Journal of Decision Science and Optimization
Vol. 2 (2026), No. 2, pp. 97-112
DOI: https://doi.org/10.55578/jdso.2606.007

ARTICLE

A New Adaptive Weighted Hybrid Conjugate Gradient Method for Unconstrained Optimization Problems

Yunchol Jong[1,*], Yongjin Kim[1], Daesong Ri[1]
1 Department of Mathematics, University of Science, Pyongyang, DPR Korea
[*]Corresponding author. Email: jyc1963@star-co.net.kp

Received: 29 Apr 2026, Accepted: 9 May 2026, Published: 2 June 2026

Abstract

In this paper, a new adaptive weighted hybrid conjugate gradient (AWHCG) method with backtracking line search is proposed to solve unconstrained optimization problems. Our method is based on the weighted hybridization of HS (Hestenes-Stiefel) and FR (Fletcher- Reeves) conjugate gradient methods. In contrast with the existing hybrid conjugate gradient methods, our AWHCG method updates adaptively the coefficient of gradient of objective function to be negative at each iteration so that our current search direction is always a combination of the anti- gradient of the objective function and the preceding direction. Under some reasonable assumptions, the global convergence of the proposed algorithm is established. Some preliminary numerical experiments show that the AWHCG algorithm is competitive.

Keywords: Unconstrained optimization, Conjugate gradient methods, Global convergence, Backtracking line search, Sufficient descent

1. INTRODUCTION

Consider the following unconstrained optimization problem

$$\min_{x\in R^n} f(x) \tag{1}$$

where $f: R^n \to R$ is continuously differentiable and bounded from below.

The conjugate gradient (CG) method is one of the most popular classes of iterative methods to solve the unconstrained optimization problem (1). The general iteration scheme of CG method [1,11] is defined by

$$x_{k+1} = x_k + t_k d_k, k = 0,1,\dots, \tag{2}$$

$$d_k = \begin{cases} -g_k, & k = 0, \\ -g_k + \beta_k d_{k-1}, & k > 0, \end{cases} \tag{3}$$

where x_k is the current iterate, t_k is the step size obtained by the appropriate line search, $g_k = \nabla f(x_k)$ and β_k is an appropriately defined real scalar, known as the CG parameter.

The line search in CG methods is usually based on the Wolfe conditions and Armijo condition [1,11]. The Wolfe conditions have attracted special attention in the convergence analyses and the actual implementations, requiring that

© 2026 The Authors. Published by Nexus Press B.V.
This is an open access article under the CC BY 4.0 license (https://creativecommons.org/licenses/by/4.0/).

$$f(x_k + t_k d_k) \leq f_k + \rho t_k g_k^T d_k, \tag{4}$$
$$g(x_k + t_k d_k)^T d_k \geq \sigma g_k^T d_k, \tag{5}$$

where $f_k = f(x_k)$ and $0 < \rho < \sigma < 1$. The condition (4) and (5) is called Armijo condition and curvature condition, respectively. The strong Wolfe condition requires (4) and

$$| g(x_k + t_k d_k)^T d_k | \leq \sigma | g_k^T d_k |. \tag{6}$$

instead of (5). The step size t_k satisfying the Armijo condition (4) is often determined using the backtracking line search procedure [1]. The procedure starts from $t = 1$, and reduces t until $x_k + td_k$ satisfies the Armijo condition (4). The procedure is as follows.

Algorithm 1 (Armijo's Backtracking Line Search)

Step 1. Given $\rho \in (0,1/2)$, $\alpha \in (0,1)$, set $t = 1$.

Step 2. Test the following inequality for true.

$$f(x_k + td_k) \leq f_k + \rho t g_k^T d_k \tag{7}$$

Step 3. If the above condition is not satisfied, set $t := t\alpha$, and go to Step 2; otherwise, set $t_k := t$.

A number of CG methods have been defined using various modifications of the CG direction d_k and the parameter β_k. The most popular CG parameters β_k are listed as follows:

$$\beta_k^{FR} = \frac{\| g_k \|^2}{\| g_{k-1} \|^2}[1,10], \beta_k^{CD} = \frac{-\| g_k \|^2}{g_k^T d_{k-1}}[11], \beta_k^{DY} = \frac{\| g_k \|^2}{y_k^T d_{k-1}}[1,12], \tag{8}$$

$$\beta_k^{HS} = \frac{y_k^T g_k}{y_k^T d_{k-1}}[1,13], \beta_k^{PRP} = \frac{y_k^T g_k}{\| g_{k-1} \|^2}[1,14], \beta_k^{LS} = \frac{-y_k^T g_k}{g_k^T d_{k-1}}[16], \tag{9}$$

where $y_{k-1} = g_k - g_{k-1}$. It is mentioned in [2] that FR [10], Conjugate Descent (CD) [11] and Dai-Yuan (DY) [12] methods have good theoretical convergence properties, but they may not show good performance in practice due to jamming. On the other hand, HS [13], Polak-Ribière-Polyak (PRP) [14, 15], and Liu-Storey (LS) [16] CG methods may not converge in general, but they often show better performance than FR, CD and DY in practice. Yang et al. [17] proposed LSCD method which combines the nice numerical performances of the LS method and the global convergence properties of the CD method. Dai and Liao [18] proposed an efficient CG type method (Dai-Liao method), and Zheng et al. [4] proposed more efficient Dai-Liao-type CG methods, known as DHSDL and DLSDL. Those are as follows:

$$\beta_k^{DL} = \frac{g_k^T (y_{k-1} - v s_{k-1})}{y_{k-1}^T s_{k-1}}[18], \beta_k^{LSCD} = \max\{0, \min\{\beta_k^L s, \beta_k^C D\}\}[17], \tag{10}$$

$$\beta_k^{DHSDL} = \frac{\|g_k\|^2 - \frac{\|g_k\|}{\|g_{k-1}\|}\|g_{k-1}\||g_k|}{|\mu| g_k^T d_{k-1}| + y_{k-1}^T d_{k-1}} - t_k \frac{g_k^T s_{k-1}}{y_{k-1}^T d_{k-1}}, \beta_k^{DLSDL} = \frac{\|g_k\|^2 - \frac{\|g_k\|}{\|g_{k-1}\|}|g_{k-1}\||g_k|}{|\mu| g_k^T d_{k-1}| - g_{k-1}^T d_{k-1}} - t_k \frac{g_k^T s_{k-1}}{y_{k-1}^T d_{k-1}},$$

where $v > 0$ is a parameter and $s_k = x_{k+1} - x_k$. The PRP-FR [19, 20] and HS-DY [22] methods are hybrid CG methods of LSCD type. In [21] and [23] have been proposed the following CG methods, WYL and YWH, respectively.

