \frac{g_k^T d_k}{\|d_k\|} &\leq \frac{-\epsilon^2}{\|d_k\|}; \\ \frac{g_k^T d_k}{\|d_k\|} &\geq \frac{\epsilon^4}{\|d_k\|^2}, \end{aligned}$$

and by summing the final expression, we obtain

$$\sum_{k=0}^{\infty} \frac{(g_k^T d_k)^2}{\|d_k\|^2} \geq \sum_{k=0}^{\infty} \frac{\epsilon^4}{\|d_k\|^2} = \infty. \tag{39}$$

The above leads to a contradiction with (31). Thus, (36) holds. \square

Remark 1. The search direction (20) satisfies the sufficient descent condition (9).

Remark 2. Lemma 1 insures that Algorithm 1 possesses the sufficient descent property and the trust region feature automatically.

Remark 3. Theorem 1 proves that the sequence $\{g_k\}$, which is generated by Algorithm 1, seeks to zero as long as $k \rightarrow \infty$.

In the following section, we use numerical differentiation to compute the approximate values of the step-size α_k and gradient vector g_k .

4. Numerical Approximation of the Gradient

In this paper, we only need to estimate the values of α_k and g_k , which is used to compute the search direction d_k that is defined by (20). Hence, the exact derivative is not considered in this paper.

A finite difference approximation to the gradient is a common approach that is used for minimizing a continuous differentiable nonlinear function. A powerful approach for derivative-free optimization is to use finite differences. Therefore, many authors have proposed several methods for the numerical approximation of the gradient, and they presented good results for estimating the gradient vector (see, for example, [53–56]). The finite difference approximations of the first derivative values of the function at different points in the neighborhood of the point $x = x_i$ are used for estimating the slope [57].

Therefore, the exact first derivative can be replaced by the following formula to obtain an approximation to the gradient vector.

$$D_f f(x_i) = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} = \frac{f(x_i + h) - f(x_i)}{h}, \tag{40}$$

where h is finite and small but not necessarily infinitesimally small.

Rationally, the approximation will improve if h is made smaller, i.e., the error will be smaller as long as the value of h is made smaller.

There are many formulas used to approximate the first derivative (gradient) at the point $x = x_i$.

The common approaches are the forward difference and the central difference [58–62]. These formulas can be derived from the Taylor series. Finding the approximate of the first derivative at the point x_i means that the derivative is estimated as the value of the slope of the line that connects the two, three, four, or five points.

For improving the difference approximation of the derivatives, the three, four, and five points can be used to derive these approaches, but they will be more expensive than using two points, so in this paper, we use two points to estimate the gradient vector.

The numerical computation of the first derivative $f'(x)$ of a given function $f(x)$ by the central difference approach is known to involve aspects of both accuracy and precision [63], but it needs $2n$ function emulations at each iteration against n function emulations at each iteration for the forward-difference approximation approach.

Thus, in this paper we use the forward-difference approximation approach that was defined by (40), because it is an inexpensive approach and it has reasonable accuracy [55,58].

The success of all finite difference approximation methods depends on selecting the fit step-size h (interval).

In the following section, we discuss the error approximation of the first derivative, which guides us to present a finite-difference interval for the forward-difference approximation scheme that balances the truncation error, which arises from the error in the Taylor series approximation, and the measurement error, which results from noise in the function evaluation [55].

4.1. Error Analysis of Finite-Difference Approximation Vector

In the above section, the forward-difference approximation formula for the first derivative is defined. Due to its errors being proportional to some power of the step-size h , it seems that the errors continue to decrease if h is minimized. However, this is only a part of the problem since it be considered only the truncation error caused by truncating the high-order terms in the Taylor series expansion and does not take into account the round-off error caused by quantization. Therefore, this section discusses the round-off error alongside the truncation error. For this purpose, assume that the function values $f(x)$, $f(x + h)$, are quantized (rounded-off) to $\theta_1 = f(x + h) + \epsilon_1$, $\theta_0 = f(x) + \epsilon_0$, where the magnitudes of the round-off errors ϵ_1 and ϵ_0 are all smaller than some positive number ϵ , that is, $|\epsilon_i| \leq \epsilon$, with $i = 0, 1$.

Therefore, the total error of the forward difference approximation defined by (40) is derived by

$$D_f f(x) = \frac{\theta_1 - \theta_0}{h} = \frac{f(x + h) + \epsilon_1 - f(x) - \epsilon_0}{h} = f'(x) + \frac{\epsilon_1 - \epsilon_0}{h} + \frac{\psi}{2}h. \tag{41}$$

Then,

$$|D_f f(x) - f'(x)| \leq \left| \frac{\epsilon_1 - \epsilon_0}{h} \right| + \left| \frac{\psi}{2} \right| h \leq \frac{2\epsilon}{h} + \frac{|\psi|}{2}h, \tag{42}$$

where $\psi = f''(x)$. Therefore, the upper bound of error is represented by the right-hand side of Inequality (42). The upper bound of error contains two terms; the first one is due to the round-off error and is in inverse proportion to the step-size h , while the second

one is due to the truncation error and is in direct proportion to h . These two parts can be formulated as a function $\phi(h)$ with respect to h as follows: $\phi(h) = \frac{2\varepsilon}{h} + \frac{|\psi|}{2}h$. Now, if we find the minimizer h^* of the function $\phi(h)$, then the value $\phi(h^*)$ is the upper bound of the total error. Hence, $\frac{d\phi(h)}{dh} = \frac{-2\varepsilon}{h^2} + \frac{|\psi|}{2} = 0$. Then,

$$h^* = 2\sqrt{\frac{\varepsilon}{|\psi|}} = 2\sqrt{\frac{\varepsilon}{|f''(x)|}}. \tag{43}$$

Hence, it can be deduced that as we make the step-size h smaller, the round-off error may increase, while the truncation error decreases. This is named the “step-size dilemma”.

Therefore, there must be some optimal step-size h^* for the forward difference approximation formula, as derived analytically in (43). However, Formula (43) is only of theoretical value and cannot be used practically to determine h^* because we do not have any information about the second derivative, and therefore, we cannot estimate the values of ψ . Moreover, that the step-size h^* minimizes not the real error but its upper bound, we can never expect the true optimal step-size h^* to be uniform for all x with Formula (40).

Therefore, there are many approaches which have been presented to deal with the step-size dilemma.

Recently, Shi et al. [55] proposed a bisection search for finding a finite-difference interval for a finite-difference method. Their approach is presented to balance the truncation error, which arises from the error in the Taylor series approximation, and the measurement error, which results from noise in the function evaluation. According to their numerical experience, the finite-difference interval h^* is bounded between the ranges $[2.00 \times 10^{-4}, 6.32 \times 10^{-1}]$, $[2.72 \times 10^{-4}, 8.26 \times 10^0]$ and $[3.94 \times 10^0, 8.44 \times 10^{-3}]$ by using the forward and central differences to estimate the values of the first derivative of the function $f(x)$.

Additionally, Berahas et al. [58] presented a theoretical and experimental comparison of gradient approximations in derivative-free optimization. They analyzed several methods for approximating gradients of noisy functions using only function values. Those methods contain finite differences with other methods. The values of the finite difference interval is as follows: $10^{-8} \leq h^* \leq 1$.

According to the previous studies, the essence of the difference between all proposed approaches is to define the step-size of h ; therefore, the value of the step-size is ranging between this range: $h^* \in [1, 12 \times 10^{-10}]$.

Therefore, the step-size in this paper is generated in a random way. Furthermore, the values of h are related to the value of f to cover this range, but the advantage here is that the value of h is changed at each iteration randomly.

The following section discusses a new approach to define the step-size h^* .

4.2. Selecting a Step-Size for a Forward Difference Method

The forward difference method is an inexpensive method versus the other different methods. Furthermore, it has shown good results for minimizing noisy black-box functions [55].

Therefore, the forward difference method defined by (40) is used to estimate the gradient value.

According to the assumptions which are mentioned in Section 3.1, for any starting point x_0 , the objective function f satisfies the following: $f_0 \geq f_1 \geq \dots \geq f_k$, for $k = 0, 1, 2, \dots$. Moreover, the numerical results which were presented in the previous studies denote that the values of the finite difference interval are $10^{-10} \leq h^* \leq 1$.

Consequently, in this section, we design an approach that guarantees that the values of the step-size h^* are inside the interval $[0.1, 10^{-8}]$. This approach is described in the following algorithm.

Algorithm for selecting the fit values of the step-size h^*

Step 1: Generated a set random values between 10^{-2} and 10^{-7} at each iteration k , the set values are $L_\epsilon = \{l_{\epsilon_1}, l_{\epsilon_2}, \dots, l_{\epsilon_{10}}\}$.

Step 2: Extract the minimum and maximum values in the set L_ϵ ; $M_\epsilon = \min\{l_{\epsilon_i}; i=1,2,\dots,10\}$, $N_\epsilon = \max\{l_{\epsilon_i}; i=1,2,\dots,10\}$ and set $M_f = M_\epsilon^{-1}$.

Step 3: The values of the function f are computed at every iteration k ; $f_k = f(x_k)$, and then we distinguish two cases of the values of $|f_k|$ as follows.

Case 1: If $|f_k| \in [10^{-2}, \infty)$, then step-size h is computed by

$$h_k = \begin{cases} N_\epsilon \cdot M_\epsilon & \text{if } |f_k| > M_f, \\ M_\epsilon \cdot \frac{1}{|f_k|} & \text{otherwise.} \end{cases} \tag{44}$$

Case 2: If $|f_k| \in [0, 10^{-2})$, then step-size h_k is computed by generated a random number form $[10^{-4}, 10^{-7}]$.

4.3. Gradient Approximation by Forward Differences

The forward finite difference (DFF) used to estimate the value of the gradient vector of f at $x \in \mathbb{R}^n$ is computed using the sample set $X = \{x + he_i\}_{i=1}^n \cup \{x\}$, where $h > 0$ is the finite difference interval defined by Algorithm 2, and $e_i \in \mathbb{R}^n$ is the i th column of the identity matrix, as follows:

$$[DFF]_i = \frac{f(x + he_i) - f(x)}{h}, \text{ for } i = 1, 2, \dots, n. \tag{45}$$

Now, we have $g(x) \approx DFF(x)$ as an approximation to the gradient of f at x ; thus, the step-size α_k is computed as follows.

At each iteration k , the function $f(x)$ is approximated by using Taylor’s expansion up to the linear term around the point x_k to obtain

$$f(x_k + p) \approx f(x_k) + g(x_k)^T p.$$

The existence of the second derivative of $f(x)$ is not necessary in our case. We define the quadratic model of $f(x)$ at x_k as

$$m_k(p) = \frac{1}{2} (f(x_k) + g(x_k)^T p)^2 = \frac{1}{2} f(x_k)^2 + f(x_k)g(x_k)^T p + \frac{1}{2} p^T g(x_k)g(x_k)^T p.$$

Now, we set $p = -\alpha g(x_k)$, where α is a step length along the negative gradient direction. We find the value of α by solving the following subproblem:

$$\min_{\alpha \in \mathbb{R}} m_k(\alpha) = \frac{1}{2} f(x_k)^2 - \alpha f(x_k)g(x_k)^T g(x_k) + \frac{1}{2} \alpha^2 (g(x_k)^T g(x_k))^2.$$

This gives

$$\alpha_k = \frac{f(x_k)}{\|g(x_k)\|^2}. \tag{46}$$

We set $g(x_k) \approx DFF(x_k)$

$$\alpha_k = \frac{f(x_k)}{\|DFF(x_k)\|^2}. \tag{47}$$

The issue of finding a local minimizer of an objective function, an optimization problem, and the convergence analysis of the method MXHZ are discussed by the above sections. Consequently, Algorithm 1 is capable of finding the local minimizer of convex or non-convex functions. Therefore, in the following section, the global minimization problem is discussed and stochastic technique is presented.

Algorithm 2 Hybrid Stochastic Parameters Conjugate Gradient Algorithm

Input: $f : \mathbb{R}^n \rightarrow \mathbb{R}, f \in C^1, f_{cg}$ obtained by Algorithm 1 and $\varepsilon > 0$.

Output: $x_{gl} = x_{ac}$ the global minimizer of $f, f(x_{gl})$, the value of f at x_{gl} .

- 1: **while** $|f_{ac} - f^*| > \varepsilon$ or FEs $< n10^4$ **do**
 - 2: f_{cg} is a new value of function f generated by Algorithm 1.
 - 3: $f_{ac} = \min\{f_{cg}, f_1, f_2\}$ and x_{ac} the point that gives the f_{ac} .
 - 4: **if** $|f_{ac} - f^*| \leq \varepsilon$ **then**
 - 5: Stop.
 - 6: **end if**
 - 7: **if** $\Delta f == 0$ **then**
 - 8: compute the point x_3 and the function value $f_3 = f(x_3)$ by (51).
 - 9: **if** $f_3 < f_{ac}$ **then**
 - 10: accept the point x_3 as the best point and set $x_{ac} \rightarrow x_3$ and $f_{ac} \rightarrow f_3$, and go to Line 1.
 - 11: **else**
 - 12: generate another point x_3 by (51).
 - 13: **end if**
 - 14: otherwise go to Line 1.
 - 15: **end if**
 - 16: **end while**
 - 17: **return** x_{ac} the best point and its function value f_{ac}
-

5. Global Minimization Problem

The theoretical results of Section 3 indicate that Algorithm 1 stops when $\|g(x_k)\| \leq \varepsilon$ is met.

This means that a pure conjugate gradient method is not capable of finding the global minimizer of a non-convex function at each run due to it having many local points.

Therefore, to make the CG method able to find the global minimizer of a non-convex function at each run, some stochastic parameters are presented, and they are hybridized with a conjugate gradient method to obtain a new hybrid algorithm that is capable of escaping from a local point. In the previous literature, the numerical results demonstrated that the hybridizing between deterministic algorithms and stochastic algorithms is very successful for overcoming the drawback of the deterministic methods in falling and sticking into a local point. See, for example, [34–38].

Consequently, in the following subsection, three of the stochastic parameter formulas are constructed and hybridized with each CG method of the set—CG methods MXHZ, MHZ, HZ, HS, and FR—to obtain five algorithms which try to solve Problem (1). These algorithms are abbreviated by HSTMXHZ, HSTMHZ, HSTHZ, HSTHS, and HSTFR, respectively.

5.1. Stochastic Technique for Unconstrained Global Optimization

A new stochastic technique (parameters) is used to make the CG method find the global optimal solution for a test problem which contains a non-convex function by helping it to escape from the local points. These stochastic parameters are described by the following algorithm.

Three different points are randomly generated by the following steps (Algorithm ST).