$$\beta_k^{WYL} = \frac{\| g_k \|^2 - \frac{\| g_k \|}{\| g_{k-1} \|} g_k^T g_{k-1}}{\| g_{k-1} \|^2}, \beta_k^{YWH} = \frac{\| g_k \|^2 - \frac{\| g_k \|}{\| g_{k-1} \|} g_k^T g_{k-1}}{d_k^T g_{k-1} y_{k-1}}. \tag{11}$$

To improve the LSCD, DHSDL and DLSDL, Stanimirović et al. [2] proposed two efficient hybrids of CG methods, termed as MLSCD and MMDL, respectively. The search directions in MLSCD and MMDL, instead of (3), are defined by

$$d_k = D(\beta_k, g_k, d_{k-1}) = -\left(1+\beta_k \frac{g_k^T d_{k-1}}{\| g_k \|^2}\right) g_k + \beta_k d_{k-1}, \tag{12}$$

where $\beta_k = \beta_k^{LSCD}$ and $\beta_k = \beta_k^{MMDL} = \max\{0, \min\{\beta_k^{HDSL}, \beta_k^{LDLSL}\}\}$, respectively [2]. They proved the global convergence of these two methods with the backtracking line search Algorithm 1 and showed that MLSCD method had the best performance with respect to the number of iterations, the number of function evaluations and the CPU time compared with LSCD, DHSDL, DLSDL and MMDL through numerical experiments. In [5] was proposed another hybrid CG method (EPF method) using the direction (12) which combines the features of the PRP and FR methods in β_k and established the global convergence of the EPF method with the strong Wolfe line search (4) and (6). With the Wolfe line search, [7] has considered the global convergence of HYBRID which is a convex combination of HS and DY methods, and [8] proposed a hybrid CG method JHJ-N which is one of hybrids of DY, FR, WYL and YWH CG parameters.

Dong et al. [3] proposed an adaptive three-term CG method satisfying the sufficient descent condition and the conjugacy condition, which is termed as CGADP method and is a little modification to the TTCG method [9] viewed as a variant of HS method. The global convergence of CGADP was proved under Wolfe line search (4) and (5). In [6], Chen et al. presented a new CG method, termed as NACG, which generates search direction satisfying both the sufficient descent condition and the Dai-Liao conjugacy condition independent of line search. The global convergence of the NACG was established with a Wolfe line search under proper assumptions. They showed that the NACG method is competitive for unconstrained large-scale optimization problems through numerical experiments.

Inspired by the research of [2,5], we are interested in developing a new adaptive hybrid CG method for unconstrained optimization (1). The new method makes use of a search direction of (12) type and CG parameter of β_k^{LSCD} type. In our method, coefficient of g_k is always negative in each iteration and the CG parameter is defined by weighted hybridization of modified HS and FR parameters.

The rest of this paper is organized as follows. The next section presents the new adaptive hybrid CG method and Sect. 3 considers global convergence of the proposed method with the backtracking line search under appropriate assumptions. In Section 4 are given numerical experiments and comparisons with some other efficient CG algorithms for some test problems with different dimensions. Some final conclusions are given in Sect. 5.

2. NEW ADAPTIVE WEIGHTED HYBRID CG METHOD

In this section, we present a new adaptive hybrid conjugate gradient method relating to the HS and FR methods. Among methods of (8) and (9), the HS method, which is like the PRP method in practical computation, was often recommended due to its superior numerical performances. The HS method satisfies automatically the standard conjugacy condition $d_k^T y_{k-1} = 0$ with the direction (3), which is independent of line search used. The FR method was proved to be globally convergent with the strong Wolfe line search with $\sigma < 1/2$ and it is one of the most widely used methods in practice. However, in

general, the FR method does not have the descent property with (3) and is often used in conjunction with exact line search.

The direction (12) satisfies the descent property independently of β_k, but HS method with (12) does not satisfy the standard conjugacy condition. Moreover, in contrast with the standard CG direction (3), it is not guaranteed that $-\left(1+\beta_k \frac{g_k^T d_{k-1}}{\|g_k\|^2}\right)$, the coefficient of g_k, is always negative.

We propose a hybrid CG method which updates the coefficient of g_k to be negative and uses weighted hybridization of the good numerical performances of the HS method and the global convergence properties of the FR method. In our new algorithm, search direction is determined by

$$l_k = \lambda_k + \beta_k \frac{g_k^T d_{k-1}}{\| g_k \|^2}, \tag{13}$$

$$d_k = D(\lambda_k, \beta_k, g_k, d_{k-1}) = -l_k g_k + \beta_k d_{k-1}, \tag{14}$$

where λ_k is a positive parameter iteratively updated so that l_k is always positive scalar. Now, we present our new algorithm.

Algorithm 2 Adaptive Weighted Hybrid CG method (AWHCG method)

Step 0. Choose $x_0 \in R^n$, $0 < \rho, \alpha, \varepsilon, \delta < 1$, $\lambda_0, \theta, w_1, w_2, \tau_1, \tau_2 > 0$. Let $d_0 = -g_0$, $k = 0$ and let f_1 be sufficiently large real scalar.

Step 1. If $\| g_k \| \leq \varepsilon$ or $\frac{|f_{k+1} - f_k|}{1+|f_k|} \leq \delta$, stop.

Step 2. Find a step size t_k using Algorithm 1 (Armijo's backtracking line search).

Step 3. Let $x_{k+1} = x_k + t_k d_k$. Compute g_{k+1}, f_{k+1}, $y_k = g_{k+1} - g_k$ and $s_k = x_{k+1} - x_k$. Set $k = k + 1$.

Step 4. Compute the following CG parameters.

$$\beta_k^{HS1} = \frac{y_{k-1}^T g_k}{\max\{y_{k-1}^T d_{k-1}, \tau_1 \| g_k \| \| d_{k-1} \|\}}, \beta_k^{FR1} = \frac{\| g_k \|^2}{\max\{\| g_{k-1} \|^2, \tau_2 \| g_k \| \| d_{k-1} \|\}},$$

$$\beta_k^{WHSFR} = \max\{0, \min\{w_1 \beta_k^{HS1}, w_2 \beta_k^{FR1}\}\}. \tag{15}$$

Step 5. Set

$$\lambda_k = \max\left\{\lambda_{k-1}, -\beta_k \frac{g_k^T d_{k-1}}{\| g_k \|^2} + \theta\right\}, \tag{16}$$

where $\beta_k = \beta_k^{WHSFR}$.