Stochastic Parameters Algorithm (ST)

Step 1: The first point is defined by the following. This point is a mixed point (deterministic and stochastic).

$$x_1 = x_{ac} + \eta_k \Psi_k, \tag{48}$$

where $\Psi_k = \alpha_k d_k$; α_k is the approximate value of the step-size along the direction d_k , and it is computed by (47); $\eta_k \in (0, 2)$ is a random number; and d_k is a direction search and defined by (20). The value of the function is computed at the point x_1 ; $f_1 = f(x_1)$.

Step 2: The second point is computed by the following. Generate $\chi_k \sim [-0.95, 1]^n$ as a random vector, set $\gamma_k = 10^{\psi_k}$, $\psi_k \in [0.01, 1]$, and compute $\sigma_k = \frac{(1+\gamma_k)^{|\chi_i|} - 0.5}{\gamma_k} S\chi_i$, where the value of ψ_k is selected randomly from the interval $[0.01, 1]$ at each iteration k , $i = 1, 2, \dots, n$, n is a number of variables, and $S\chi_i$ represents the signs of the vector χ_i defined by

$$S\chi_i = \begin{cases} 1 & \text{if } \chi_i > 0, \\ -1 & \text{otherwise.} \end{cases} \tag{49}$$

Therefore, the new point is computed by

$$x_2 = x_{ac} + \sigma_k, \tag{50}$$

where x_{ac} is the best point accepted so far, and then the value of the function f is calculated at the point x_2 ; $f_2 = f(x_2)$.

Step 3: The third point is computed randomly as follows:

$$x_3 = X_r + \frac{1}{2}DR, \tag{51}$$

where $DR = \frac{(1+\mu_k)^{|\chi_i|} - \omega}{(\mu_k + 0.1)} S\chi_i$; $\mu_k = \|x_{ac}\|^2$; x_{ac} is the best point that has been accepted so far; $\omega \in (0, 1)$ is taken randomly; X_r is a random variable generated from a research domain of the test problem; $X_r \sim [a, b]^n$, a , and b are the lower and upper bounds of the research domain, respectively; and the random vector χ_i with its signs $S\chi_i$ are defined by Step 2. Then, the value of function f is computed at the point x_3 ; $f_3 = f(x_3)$.

It is worth noticing that the ST technique allows for doing a comprehensive scanning of the search domain to ensure that the condition of finding the global point is satisfied at least once.

Therefore, the ST technique is hybridized with each CG method of a set methods MXHZ, MHZ, HZ, HS, and FR. The result is five global optimization algorithms which try to solve Problem (1). These algorithms are abbreviated by HSTMXHZ, HSTMHZ, HSTHZ, HSTHS, and HSTFR, respectively.

Consequently, Algorithm 2 contains five algorithms, being HSTMXHZ, HSTMHZ, HSTHZ, HSTHS, and HSTFR, respectively.

5.2. A Mechanism Running Algorithm 2

Algorithm 2 is a hybrid algorithm of two techniques. The first one is a conjugate gradient CG of the set methods $CG = \{MXHZ, MHZ, HZ, SH, FR\}$, which are shown in Section 3, and the second one is a stochastic technique described by Algorithm ST.

The point x_{cg} is generated by Algorithm 1, and it is an input to Algorithm 2.

Algorithm 2 starts from Line 1, which represents the stopping criterion of the algorithm, such that Algorithm 2 stops when one of the following conditions is met: if $|f_{ac} - f^*| \leq \varepsilon$ or $FEs \geq n10^4$, where f_{ac} is the best value of the function f obtained, f^* is the exact solution, $\varepsilon = 10^{-6}$, and FEs is the number of function evaluations denoted by [64,65] as a stopping criterion.

In Line 3, the best value of f is picked from the three function values f_{cg} , f_1 , and f_2 and denoted by f_{ac} . These three values of f are computed by Algorithm 1, (48), and (50), respectively, and x_{ac} denotes the best point that corresponds to the value f_{ac} . In Line 4, if $|f_{ac} - f^*| \leq \varepsilon$ is satisfied, the algorithm stops. The condition in Line 7 grants the algorithm a chance to escape from any local point, because at $\Delta f = 0$ (it is defined by (24)), this means that Algorithm 2 is at a critical point (stable point, stationary point, saddle point). If the norm of the gradient value is zero or near zero, then this point is either a local point, a saddle point, or the global point. In a case in which it is not the global point, then the algorithm does not stop, because the stopping criterion is not met. Hence, Algorithm 2 is given consecutive opportunities to escape this trap. Therefore, the processes in Lines

8–12 are capable of helping the algorithm to escape this trap, especially since the second stopping criterion ensures that most of the research domain will be surveyed.

Therefore, Algorithm 2 gains two fundamental features as follows: a deterministic research direction, which satisfies the descent direction, and walking randomly, which prevents falling into the trap.

6. Numerical Experiments

In this section, we present numerical results demonstrating the efficiency, reliability, and effectiveness of the proposed algorithm MXSTHZ in finding the global minimizer of the objective function f for a set of unconstrained optimization test problems.

All experiments were run on a PC with an Intel(R) Core(TM) i5-3230M CPU@2.60GHz 2.60 GHz with 4.00 GB of RAM on the Windows 10 operating system. All five algorithms were programmed using MATLAB version 8.5.0.197613 (R2015a), and the machine epsilon is about 10^{-16} .

The benchmark optimization test problems are divided into two types of test problems: the first one is the test problems whose objective functions have only one minimum point (no local minima except the global one (convex function)), and the second one is the test problems whose objective functions have several local minima with one global minimum (non-convex function). The test problems of the second type are denoted by * in Tables 1 and 2. In Table 1, the set of test problems is listed as follows: Columns 1–4 present the data of the problems; Column 1 presents the name of the function f ; Column 2 gives the number of variables; Column 3 gives the value of the function $f(x^*)$ at the exact global solution x^* ; and Column 4 gives the exact value the norm of the gradient $\|g(x^*)\|$. The market “-” means the values of the norm of the the gradient $\|g(x^*)\|$ for the convex function met the stopping criterion $\|g(x^*)\| < 10^{-6}$. Columns 5–8 are as Columns 1–4.

Table 1. Listing of test problems and their exact solutions.

f	n	$f(x^*)$	$\ g(x^*)\ $	f	n	$f(x^*)$	$\ g(x^*)\ $
CB*	2	-1.0316285	2×10^{-5}	Ma	2	0	-
GP*	2	3	2×10^{-6}	Le*	10	0	2.1×10^{-6}
DJ	3	0	-	HM*	2	0	1.1×10^{-8}
SH*	2	-186.7309	2×10^{-6}	BO	2	0	-
Ras*	2	-2	2.5×10^{-6}	Bh1*	2	0	2.4×10^{-5}
H3*	3	-3.86278	2×10^{-5}	H6*	6	-3.32237	6×10^{-5}
CV	4	0	-	P16*	5	0	1.2×10^{-6}
S5*	4	-10.1532	3.2×10^{-5}	S10*	4	-10.5364	3×10^{-5}
SP	10, 30, 80, 100	0	-	S7*	4	-10.4029	-
PWQ	8, 32, 84, 120	0	-	P8*	3	0	-
Rn	10, 30, 50, 80, 100	0	-	BR	2	0.397887	-
Zn	10, 30, 50, 80, 100	0	-				
Tr	10, 30, 60, 80	$\frac{-n(n+4)(n-1)}{6}$	-				
Su	10, 30, 50, 80, 100	0	-				

Table 2 shows some local points of the second type of test problems, Column 1 of Table 2, contains the name of non-convex function f , Column 2 presents some of local points x_{lo} , Column 3 gives the value of the $f(x_{lo})$, Column 4 presents the the norm of the gradient vector $\|g(x_{lo})\|$ at the point x_{lo} , and Column 5 gives the h.m (Hessian matrix), where the sign + means that the Hessian matrix is positive definite, i.e, the second-order necessary condition is satisfied.

The data in Tables 1 and 2 are taken from [35]. These benchmark optimization test problems are taken from [36,66–68].

In the following section, the performance profiles of the five algorithms are presented.

Performance Profiles

Performance profiles are the best tool for evaluating and comparing the performance of optimization algorithms [66,69–72].

Barbosa et al. [69] used the performance profiles to analyze the results of the 2006 CEC constrained optimization competition.

A fair comparison among different solvers should be based on the number of function evaluations, instead of based on the number of iterations or on the CPU time only.

The number of iterations is not a reliable measure because the amount of work done in each iteration is completely different among solvers, since some are population based, and others are single point based, since the quality of the solution is also an important measure of performance.

Therefore, the worst and best of the number of iterations and function emulations with the average of the CPU time, iterations, and function emulations are used to compare the performance of the five algorithms. Hence, we present the numerical results in the form of performance profiles, as described in [70]. This procedure was developed to benchmark optimization software, i.e., to compare different solvers on several test problems.

One advantage of the performance profiles is that they can be presented in one figure by plotting a cumulative distribution function $\rho_s(\tau)$ for the different solvers.

The performance ratio is defined by first setting $r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s}:s \in S\}}$, where $p \in P$, P is a set of test problems, S is the set of solvers, and $t_{p,s}$ is the value obtained by solver s on test problem p .

Then, define $\rho_s(\tau) = \frac{1}{|P|} \text{size}\{p \in P : r_{p,s} \leq \tau\}$, where $|P|$ is the number of test problems.

In the following, we show how the performance profiles are used to compare the performance of the five algorithms $S = \{HSTMXHZ, HSTMHZ, HSTHZ, HSTSH, HSTFR\}$, according to the worst and best number of iterations and function emulations and the average of the CPU time, iterations, and function emulations. They are denoted by itr.w, itr.be, FEs.w, FEs.be, time.ave, itr.ave, and EFs.ave, respectively.

Therefore, the term $t_{p,s}$ denotes one of the set $Fit = \{\text{itr.w, itr.be, FEs.w, FEs.be, time.ave, itr.ave, EFs.ave}\}$, and $|P| = 46$ is the number of test problems; we have 46 unconstrained test problems, 14 of them containing non-convex functions (they have many local minimizers). Therefore, the set of solvers S seeks to find the global minimizers of the 46 test problems. Hence, the values of the set Fit are computed as follows. Each solver s of the set S is run 51 times for every problem of 46 problems; at each run, the number of iterations and function emulations with the CPU time are computed. Thus, the average, best, and worst behavior can be analyzed as follows.

For each problem p and solver s , the performance ratio is defined as

$$r_{p,s} = \begin{cases} \frac{\text{fit}_{p,s}}{\min\{\text{fit}_{p,s}:s \in S\}} & \text{if convergence test passed,} \\ \infty & \text{otherwise,} \end{cases} \tag{52}$$

where $\text{fit}_{p,s}$ represents one element of the set Fit for a test problem p , which is solved by a solver s . We have

$$\delta(r_{p,s}, \tau) = \begin{cases} 1 & \text{if } r_{p,s} \leq \tau, \\ 0 & \text{otherwise.} \end{cases} \tag{53}$$

Therefore, the performance profile for solver s is then given by the following function:

$$\rho_s(\tau) = \frac{1}{|P|} \left\{ \sum_{p \in P} \delta(r_{p,s}, \tau) \right\}, \tau \geq 1. \tag{54}$$

In this paper, $|P| = 46$ is a number of test problems, and we set $\tau \in [1, 60]$.

By definition of $\text{Fit}_{p,s}$, $\rho_s(1)$ denotes the fraction of test problems for which solver s performs the best, $\rho_s(2)$ gives the fraction of problems for which the solver's performance is within a factor of 2 of the best, and for a τ sufficiently large, $\rho_s(\tau)$ is the fraction of problems solved by s .

In general, $\rho_s(\tau)$ can be interpreted as the probability for solver $s \in S$ that the performance ratio $r_{p,s}$ is within a factor τ of the best possible ratio. Therefore, $\rho_s(1)$ measures the efficiency of the solver, while its robustness (high probability of success on the set P) is measured in terms of $\rho_s(\infty)$. Hence, if we are only interested in determining which solver is the best, i.e., wins the most, we compare the values of $\rho_s(1)$ for all the solvers.

A core feature of performance profiles is that they give information on the relative performance of many solvers [70,71].

In the following, the results of the five algorithms are annualized by showing the performance profiles for each algorithm.

Figures 1–4 show the set solvers' performance profiles (five algorithms) according to the criteria mentioned in the set Fit . The performance profile in the left graph of Figure 1 (in terms of the worst number of iterations) compares the performance of the five algorithms on the 46 test problems.

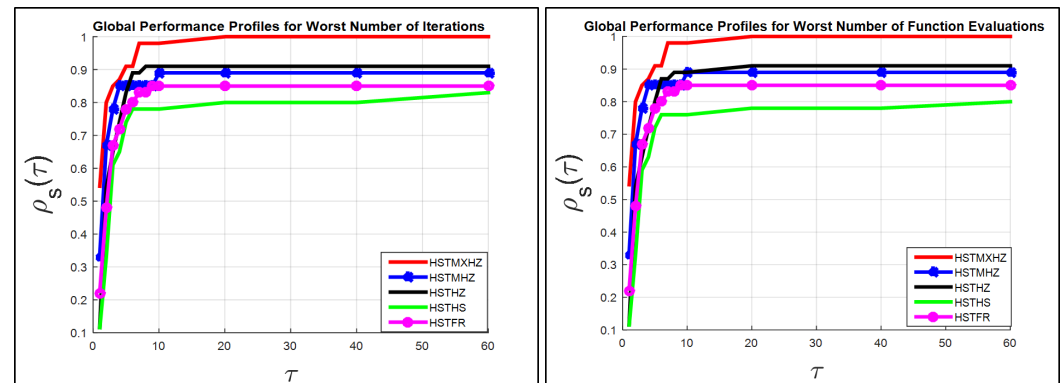


Figure 1. Worst number of iterations and function evaluations performance profile.

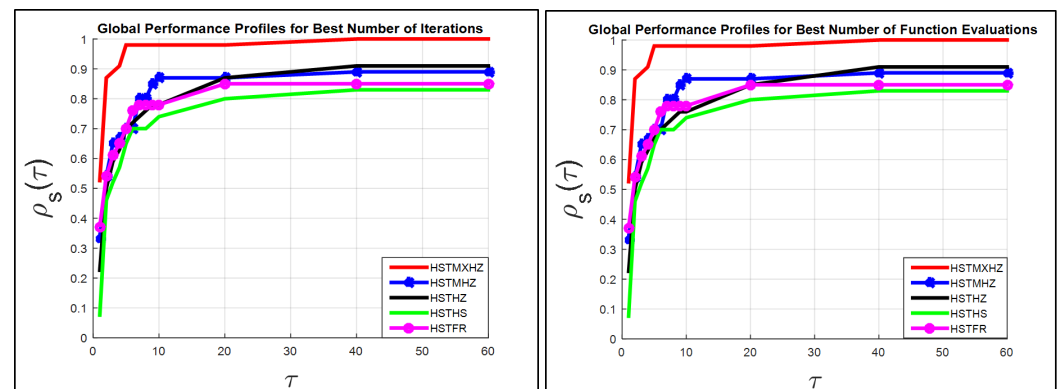


Figure 2. Best number of iterations and function evaluations performance profile.