Step 6. Compute the direction $d_k = D(\lambda_k, \beta_k^{WHSFR}, g_k, d_{k-1})$ by (14), (15) and (16). Go to step 1.

Remark 1. *In Step 5, λ_k is updated so that l_k defined by (13) is positive. Indeed, if $\lambda_k = \lambda_{k-1}$ in (16), then we have $\lambda_k \geq -\beta_k \frac{g_k^T d_{k-1}}{\|g_k\|^2} + \theta$, which implies that $l_k = \lambda_k + \beta_k \frac{g_k^T d_{k-1}}{\|g_k\|^2} \geq \theta$, and if $\lambda_k = -\beta_k \frac{g_k^T d_{k-1}}{\|g_k\|^2} + \theta$, we have $l_k = \theta$. Therefore, we have $l_k \geq \theta > 0$ for every k.*

Lemma 1. *The direction d_k generated by (14) satisfies the following sufficient descent property for any CG parameter β_k.*

$$g_k^T d_k \leq -\lambda_0 \| g_k \|^2. \tag{17}$$

Proof. Since $\lambda_0 \leq \lambda_k$ by (16), it follows from (14) that

$$g_k^T d_k = -\left(\lambda_k + \beta_k \frac{g_k^T d_{k-1}}{\| g_k \|^2}\right) \| g_k \|^2 + \beta_k g_k^T d_{k-1} = -\lambda_k \| g_k \|^2 \leq -\lambda_0 \| g_k \|^2.$$

Lemma 2. *In the step 2 of Algorithm 2, the Armijo's backtracking line search finds a step size* t_k *after a finite number of backtracking for every* k.

Proof. Suppose that the Armijo's line search does not terminate after a finite number of backtracking for some k. Then, there exists a k such that

$$f(x_k + \alpha^i d_k) > f_k + \rho \alpha^i g_k^T d_k, \forall i \geq 0,$$

which implies that

$$\frac{f(x_k + \alpha^i d_k) - f_k}{\alpha^i} > \rho g_k^T d_k, \forall i \geq 0.$$

Since $f(x)$ is continuously differentiable, it follows from the above inequality that $g_k^T d_k \geq \rho g_k^T d_k$ as $i \to \infty$. Thus, we have $g_k^T d_k \geq 0$, contradicting to (17).

3. CONVERGENCE ANALYSIS

In this section, under mild assumption, the global convergence of Algorithm 2 is established. We make the following basic assumption.

Assumption 1. *The level set* $L(x_0) = \{x \in R^n \mid f(x) \leq f(x_0)\}$ *is bounded for any given* $x_0 \in R^n$ *and* $f(x)$ *is twice continuously differentiable in a neighbour* $\Re$ *of* $L(x_0)$.

Remark 2. *By Assumption 1, there exists a constant* $M > 0$ *such that*

$$\| \nabla^2 f(x) \| \leq M \quad \forall x \in \Re. \tag{18}$$

Now, it is well known that if the Hessian matrix is bounded by M, *then the gradient is Lipschitz continuous with Lipschitz constant* M. *Therefore, we have*

$$\| g(x) - g(y) \| \leq M \| x - y \| \forall x, y \in \Re,$$

which implies that there exists a constant $\gamma > 0$ such that

$$\| g(x) \| \leq \gamma \quad \forall x \in \Re. \tag{19}$$

Lemma 3. *Suppose that Assumption 1 holds and* d_k *is generated by Algorithm 2. Then the step size* t_k *generated by Algorithm 1 satisfies*

$$t_k > \min\{\frac{\alpha}{2}, -C \frac{g_k^T d_k}{\| d_k \|^2}\}, \tag{20}$$

where $C = \frac{2\alpha(1-\rho)}{M}$.

Proof. If $t_k > \frac{\alpha}{2}$, (20) is obvious. Therefore, consider the case of $t_k \leq \frac{\alpha}{2}$. From Step 2 and Step 3 of Algorithm 1, we have

$$f\left(x_k + \frac{t_k}{\alpha} d_k\right) > f_k + \rho \frac{t_k}{\alpha} g_k^T d_k. \tag{21}$$

Using Taylor expansion, we get

$$f\left(x_k + \frac{t_k}{\alpha} d_k\right) = f_k + \frac{t_k}{\alpha} g_k^T d_k + \frac{1}{2}\left(\frac{t_k}{\alpha}\right)^2 d_k^T \nabla^2 f(\xi_k) d_k, \tag{22}$$

where $\xi_k \in (x_k, x_k + \frac{t_k}{\alpha} d_k)$. From Assumption 1, (18), (21) and (22), we obtain

$$f_k + \rho \frac{t_k}{\alpha} g_k^T d_k < f_k + \frac{t_k}{\alpha} g_k^T d_k + \frac{1}{2}\left(\frac{t_k}{\alpha}\right)^2 M \parallel d_k \parallel^2.$$

Therefore, we have $-(1-\rho) g_k^T d_k < \frac{1}{2}\frac{t_k}{\alpha} M \parallel d_k \parallel^2$, which implies that

$$t_k > -C \frac{g_k^T d_k}{\parallel d_k \parallel^2}. \tag{23}$$

Thus, we have (20).

From (17), $g_k \neq 0$ implies that $g_k^T d_k \neq 0$. Therefore (20) indicates that the steplength t_k obtained by the Armijo's backtracking line search is bounded away from zero.