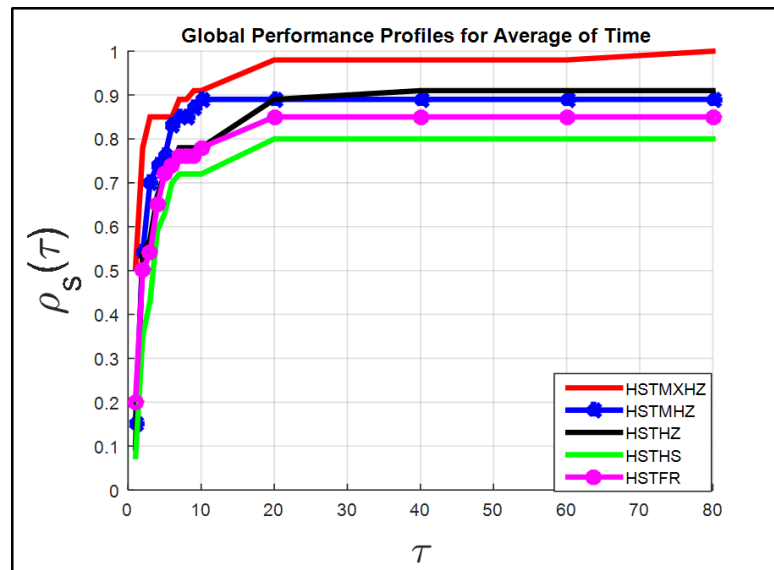


Figure 3. Average of CPU time performance profile.

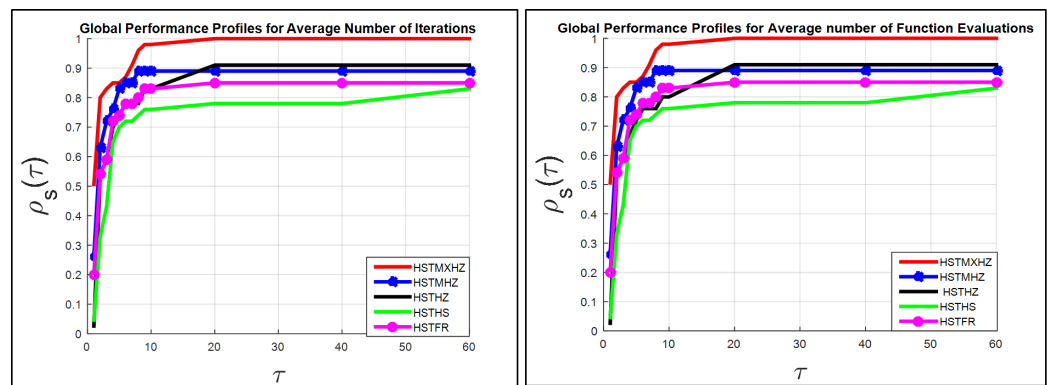


Figure 4. Average of iterations and function evaluations performance profile.

The HSTMXHZ algorithm has the best performance (in terms of the worst number of iterations) for all test problems, meaning that the HSTMXHZ algorithm is able to find the global optimal solutions of all test problems as fast as or faster than the other four algorithms. For example, when $\tau = 1$, the method HSTMXHZ finds the optimal solution for 54% of the test problems versus 33%, 11%, 11%, and 22% of the test problems being solved by HSTMHZ, HSTHZ, HSTHZ, HSTHS, and HSTFR algorithms, respectively.

Therefore, regarding the worst number of iterations criterion, the $\tau = 60$ shows us that all test problems are solved by HSTMXHZ, while 89%, 91%, 83%, and 85% of the test problems are solved by HSTMHZ, HSTHZ, HSTHZ, HSTHS, and HSTFR algorithms, respectively.

In general, the performance profiles of all algorithms depicted in Figures 1–4 show that the MXSTHZ algorithm has the features of efficiency, reliability, and effectiveness in finding the global minimizer of all test problems.

Consequently, the performance profiles illustrated in Figures 1–4 are summarized in one figure as a percentage of fails of all five algorithms, and this is depicted in Figure 5 as follows.

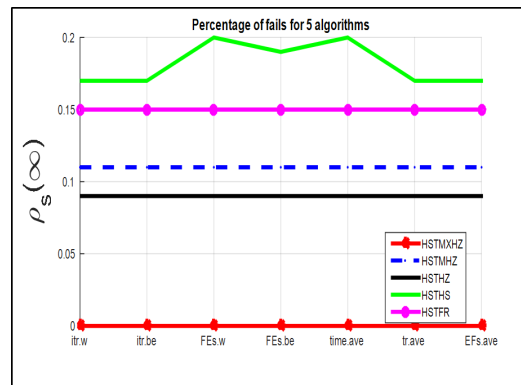


Figure 5. Percentage of fails for all 5 algorithms.

At $\tau = \infty$, we obtain the following results: algorithm HSTMXHZ does not fail to solve any of the test problems for the set criteria Fit. Algorithm HSTMHZ fails to solve 11% of the test problems according to the set criteria Fit. Algorithm HSTHZ fails to solve 9% of the test problems according to the set criteria Fit. Algorithm HSTHS fails to solve 17–20% of the test problems according to the set criteria Fit. Algorithm HSTFR fails to solve 15% of the test problems according to the set criteria Fit.

Note: The maximum value of the parameter τ in the X-axis is $\tau = 60$ in all figures except Figure 3, because it is shown that our proposed algorithm HSTMXHZ is not capable of solving all test problems at $\tau = 60$; thus, we increase the value of the parameter τ to $\tau = 80$. Hence, $\tau = 80$ shows that the HSTMXHZ algorithm solves all test problems regarding the average CPU time criterion.

Table 2. Local minima and their function values of the second type found by GL.

f	x_{l_0}	$f(x_{l_0})$	$\ \nabla f(x_{l_0})\ _2$	h.m
GP^*	−0.6	30	2.8×10^{-10}	+
	−0.4			
	1.8	84	2.8×10^{-12}	+
	0.2			
Ras^*	−1.2224947770423 −0.933379140693944	3.51839652096461	4.5804×10^{-8}	+
	$6.93003142568068 \times 10^{-21}$ 0.933379142773859	1.87456993111744	5.0516×10^{-8}	+
	0 −0.469527545774886	0.469882414921396	6.6593×10^{-9}	+
	0.618612067329937 −2.34734550064544 $\times 10^{-10}$	0.412926830275815	2.0389×10^{-8}	+
	0.61861206782795 −0.469527545778679	0.882809245197211	6.8967×10^{-9}	+
	−0.693844456281809 0.346923814676768	−1.39450436402534	9.1793×10^{-8}	+
	0.346923814707539 0.346923814679388	−1.75780130306047	9.256×10^{-9}	+
	−0.693844456091345 −8.86514604640448 $\times 10^{-12}$	−1.5156037124951	7.1113×10^{-8}	+

Table 2. *Cont.*

f	x_{lo}	$f(x_{lo})$	$\ \nabla f(x_{lo})\ _2$	h.m
HM^*	−1.60710475355569 −0.568651455138315	3.13588	6.8592×10^{-9}	+
	1.60710475653107 0.56865145632886	3.13587881031126	3.8976×10^{-8}	+
	1.70360671546671 −0.796083566882986	0.8161645	4.22×10^{-8}	+
$Bh1^*$	−0.61861206782795 0.469527545536589	0.882809245197211	8.2559×10^{-9}	+
	−1.22249477715681 0.469527545801467	2.11370900476857	8.323×10^{-9}	+
	0.618612068094813 −1.15759684702281 $\times 10^{-10}$	0.412926830275815	1.0414×10^{-8}	+
SH^*	−7.70831373549935 −8.29038801395222	−79.4109132228676	6.9801×10^{-6}	+
	0.334243912680165 0.821783925757999	−14.427	9.1083×10^{-7}	+
	6.08779955712239 9.56321202922249	−30.781	1.2437×10^{-6}	+
	3.77230798892767 −0.800321100471973	−52.05	2.5241×10^{-6}	+
	−0.195385750029861 −0.800321100471974	−123.58	1.1747×10^{-7}	+
$P8^*$	−8.91948740318191 2.98927244410315 −8.91911303921606	9.33771570831482	9.5779×10^{-9}	+
	2.95940520403097 2.95734250689826 6.91780883791202	6.219	3.5487×10^{-8}	+
	2.95985170644194 −1.00000000000001 −1	1.0367	1.2419×10^{-9}	+
	−4.95943986528779 −4.95903229916874 −0.999999999999996	2.0732	5.207×10^{-8}	+
$P16^*$	−0.316713428439135 5.96534445787178 0.000485720005736201 3.30614825102299 2.99682664024085	3.6966869674758	6.6557×10^{-8}	+
	2.31787978846433 3.64917986009329 1.00000000064691 1.00000015159645 1.0000002427518	0.882277405883787	9.9009×10^{-8}	+
	1.65876201672915 1.97331911501372 1.98801852779664 1.98844579485365 0.00794023516825855	0.438258122943419	6.1217×10^{-8}	+

Table 2. *Cont.*

f	x_{l_0}	$f(x_{l_0})$	$\ \nabla f(x_{l_0})\ _2$	h.m
CB^*	−1.60710475750554 −0.56865145671785	2.10425031031126	4.9491×10^{-8}	+
	−1.70360671560327 0.796083566391017	−0.215463824383719	5.3745×10^{-8}	+
$H3^*$	0.368722727070142 0.117561628830344 0.267573742135187	−1.0008	5.262×10^{-8}	+
	0.109337500834921 0.860524221821118 0.564123171483196	−3.0898	6.7792×10^{-8}	+
	7.99958330481318 7.99964158942217 7.99958330481318 7.999641589422	−5.1008	5.4716×10^{-8}	+
$S5^*$	1.00013158830876 1.00015634035488 1.00013158830876 1.00015634035488	−5.0552	8.901×10^{-8}	+
	3.00179639486718 6.99833393860998 3.00179639486718 6.99833393860998	−2.6305	1.7899×10^{-8}	+
	5.9987493870904 6.00028741439389 5.9987493870904 6.00028741439389	−2.6829	2.777×10^{-6}	+
	1.00023247902693 1.00027365356478 1.00018321203747 1.0002243825922	−5.0877	1.0202×10^{-7}	+
$S7^*$	3.00090958754435 7.00064162342178 3.00036903207152 7.00010106794894	−2.7659	9.2043×10^{-9}	+
	4.99422913368074 4.99499394371211 3.00606373374785 3.00682853864518	−3.7243	5.9883×10^{-8}	+
	5.99810675361719 6.00008258056314 5.99732997276745 5.99930579971341	−2.7519	8.2126×10^{-10}	+

Table 2. *Cont.*

f	x_{lo}	$f(x_{lo})$	$\ \nabla f(x_{lo})\ _2$	h.m			
S10*	1.00036625923444	−5.1285	1.1938×10^{-7}	+			
	1.00030224282025						
	1.00031698868663						
	1.00025296814889						
	4.99487209870931	−3.8354	2.7328×10^{-8}	+			
	4.99398146021396						
	3.00755591238878						
	3.00666527389401						
	6.00557890531025	−2.4217	7.9048×10^{-13}	+			
	2.01001498366408						
	6.00437006308492						
	2.00880614143875						
	5.99901345118408	−2.8711	6.5965×10^{-10}	+			
	5.99728366454457						
	5.99823624874706						
	5.99650646210755						
	6.99163536369117	−2.4273	8.2572×10^{-14}	+			
	3.59557985427872						
	6.99065644577223						
	3.59460093635978						
H6*	0.40465313	−3.2032	2.5475×10^{-7}	+			
	0.882444924						
	0.846101569						
	0.573989692						
	0.138926604						
	0.038495892						
Le*	1	12.8829396938135	5.3753×10^{-8}	+			
	7.70247946443158						
	7.70247946470933						
	−4.2421171417158						
	1.00000004177889						
	1.00000002319757						
	0.99999998279399						
	7.70247946465686						
	−4.2421171417158						
	−2.89064960846171						
	3.791082199734824				2.24815774868836	8.7862×10^{-8}	+
	1						
	1						
	−0.0929936950939674						
	1						
	1						
	3.66435455838079	2.24815774868836	8.7862×10^{-8}	+			
	3.66435455838079						
−0.0929936950939674							
0.999999472819993							
1							
1							

Table 2. *Cont.*

f	x_{l0}	$f(x_{l0})$	$\ \nabla f(x_{l0})\ _2$	h.m
	3.79108219979366	3.06727753545346	8.2255×10^{-8}	+
	1			
	3.66435455838079			
	−0.0929936950939674			
	3.66435455838079			
Le^*	1			
	3.66435455838079			
	1			
	3.66435455838079			
	1.00000046273142			

h.m: Hessian matrix information determines the nature of the stationary point.

7. Conclusions and Future Work

We have proposed a conjugate gradient method which is abbreviated by MXHZ for solving unconstrained optimization problems. Although the updated formulas of the proposed method MXHZ are more complicated than previous formulas, the scheme is very robust in numerical experiments. The global convergence analysis of the MXHZ method is established. Furthermore, we have used finite differences (forward differences DFF) for computing the approximate gradient vector ($g(x) \approx DFF$). We have provided the method (DFF) with a new approach for estimating an appropriate step-size randomly. Calculating the value of the step-size h is connected to the value of a function f at each iteration. Extensive numerical experiments showed that the results of the proposed algorithm (HSTMXHZ) are highly competitive with four other algorithms on performance profiles. The suggested stochastic approach has played a key role in making the HSTMXHZ algorithm capable of finding the global minimizers of unconstrained optimization test problems, especially when the objective function is not convex. Comparing the final results between the true values of the gradient vector with the estimated values which are obtained by the method DFF shows that the new approach has succeeded in selecting the proper step-size h .

Therefore, our suggested algorithm HSTMXHZ gains two fundamental features: a deterministic research direction which satisfies the descent direction and walking randomly that prevents falling into a local point. For future work, the proposed algorithm can be enhanced and modified to solve constrained and multi-objective optimization problems, and the global convergence analysis of the HSTMXHZ algorithm will be considered. Furthermore, the conjugate gradient methods play an important role in many fields of optimization problems. Therefore, it is necessary for us to further improve the theoretical properties and numerical performance of these methods and make them more widely used: for example, they can be used to solve unconstrained and constrained multi-objective optimization problems as well as nonlinear equations.

Author Contributions: Conceptualization, K.A.A.; Data curation, A.M.A.; Formal analysis, S.M.; Funding acquisition, A.M.A., A.F.A. and K.A.A.; Investigation, A.M.A. and A.F.A.; Methodology, S.M. and A.W.M.; Project administration, A.W.M.; Resources, A.F.A. and K.A.A.; Software, S.M.; Supervision, A.W.M.; Writing—original draft, S.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Researchers Supporting Program at King Saud University, (Project# RSP-2021/323).