Theorem 1. *Suppose that Assumption 1 holds and the sequence* $\{x_k\}$ *is generated by Algorithm 2. Then, it holds that*

$$\sum_{k=0}^{\infty} \frac{(g_k^T d_k)^2}{\parallel d_k \parallel^2} < +\infty. \tag{24}$$

Proof. First, let's consider the case of $t_k \leq \frac{\alpha}{2}$. By (23), there is a constant $C > 0$ such that

$$t_k > -C \frac{g_k^T d_k}{\parallel d_k \parallel^2}. \tag{25}$$

Then, it follows from (7) and (25) that

$$f_k - f(x_{k+1}) \geq -\rho t_k g_k^T d_k \geq \rho C \frac{(g_k^T d_k)^2}{\parallel d_k \parallel^2},$$

which implies that

$$\frac{(g_k^T d_k)^2}{\parallel d_k \parallel^2} \leq \frac{1}{\rho C}(f_k - f(x_{k+1})). \tag{26}$$

Next, let's consider the case of $t_k > \frac{\alpha}{2}$. Then, there are two cases: $\frac{\alpha}{2} \geq -C\frac{g_k^T d_k}{\|d_k\|^2}$ and $\frac{\alpha}{2} < -C\frac{g_k^T d_k}{\|d_k\|^2}$. In the former case, (25) holds and we have (26). Now, let's consider the latter case. If $t_k > -C\frac{g_k^T d_k}{\|d_k\|^2}$, it holds (25) and then we have again (26). Therefore, let's consider the case of $\frac{\alpha}{2} < t_k \leq -C\frac{g_k^T d_k}{\|d_k\|^2}$. In this case, $-g_k^T d_k \geq \frac{1}{C} t_k \parallel d_k \parallel^2$ and then, by (7), we have

$$f_k - f(x_{k+1}) \geq -\rho t_k g_k^T d_k > \rho \frac{t_k^2}{C} \parallel d_k \parallel^2 \geq \frac{\rho \alpha^2}{4C} \parallel d_k \parallel^2.$$

By multiplying $\frac{(g_k^T d_k)^2}{\|d_k\|^4}$ to both sides of the above inequality, we have

$$\frac{\rho \alpha^2}{4C} \frac{(g_k^T d_k)^2}{\parallel d_k \parallel^2} \leq \frac{(g_k^T d_k)^2}{\parallel d_k \parallel^4}(f_k - f(x_{k+1})).$$

Since (17) implies that $\lambda_0 \parallel g_k \parallel \leq \parallel d_k \parallel$, from the above inequality we have

$$\frac{(g_k^T d_k)^2}{\parallel d_k \parallel^2} \leq \frac{4C}{\rho \alpha^2} \frac{\parallel g_k \parallel^2 \parallel d_k \parallel^2}{\parallel d_k \parallel^4}(f_k - f(x_{k+1}))$$

$$\leq \frac{4C}{\rho \alpha^2} \frac{\parallel g_k \parallel^2}{\parallel d_k \parallel^2}(f_k - f(x_{k+1})) \leq \frac{4C}{\rho \alpha^2 \lambda_0^2}(f_k - f(x_{k+1})).$$

Therefore, letting $M_2 = \min\{\frac{1}{\rho C}, \frac{4C}{\rho\alpha^2\lambda_0^2}\}$, it follows from (26) and the above inequality that

$$\frac{(g_k^T d_k)^2}{\| d_k \|^2} \leq M_2(f_k - f(x_{k+1})).$$

Since the sequence $\{f(x_k)\}$ is monotonically decreasing by (7) and bounded from below, it has a limit, i.e. $\lim_{k\to\infty} f(x_k) = f^*$. Thus, we have

$$\sum_{k=0}^{\infty} \frac{(g_k^T d_k)^2}{\|d_k\|^2} \leq M_2 \sum_{k=0}^{\infty}(f_k - f(x_{k+1})) = M_2 \lim_{k\to\infty}(f(x_0) - f(x_{k+1}) = M_2(f(x_0) - f^*) < \infty,$$

which proves (24).

Theorem 2. *Suppose that Assumption 1 holds and let* $\{x_k\}$ *be generated by Algorithm 2. Then, there exists* $\bar{\lambda}$ *such that* $\lambda_k \leq \bar{\lambda}$ *for all* k, *and* $\{x_k\}$ *is bounded and satisfies*

$$\lim_{k\to\infty} \inf \| g_k \| = 0. \tag{27}$$

Proof. By Armijo condition (7), $\{x_k\} \in L(x_0)$ and then $\{x_k\}$ is bounded from Assumption 1. Suppose that (27) does not hold. Then, there exists a constant $\eta > 0$ such that

$$\| g_k \| \geq \eta, \forall k \geq 0. \tag{28}$$

By (15), we have

$$\beta_k^{WHSFR} = \max\{0, \min\{w_1\beta_k^{HS1}, w_2\beta_k^{FR1}\}\} \leq w_2\beta_k^{FR1}, \tag{29}$$

which gives $\lambda_k \leq \bar{\lambda} = \frac{w_2}{\tau_2} + \theta$ together with (15).

Let's estimate an upper bound of l_k defined in (13). By (15), (29), Cauchy- Schwartz inequality and the assumption, we have

$$\begin{aligned} | l_k | &= | \lambda_k + \beta_k \frac{g_k^T d_{k-1}}{\| g_k \|^2} | \\ &\leq | \lambda_k | + w_2\beta_k^{FR1} | \frac{g_k^T d_{k-1}}{\| g_k \|^2} | \\ &\leq \bar{\lambda} + \frac{w_2 \| g_k \|^2}{max\{| g_{k-1} \|^2, \tau_2 \| g_k \| \| d_{k-1} \|\}} \frac{\| g_k \| \| d_{k-1} \|}{\| g_k \|^2} \\ &\leq \bar{\lambda} + \frac{w_2 \| g_k \|^2}{\tau_2 \| g_k \| \| d_{k-1} \|} \frac{\| g_k \| \| d_{k-1} \|}{\| g_k \|^2} \leq \bar{\lambda} + \frac{w_2}{\tau_2}. \end{aligned} \tag{30}$$

From (14), (30), (29) and (19), it follows that

$$\begin{aligned} \| d_k \| &\leq | l_k | \| g_k \| + w_2\beta_k^{FR1} \| d_{k-1} \| \\ &\leq \left(\bar{\lambda} + \frac{w_2}{\tau_2}\right)\gamma + \frac{w_2 \| g_k \|^2}{\tau_2 \| g_k \| \| d_{k-1} \|} \| d_{k-1} \| \\ &\leq \left(\bar{\lambda} + \frac{w_2}{\tau_2}\right)\gamma + \frac{w_2}{\tau_2}\gamma = \left(\bar{\lambda} + \frac{2w_2}{\tau_2}\right)\gamma. \end{aligned} \tag{31}$$

Thus, from (31), (17) and (28), it follows that

$$\sum_{k=0}^{\infty} \frac{(g_k^T d_k)^2}{\| d_k \|^2} \geq \sum_{k=0}^{\infty} \frac{(g_k^T d_k)^2}{\left(\left(\bar{\lambda} + \frac{2w_2}{\tau_2}\right)\gamma\right)^2} \geq \sum_{k=0}^{\infty} \frac{(\lambda_0 \| g_k \|^2)^2}{\left(\left(\bar{\lambda} + \frac{2w_2}{\tau_2}\right)\gamma\right)^2} \geq \sum_{k=0}^{\infty} \frac{\lambda_0^2\eta^4}{\left(\left(\bar{\lambda} + \frac{2w_2}{\tau_2}\right)\gamma\right)^2} = \infty,$$

which contradicts (24).