Acknowledgments: The authors present their appreciation to King Saud University for funding the publication of this research through the Researchers Supporting Program (RSP-2021/323), King Saud University, Riyadh, Saudi Arabia.

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

References

1. 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.
2. Hillier, F.S.; Price, C.C. *International Series in Operations Research & Management Science*; Springer Nature: Cham, Switzerland, 2001.
3. Laarhoven, P.J.V.; Aarts, E.H. *Simulated Annealing: Theory and Applications*; Springer-Science + Business Media, B.V.: Dordrecht, The Netherlands, 1987.
4. Hager, W.W.; Zhang, H. Algorithm 851: CG_DESCENT, a conjugate gradient method with guaranteed descent. *ACM Trans. Math. Softw. (TOMS)* **2006**, *32*, 113–137. [[CrossRef](#)]
5. Zhang, L.; Zhou, W.; Li, D.H. A descent modified Polak–Ribière–Polyak conjugate gradient method and its global convergence. *IMA J. Numer. Anal.* **2006**, *26*, 629–640. [[CrossRef](#)]
6. Waziri, M.Y.; Kiri, A.I.; Kiri, A.A.; Halilu, A.S.; Ahmed, K. A modified conjugate gradient parameter via hybridization approach for solving large-scale systems of nonlinear equations. *SeMA J.* **2022**, 1–23. [[CrossRef](#)]
7. Zhang, L.; Zhou, W.; Li, D. Global convergence of a modified Fletcher–Reeves conjugate gradient method with Armijo-type line search. *Numer. Math.* **2006**, *104*, 561–572. [[CrossRef](#)]
8. Aji, S.; Kumam, P.; Siricharoen, P.; Abubakar, A.B.; Yahaya, M.M. A modified conjugate descent projection method for monotone nonlinear equations and image restoration. *IEEE Access* **2020**, *8*, 158656–158665. [[CrossRef](#)]
9. Ibrahim, A.H.; Kumam, P.; Kumam, W. A family of derivative-free conjugate gradient methods for constrained nonlinear equations and image restoration. *IEEE Access* **2020**, *8*, 162714–162729. [[CrossRef](#)]
10. Su, Z.; Li, M. A Derivative-Free Liu–Storey Method for Solving Large-Scale Nonlinear Systems of Equations. *Math. Probl. Eng.* **2020**, *2020*, 6854501. [[CrossRef](#)]
11. Xiao, Y.; Zhu, H. A conjugate gradient method to solve convex constrained monotone equations with applications in compressive sensing. *J. Math. Anal. Appl.* **2013**, *405*, 310–319. [[CrossRef](#)]
12. Yuan, G.; Li, T.; Hu, W. A conjugate gradient algorithm for large-scale nonlinear equations and image restoration problems. *Appl. Numer. Math.* **2020**, *147*, 129–141. [[CrossRef](#)]
13. Golub, G.H.; O’Leary, D.P. Some history of the conjugate gradient and Lanczos algorithms: 1948–1976. *SIAM Rev.* **1989**, *31*, 50–102. [[CrossRef](#)]
14. Hager, W.W.; Zhang, H. A survey of nonlinear conjugate gradient methods. *Pac. J. Optim.* **2006**, *2*, 35–58.
15. Hestenes, M.R.; Stiefel, E. Methods of Conjugate Gradients for Solving. *J. Res. Natl. Bur. Stand.* **1952**, *49*, 409. [[CrossRef](#)]
16. Fletcher, R.; Reeves, C.M. Function minimization by conjugate gradients. *Comput. J.* **1964**, *7*, 149–154. [[CrossRef](#)]
17. Polak, E.; Ribiere, G. Note sur la convergence de méthodes de directions conjuguées. *ESAIM Math. Model. Numer. Anal.* **1969**, *3*, 35–43. [[CrossRef](#)]
18. Polyak, B.T. The conjugate gradient method in extremal problems. *USSR Comput. Math. Math. Phys.* **1969**, *9*, 94–112. [[CrossRef](#)]
19. Liu, Y.; Storey, C. Efficient generalized conjugate gradient algorithms, part 1: Theory. *J. Optim. Theory Appl.* **1991**, *69*, 129–137. [[CrossRef](#)]
20. Dai, Y.H.; Yuan, Y. A nonlinear conjugate gradient method with a strong global convergence property. *SIAM J. Optim.* **1999**, *10*, 177–182. [[CrossRef](#)]
21. Abubakar, A.B.; Kumam, P. A descent Dai–Liao conjugate gradient method for nonlinear equations. *Numer. Algorithms* **2019**, *81*, 197–210. [[CrossRef](#)]
22. Abubakar, A.B.; Muangchoo, K.; Ibrahim, A.H.; Muhammad, A.B.; Jolaoso, L.O.; Aremu, K.O. A new three-term Hestenes–Stiefel type method for nonlinear monotone operator equations and image restoration. *IEEE Access* **2021**, *9*, 18262–18277. [[CrossRef](#)]
23. Babaie-Kafaki, S.; Ghanbari, R. A descent family of Dai–Liao conjugate gradient methods. *Optim. Methods Softw.* **2014**, *29*, 583–591. [[CrossRef](#)]
24. Dai, Y.-H.; Liao, L.-Z. New conjugacy conditions and related nonlinear conjugate gradient methods. *Appl. Math. Optim.* **2001**, *43*, 87–101. [[CrossRef](#)]
25. Yuan, G.; Zhang, M. A three-terms Polak–Ribière–Polyak conjugate gradient algorithm for large-scale nonlinear equations. *J. Comput. Appl. Math.* **2015**, *286*, 186–195. [[CrossRef](#)]
26. Yuan, G.; Jian, A.; Zhang, M.; Yu, J. A modified HZ conjugate gradient algorithm without gradient Lipschitz continuous condition for non convex functions. *J. Appl. Math. Comput.* **2022**, 1–22. [[CrossRef](#)]
27. Zhou, Y.; Wu, Y.; Li, X. A new hybrid prpr conjugate gradient method for solving nonlinear monotone equations and image restoration problems. *Math. Probl. Eng.* **2020**, *2020*, 6391321. [[CrossRef](#)]
28. Abubakar, A.B.; Malik, M.; Kumam, P.; Mohammad, H.; Sun, M.; Ibrahim, A.H.; Kiri, A.I. A Liu–Storey-type conjugate gradient method for unconstrained minimization problem with application in motion control. *J. King Saud Univ.-Sci.* **2022**, *34*, 101923. [[CrossRef](#)]
29. Dai, Y.H.; Yuan, Y. An efficient hybrid conjugate gradient method for unconstrained optimization. *Ann. Oper. Res.* **2001**, *103*, 33–47. [[CrossRef](#)]
30. Deng, S.; Wan, Z. A three-term conjugate gradient algorithm for large-scale unconstrained optimization problems. *Appl. Numer. Math.* **2015**, *92*, 70–81. [[CrossRef](#)]
31. Ma, G.; Lin, H.; Jin, W.; Han, D. Two modified conjugate gradient methods for unconstrained optimization with applications in image restoration problems. *J. Appl. Math. Comput.* **2022**, 1–26. [[CrossRef](#)]