4. NUMERICAL EXPERIMENTS

This section shows preliminary numerical experiment results to analyze the feasibility and effectiveness of Algorithm 2. The CG algorithms NACG [6], CGADP1, CGADP2, CGADP+ [3], EPF [5], MLSCD [2], HYBRID [7], JHJ-N [8] and our algorithm AWHCG were coded in Matlab programming language. The algorithms are tested on 19 nonlinear unconstrained optimization problems [1,25,26,27,29,30] with different dimensions (see Table 1 and Table 2). All tests are performed on a CPU 1.33GHz and RAM 1.86 GB personal computer using Matlab R2013a.8.1.0.604. The algorithms CGADPs, HYBRID, NACG and EPF implemented the Wolfe line search and the strong Wolfe line search by the algorithm presented in Sect.2.6 of [11] with $\rho = 0.01$ and $\sigma = 0.1$, respectively. In numerical experiments of HJH-N, $\rho = 0.0001$ and $\sigma = 0.9$ were taken for the Wolfe line search. The algorithms MLSCD and AWHCG implemented the Algorithm 1 (Armijo's line search) with $\rho = 0.008$ and $\alpha = 0.5$. The stopping criteria we use for all tested algorithms are $\| g_k \| \leq 10^{-5}$ or $\frac{|f_{k+1} - f_k|}{1+|f_k|} \leq 10^{-12}$. In AWHCG, the weight of β_k^{HS1} and β_k^{FR1} parameters is chosen as $w_1 = 4$ and $w_2 = 1$, respectively, and $\theta = 1, \tau_1 = 0.01$ and $\tau_2 = 0.02$ are chosen.

Table 1. Test problems and its optimal values

No	Objective function	Optimal value
1	$\frac{\pi}{n}\left\{10\sin^2(\pi x_1) + (x_n - 1)^2 + \sum_{i=1}^{n-1}[(x_i - 1)^2(1 + 10\sin^2(\pi x_{i+1}))]\right\}$	0
2	$-20\exp(-0.2\sqrt{\| x \|^2/n} - \exp(\sum_{i=1}^{n}\cos(2\pi x_i/n)) + 20 + e$	0
3	$\sum_{i=1}^{n} x_i^2 + 10n - 10\sum_{i=1}^{n}\cos(2\pi x_i)$	0
4	$20 + e - 20\exp(-0.2\sqrt{\| x \|^2/n}) - \exp(\sum_{i=1}^{n}\cos(\pi x_i/n))$	0
5	$\sum_{i=1}^{n}[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2]$	0
6	$100(x_2 - x_1)^2 + (1 - x_1)^2$	0
7	$\sum_{i=2}^{n-1}[(3 - 2x_i)x_i - x_{i-1} - 2x_{i+1} + 1]^2$	0
8	$100(x_1^2 - x_2)^2 + (x_1 - 1)^2 + (x_3 - 1)^2 + 90(x_3^2 - x_4)^2 + 10.1((x_2 - 1)^2 + (x_4 - 1)^2) + 19.8(x_2 - 1)(x_4 - 1)$	0
9	$4x_1^2 - 2.1x_1^4 + x_1^6/3 - x_1x_2 - 4x_2^2 + 4x_2^4$	-1.0316
10	$(1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2))/(30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2))$	3
11	$\sum_{i=1}^{n/4}[(x_{4i-1} + 10x_{4i-2})^2 + 5(x_{4i-2} - x_{4i})^2 + (x_{4i-2} - 2x_{4i-1})^2 + 10(x_{4i-3} - x_{4i})^2]$	0
12	$100[(x_3 - 10\theta)^2 + (\sqrt{x_1^2 + x_2^2} - 1)^2] + x_3$, if $x_1 > \epsilon, \theta = \arctan(x_1/x_2)/(2\pi)$; if $x_1 < -\epsilon, \theta = 0.5 + \arctan(x_1/x_2)/(2\pi)$; if $\| x_1 \| \leq \epsilon, \theta = \epsilon$, where $\epsilon = 10^{-5}$.	0

13	$\sum_{i=1}^{n}[n+i(1-\cos x_i)-\sin x_i-\sum_{j=1}^{n}\cos x_j]^2$	0
14	$\left(\sum_{i=1}^{5} i\,(\cos[(i+1)x_i+i])\right)\left(\sum_{i=1}^{5} i\,(\cos[(i+1)x_2+i])\right)$	-186.7309
15	$\sum_{i=1}^{n/2}\{[-13+x_{2i-1}+((5-x_{2i})x_{2i}-1)x_{2i}]^2+[-29+x_{2i-1}+((x_{2i}+1)x_{2i}-14)x_{2i}]^2\}$	0
16	$\sum_{i=1}^{n/10}[(1-x_{10i-9})^2+(1-x_{10i})^2+\sum_{j=10i-9}^{10i-1}(x_j^2-x_{j+1})^2]$	0
17	$\sum_{i=1}^{n/2}[(1.5-x_{2i-1}(1-x_{2i}))^2+(2.25-x_{2i-1}(1-x_{2i}))^2+(1-x_{2i})(1-x_{2i})^2]$	0
18	$\sum_{i=1}^{n-1}(i\,x_i)^2$	$-(n-1)$
19	$\sum_{i=1}^{n-1}\cos(x_i^2-0.5x_{i+1}), n=1000,2000,\ldots$	$-(n-1)$

Table 2. Names, dimensions and initial points of test problems

No	Name	dimensions	x_0
1	n-Dimensions Sine-Square II [28,30]	100, 1000, 2000, 3000	$2.5(1,\ldots,1)^T$
2	Ackley [27,30]	1000, 2000, 3000, 4000	$0.5(1,\ldots,1)^T$
3	Rastrigin [27,28,30]	1000, 3000, 5000	$(1,\ldots,1)^T$
4	Shekel [27]	1000, 2000, 3000	$(1,\ldots,1)^T$
5	Extended Rosenbrock [1,28,29, 30]	100, 500, 1000, 2000	$(-1.2,1,\ldots,-1.2,1)^T$
6	Cube [1]	2	$(-1.2,-1)^T$
7	Broyden Tridiagonal [26]	1000, 3000	$-(1,\ldots,1)^T$
8	Wood [1,28,30]	4	$(-3,-1,-3,-1)^T$
9	Six-hump Camel Back [27,28,30]	2	$(-1.2,-1)^T$
10	Goldstein Price [27,28,30]	2	$(-1,-2)^T$
11	Extended Powell-Singular [26]	1000, 3000	$(3,-1,0,1,\ldots,3,-1,0,1)^T$
12	Hellical Valley [1,30]	3	$(-1,0,0)^T$
13	Trigonometric [1,26,27]	2000, 3000, 5000	$(1,\ldots,1)^T/5n$
14	Shubert [30]	2	$(-1.5,-1)^T$
15	Extended Freudenstein-Roth [29]	500, 1000	$(20,-1,\ldots,20,-1)^T$
16	Extended Dixson [26]	1000, 3000	$-2(1,\ldots,1)^T$
17	Extended Beale [26]	1000, 2000, 3000	$(1,\ldots,1)^T$
18	Power [25]	100, 500, 1000	$(1,\ldots,1)^T$
19	Cosine [25]	1000, 2000, 3000, 4000	$(1,\ldots,1)^T$

In the tables below, “iter” is the number of iterations needed during the algorithms’ work, “time” is the CPU time, x0 is the initial point, and “-” means that the algorithm did not stop within CPU time of 1000 seconds. In order to show the effectiveness of our algorithm AWHCG, some numerical results for the examples 1 and 3 are given in Table 3, Table 4 and Table 5.

Table 3. Numerical results for Example 1

$x_0 = 2.5(1,\ldots,1)^T$, $n = 100$			
Method	f_k	iter	time(s)
AWHCG, $\lambda_0 = 0.3$	8.5745e-11	47	0.2309
$\lambda_0 = 0.5$	1.3580e-10	33	0.1862
$\lambda_0 = 0.6$	1.0225e-10	47	0.2336
$\lambda_0 = 0.7$	1.0922e-10	41	0.2174
$\lambda_0 = 0.8$	2.4498e-10	38	0.1935
$\lambda_0 = 1.0$	3.3073e-12	48	0.2363
$\lambda_0 = 1.1$	1.1132e-11	52	0.2530
$\lambda_0 = 2.1$	6.2398e-10	49	0.2412
$\lambda_0 = 2.5$	9.5857e-11	40	0.1957
$\lambda_0 = 3.5$	2.2343e-10	48	0.2384
MLSCD	6.0694e-11	48	0.2634
CGADP+	4.5292e-14	49	0.9002
CGADP1	4.4597e-10	37	0.7615
CGADP2	5.1050	313	30.7382
EPF	8.5261e-15	15	0.3498
NACG	1.3703e-10	20	0.3703
HYBRID	1.2105e-10	58	1.0935
JHJ-N	0.0311	31	0.6946

Table 4. Numerical results for Example 1

$x_0 = 2.5(1,\ldots,1)^T$, $n = 2000$			
Method	f_k	iter	time(s)
AWHCG, $\lambda_0 = 2.3$	3.1094	16	10.5489
$\lambda_0 = 7$	3.1094	8	5.3692
$\lambda_0 = 13$	0.0016	171	102.5382
$\lambda_0 = 14$	1.2314e-08	144	89.0191
$\lambda_0 = 15$	8.4729e-09	137	80.1580
$\lambda_0 = 16$	1.0671e-08	149	90.1271
$\lambda_0 = 17$	7.4723e-09	162	97.4797
$\lambda_0 = 18$	8.3895e-09	156	95.1742
$\lambda_0 = 19$	0.0016	171	102.1295
$\lambda_0 = 20$	0.0078	733	494.5868
MLSCD	3.1094	39	25.0244
CGADP+	0.0140	115	345.0503
CGADP1	0.0140	70	206.2090
CGADP2	-	-	-

EPF	0.0154	48	142.0608
NACG	12.5344	10	26.1220
HYBRID	12.5344	11	42.2331
JHJ-N	12.5344	24	72.9036

We can see from Table 3 and Table 4 that the methods CGADP+, CGADP1, EPF , NACG, JHJ-N and MLSCD have good performance for Example 1 in case of n = 100 and all methods except AWHCG failed in case of n = 2000. From numerical experiments for the examples 1, 2, 4, 5, 8, 9, 10, 15 and 18, we can see that AWHCG can succeed in finding global optimum even though the other CG methods failed to solve the problems. And Table 5 shows that AWHCG is about twice as fast as the other methods except MLSCD for Example 3.

Table 5. Numerical results for Example 3

$x_0 = 2.5(1,\ldots,1)^T$, $n = 5000$			
Method	f_k	iter	time(s)
AWHCG, $\lambda_0 = 2.1$	0	3	9.2174
$\lambda_0 = 2.3$	0	3	9.2315
$\lambda_0 = 2.5$	0	2	7.9061
$\lambda_0 = 2.6$	0	2	7.7688
$\lambda_0 = 2.7$	0	2	7.1114
$\lambda_0 = 2.8$	0	3	9.2472
$\lambda_0 = 2.9$	0	3	9.1354
$\lambda_0 = 3.0$	0	3	9.2246
$\lambda_0 = 3.2$	7.2760e-12	2	7.4651
$\lambda_0 = 3.3$	7.2760e-12	2	7.1033
$\lambda_0 = 3.4$	7.2760e-12	2	7.2712
$\lambda_0 = 3.5$	7.2760e-12	2	7.2357
$\lambda_0 = 3.6$	7.2760e-12	2	7.2665
$\lambda_0 = 3.7$	7.2760e-12	2	7.3952
MLSCD	0	3	9.2354
CGADP+	0	2	19.2340
CGADP1	0	2	19.2378
CGADP2	0	2	19.2153
EPF	0	2	19.1442
NACG	1.2490e-06	3	19.5484
HYBRID	0	2	19.3730
JHJ-N	0	2	19.5134

If a method failed to find the global optimum or did not terminate within CPU time of 1000 seconds for a test, we regard the method as failed. In Table 6 and Table 7, total number of iterations and total CPU

times for all tests are given, in which contents of "iter" and "time" for failed methods were replaced by sufficiently large number. Number of iterations and CPU time of AWHCG for each example is mean of iteration numbers and CPU times corresponding to different values of λ, respectively.