32. Mtagulwa, P.; Kaelo, P. An efficient modified PRP-FR hybrid conjugate gradient method for solving unconstrained optimization problems. *Appl. Numer. Math.* **2019**, *145*, 111–120. [[CrossRef](#)]
33. Kan, A.R.; Timmer, G. Stochastic methods for global optimization. *Am. J. Math. Manag. Sci.* **1984**, *4*, 7–40. [[CrossRef](#)]
34. 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. [[CrossRef](#)]
35. 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](#)]
36. Hedar, A.R.; Fukushima, M. Hybrid simulated annealing and direct search method for nonlinear unconstrained global optimization. *Optim. Methods Softw.* **2002**, *17*, 891–912. [[CrossRef](#)]
37. Pedamallu, C.S.; Ozdamar, L. Investigating a hybrid simulated annealing and local search algorithm for constrained optimization. *Eur. J. Oper. Res.* **2008**, *185*, 1230–1245. [[CrossRef](#)]
38. Yiu, K.F.C.; Liu, Y.; Teo, K.L. A hybrid descent method for global optimization. *J. Glob. Optim.* **2004**, *28*, 229–238. [[CrossRef](#)]
39. Bertsekas, D.P. *Nonlinear Programming*; Athena Scientific: Belmont, MA, USA, 1999.
40. Bonnans, J.F.; Gilbert, J.C.; Lemaréchal, C.; Sagastizábal, C.A. *Numerical Optimization: Theoretical and Practical Aspects*; Springer Science & Business Media: Berlin, Germany, 2006.
41. Dennis, J.E., Jr.; Schnabel, R.B. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*; SIAM: Philadelphia, PA, USA, 1996; Volume 16.
42. Gilbert, J.C.; Nocedal, J. Global convergence properties of conjugate gradient methods for optimization. *SIAM J. Optim.* **1992**, *2*, 21–42. [[CrossRef](#)]
43. Nocedal, J.; Wright, S. *Numerical Optimization*; Springer Science & Business Media: Berlin, Germany, 2006.
44. Chan, T.F.; Esedoglu, S.; Nikolova, M. Algorithms for finding global minimizers of image segmentation and denoising models. *SIAM J. Appl. Math.* **2006**, *66*, 1632–1648. [[CrossRef](#)]
45. Zhenjun, S. A new memory gradient method under exact line search. *Asia-Pac. J. Oper. Res.* **2003**, *20*, 275–284.
46. Hager, W.W.; Zhang, H. A new conjugate gradient method with guaranteed descent and an efficient line search. *SIAM J. Optim.* **2005**, *16*, 170–192. [[CrossRef](#)]
47. Al-Baali, M. Descent property and global convergence of the Fletcher Reeves method with inexact line search. *IMA J. Numer. Anal.* **1985**, *5*, 121–124. [[CrossRef](#)]
48. Yuan, G.; Meng, Z.; Li, Y. A modified Hestenes and Stiefel conjugate gradient algorithm for large-scale nonsmooth minimizations and nonlinear equations. *J. Optim. Theory Appl.* **2016**, *168*, 129–152. [[CrossRef](#)]
49. Yuan, G.; Wei, Z.; Yang, Y. The global convergence of the Polak–Ribière–Polyak conjugate gradient algorithm under inexact line search for nonconvex functions. *J. Comput. Appl. Math.* **2019**, *362*, 262–275. [[CrossRef](#)]
50. Zoutendijk, G. Nonlinear programming, computational methods. *Integer Nonlinear Program.* **1970**, 37–86.
51. Wolfe, P. Convergence conditions for ascent methods. *SIAM Rev.* **1969**, *11*, 226–235. [[CrossRef](#)]
52. Wolfe, P. Convergence conditions for ascent methods. II: Some corrections. *SIAM Rev.* **1971**, *13*, 185–188. [[CrossRef](#)]
53. Kramer, O.; Ciaurri, D.E.; Koziel, S. Derivative-free optimization. In *Computational Optimization, Methods and Algorithms*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 61–83.
54. Larson, J.; Menickelly, M.; Wild, S.M. Derivative-free optimization methods. *Acta Numer.* **2019**, *28*, 287–404. [[CrossRef](#)]
55. Shi, H.J.M.; Xie, Y.; Xuan, M.Q.; Nocedal, J. Adaptive Finite-Difference Interval Estimation for Noisy Derivative-Free Optimization. *arXiv* **2021**, arXiv:2110.06380.
56. Shi, H.J.M.; Xuan, M.Q.; Oztoprak, F.; Nocedal, J. On the numerical performance of derivative-free optimization methods based on finite-difference approximations. *arXiv* **2021**, arXiv:2102.09762.
57. Oliver, J.; Ruffhead, A. The selection of interpolation points in numerical differentiation. *BIT Numer. Math.* **1975**, *15*, 283–295. [[CrossRef](#)]
58. Berahas, A.S.; Cao, L.; Choromanski, K.; Scheinberg, K. A theoretical and empirical comparison of gradient approximations in derivative-free optimization. *Found. Comput. Math.* **2022**, *22*, 507–560. [[CrossRef](#)]
59. Curtis, A.; Reid, J. The choice of step lengths when using differences to approximate Jacobian matrices. *IMA J. Appl. Math.* **1974**, *13*, 121–126. [[CrossRef](#)]
60. Calio, F.; Frontini, M.; Milovanović, G.V. Numerical differentiation of analytic functions using quadratures on the semicircle. *Comput. Math. Appl.* **1991**, *22*, 99–106. [[CrossRef](#)]
61. Gill, P.E.; Murray, W.; Saunders, M.A.; Wright, M.H. Computing forward-difference intervals for numerical optimization. *SIAM J. Sci. Stat. Comput.* **1983**, *4*, 310–321. [[CrossRef](#)]
62. Xie, Y. Methods for Nonlinear and Noisy Optimization. Ph.D. Thesis, Northwestern University, Evanston, IL, USA, 2021.
63. De Levie, R. An improved numerical approximation for the first derivative. *J. Chem. Sci.* **2009**, *121*, 935–950. [[CrossRef](#)]
64. 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.
65. Mohamed, A.W.; Hadi, A.A.; Mohamed, A.K.; Awad, N.H. Evaluating the performance of adaptive gainsharing knowledge based algorithm on cec 2020 benchmark problems. In Proceedings of the 2020 IEEE Congress on Evolutionary Computation (CEC), Glasgow, UK, 19–24 July 2020; IEEE: New York, NY, USA, 2020; pp. 1–8.

66. Ali, M.M.; Khompatraporn, C.; Zabinsky, Z.B. A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. *J. Glob. Optim.* **2005**, *31*, 635–672. [[CrossRef](#)]
67. Dekkers, A.; Aarts, E. Global optimization and simulated annealing. *Math. Program.* **1991**, *50*, 367–393. [[CrossRef](#)]
68. Tsoulos, I.G.; Stavrakoudis, A. Enhancing PSO methods for global optimization. *Appl. Math. Comput.* **2010**, *216*, 2988–3001. [[CrossRef](#)]
69. Barbosa, H.J.; Bernardino, H.S.; Barreto, A.M. Using performance profiles to analyze the results of the 2006 CEC constrained optimization competition. In Proceedings of the IEEE Congress on Evolutionary Computation, Glasgow, UK, 19–24 July 2020; IEEE: New York, NY, USA, 2010; pp. 1–8.
70. Dolan, E.D.; Moré, J.J. Benchmarking optimization software with performance profiles. *Math. Program.* **2002**, *91*, 201–213. [[CrossRef](#)]
71. Moré, J.J.; Wild, S.M. Benchmarking derivative-free optimization algorithms. *SIAM J. Optim.* **2009**, *20*, 172–191. [[CrossRef](#)]
72. Vaz, A.I.F.; Vicente, L.N. A particle swarm pattern search method for bound constrained global optimization. *J. Glob. Optim.* **2007**, *39*, 197–219. [[CrossRef](#)]