Table 6. Total number of iterations for the examples 1-19

Example	AWHCG	MLSCD	CGADP+	CGADP1	CGADP2	EPF	NACG	HYBRID	JHJ-N
1	**716.2**	1134	1127	1136	1168	1101	1101	1182	1196
2	**147**	752	1220	1220	1220	1220	1220	1220	1220
3	6.4	9	**6**	**6**	**6**	**6**	9	**6**	**6**
4	**186**	1148	2100	2100	2100	2100	2100	2100	2100
5	**16072**	23515	31000	31000	31000	31000	31000	31000	31000
6	82	71	200	**45**	74	1000	1000	38	113
7	113.16	95	64	**50**	**50**	433	61	53	57
8	164	**120**	1000	1000	1000	1000	252	183	603
9	**55.67**	150	150	150	150	150	150	150	150
10	**32.0**	88	200	200	200	200	200	200	200
11	153	147	44	40	46	**11**	300	135	262
12	58.5	86	2400	55	**32**	2400	67	50	57
13	**12**	**12**	93	24	96	120	16	13	22
14	**10**	**10**	20	20	12	**10**	20	16	12
15	**1047**	12000	12000	12000	12000	12000	12000	12000	12000
16	**137**	215	249	175	245	254	230	247	342
17	212	148	57	57	51	300	45	**41**	105
18	**15850**	19294	25000	25000	25000	25000	25000	25000	25000
19	112	260	46	39	43	**33**	44	73	49

Table 7. Total CPU times for the examples 1-19

Example	AWHCG	MLSCD	CGADP+	CGADP1	CGADP2	EPF	NACG	HYBRID	JHJ-N
1	**536.63**	918.01	918.32	918.85	919.85	918.09	918.08	921.47	928.23
2	**133.76**	770.07	840	840	840	840	840	840	840
3	**11.52**	13.67	26.58	26.51	26.49	26.42	27.03	26.47	26.64
4	**107.97**	742.52	1410	1410	1410	1410	1410	1410	1410
5	**2077.30**	3611	6257	6257	6257	6257	6257	6257	6257
6	0.0215	**0.0121**	2	0.0588	0.0652	6	2	0.0399	0.0658
7	114.57	**91.35**	202.85	185.82	192.57	1696.50	237.49	192.95	194.36
8	0.0304	**0.0201**	5.6	5.6	5.6	5.6	-	0.127	0.30
9	**0.010**	0.12	0.12	0.12	0.12	-	0.12	0.12	0.12
10	**0.0082**	0.0136	0.06	0.06	0.06	0.06	-	0.06	0.06
11	66.04	59.31	74.01	64.62	81.19	**18.50**	800	227.19	544.84
12	**0.0164**	0.0207	0.2	0.0418	0.0256	73.14	0.0688	0.0406	0.0468
13	34.41	**31.65**	1074.29	207.18	1074.96	1250	68.07	65.38	73.74

14	**0.0085**	**0.0085**	0.03	0.03	0.0106	0.0103	0.03	0.021	0.0105
15	**152.22**	500	500	500	500	500	500	500	500
16	**82.62**	131.27	680.54	515.43	688.23	694.33	680.46	695.68	683.24
17	358.52	**239.18**	479.48	490.92	458.52	800	377.41	340.01	727.03
18	**1060.70**	1096.95	2560	2560	2560	2560	2560	2560	2560
19	**154.65**	377.31	233.72	199.30	190.58	178.52	247.75	431.86	280.13

Based on Table 6 and Table 7, the performance profiles [24] for iteration numbers and CPU time were constructed to compare the efficiency of algorithms used in the computational experiments (see Figure 1).

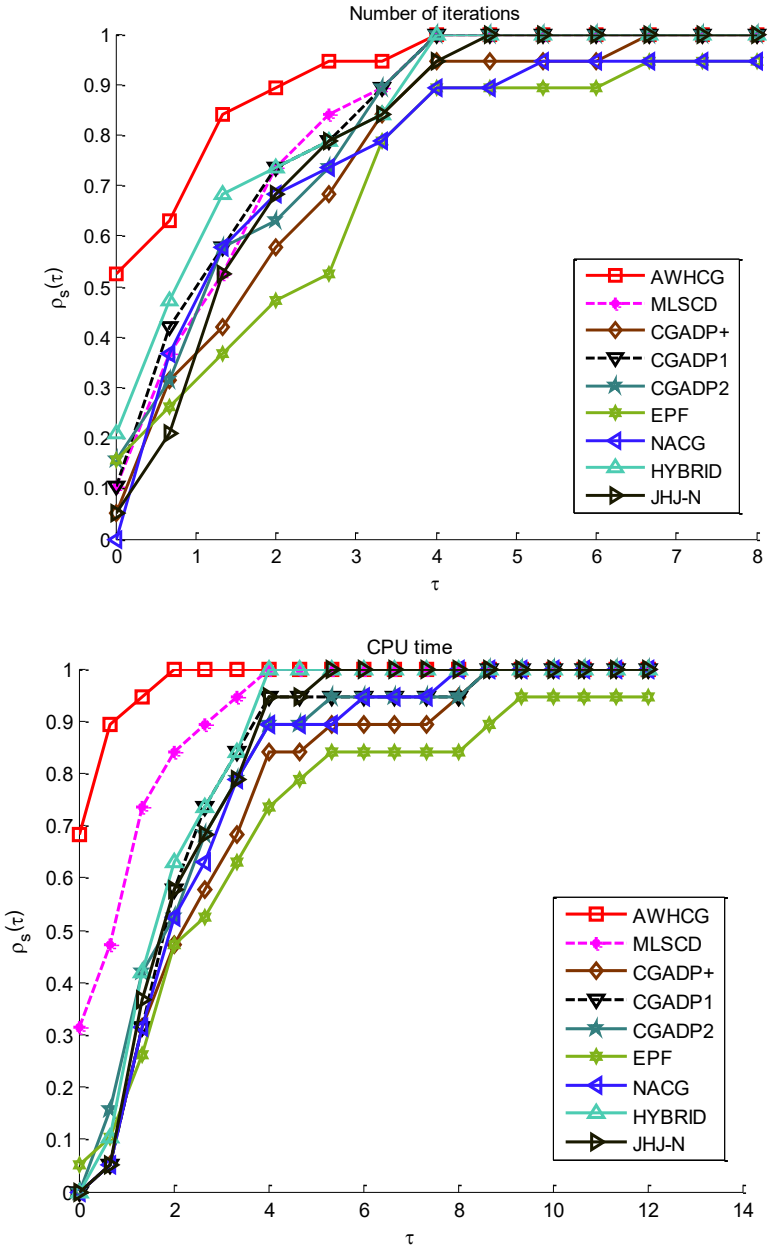

Figure 1. Performance proles for iteration numbers and CPU times of tested algorithms

Let S be the set of n_s solvers in comparison and P be the set of n_p test problems. For each problem $p \in P$ and solver $s \in S$, let $f_{p,s}$ denote either number of iterations or CPU time required to solve problem $p \in P$ by solver $s \in S$. Then comparison of the solvers is based on the performance ratio

$$r_{p,s} = \frac{f_{p,s}}{\min\{f_{p,s} \mid s \in S, p \in P\}}.$$

The overall assessment of the performance of the solvers is then obtained from a performance profile function given by

$$\rho_s(\tau) = \frac{1}{n_p} \mid \{p \mid 1 \leq p \leq n_p, \ln(r_{p,s}) \leq \tau\} \mid,$$

where $\tau \geq 0$.

It is obvious from the figure that AWHCG is competitive with MLSCD, CGADPs, EPF, HYBRID, JHJ-N and NACG. This observation shows that the new adaptive hybrid CG method based on the weighted hybridization of appropriate CG parameters (e.g. HS and FR) can achieve a better performance than other hybrid CG methods.

5. CONCLUSION

We propose a new adaptive hybrid conjugate gradient (AWHCG) method for unconstrained optimization problems, in which the search direction always satisfy the sufficient descent condition. This method is based on replacing the CG direction dk of the form (12) by the new direction (14) with an adaptive parameter and weighted hybrids of modified HS parameter and FR parameter. The adaptive parameter is introduced to make the coefficient of the gradient g_k to be negative like as the standard CG method. Based on the backtracking line search conditions, we show that our method has global convergence for general functions under mild assumptions. Criteria for comparing CG methods are the number of iterative steps and spent CPU time. Numerical results illustrate that the proposed method with proper choice of parameters λ_0, θ, w_1 and w_2 can outperform the existing CG methods. We believe that further research on how to adaptively choose the appropriate weights of CG parameters will be necessary and worthwhile.

Acknowledgements: The authors would like to thank the editor Lynne Hsieh and the anonymous reviewers for careful consideration of this paper.

Data Availability Statement: The authors declare that all data in the paper are available.

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

Funding Statement: This study was not funded by any external sources.

Informed Consent Statement: Not applicable.

REFERENCES

[1] Sun, W., & Yuan, Y. X. (2006). *Optimization theory and methods: Nonlinear programming*. Springer.

[2] Stanimirović, P. S., Ivanov, B., Djordjević, S., & Brajević, I. (2018). New hybrid conjugate gradient and Broyden-Fletcher-Goldfarb-Shanno conjugate gradient methods. *Journal of Optimization Theory and Applications*, *178*, 860-884.

[3] Dong, X. L., Dai, Z. F., Ghanbari, R., & Li, X. L. (2019). An adaptive three-term conjugate gradient method with sufficient descent condition and conjugacy condition. *Journal of the Operations Research Society of China*. https://doi.org/10.1007/s40305-019-00257-w

[4] Zheng, Y., & Zheng, B. (2017). Two new Dai-Liao-type conjugate gradient methods for unconstrained optimization problems. *Journal of Optimization Theory and Applications*, *175*, 502-509.

[5] Mtagulwa, P., & Kaelo, P. (2019). An efficient modified PRP-FR hybrid conjugate gradient method for solving unconstrained optimization problems. *Applied Numerical Mathematics*, *145*, 111-120.

[6] Chen, Y., Cao, M., & Yanget, Y. (2019). A new accelerated conjugate gradient method for large-scale unconstrained optimization. *Journal of Inequalities and Applications*, *2019*, Article 300.

[7] Zheng, X., Dong, X., Shi, J., & Yang, W. (2020). Further comment on another hybrid conjugate gradient algorithm for unconstrained optimization by Andrei. *Numerical Algorithms*, *84*, 603-608.

[8] Jian, J., Han, L., & Jiang, X. (2015). A hybrid conjugate gradient method with descent property for unconstrained optimization. *Applied Mathematical Modelling*, *39*, 1281-1290.

[9] Andrei, N. (2013). A simple three-term conjugate gradient algorithm for unconstrained optimization. *Journal of Computational and Applied Mathematics*, *241*, 19-29.

[10] Fletcher, R., & Reeves, C. M. (1964). Function minimization by conjugate gradients. *The Computer Journal*, *7*(2), 149-154.

[11] Fletcher, R. (1987). *Practical methods of optimization: Unconstrained optimization* (Vol. 1). Wiley.

[12] Dai, Y. H., & Yuan, Y. (1999). A nonlinear conjugate gradient method with a strong global convergence property. *SIAM Journal on Optimization*, *10*(1), 177-182.

[13] Hestenes, M. R., & Stiefel, E. L. (1952). Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, *49*(6), 409-436.

[14] Polak, E., & Ribière, G. (1969). Note sur la convergence des méthodes de directions conjuguées. *Revue Française d'Informatique et de Recherche Opérationnelle*, *16*, 35-43.

[15] Polyak, B. T. (1969). The conjugate gradient method in extreme problems. *U.S.S.R. Computational Mathematics and Mathematical Physics*, *9*, 94-112.

[16] Liu, Y., & Storey, C. (1991). Efficient generalized conjugate gradient algorithms, part 1: theory. *Journal of Optimization Theory and Applications*, *69*(1), 129-137.

[17] Yang, X., Luo, Z., & Dai, X. (2013). A global convergence of LS-CD hybrid conjugate gradient method. *Advances in Numerical Analysis*, *2013*, Article 517452.

[18] Dai, Y. H., & Liao, L. Z. (2001). New conjugacy conditions and related nonlinear conjugate gradient methods. *Applied Mathematics and Optimization*, *43*(1), 87-101.

[19] Touati-Ahmed, D., & Storey, C. (1990). Efficient hybrid conjugate gradient techniques. *Journal of Optimization Theory and Applications*, *64*(2), 379-397.

[20] Zhang, L., & Zhou, W. (2008). Two descent hybrid conjugate gradient methods for optimization. *Journal of Computational and Applied Mathematics*, *216*, 251-264.

[21] Wei, Z. X., Yao, S. W., & Liu, L. Y. (2006). The convergence properties of some new conjugate gradient methods. *Applied Mathematics and Computation*, *183*, 1341-1350.

[22] Dai, Y. H., & Yuan, Y. (2001). An efficient hybrid conjugate gradient method for unconstrained optimization. *Annals of Operations Research*, *103*, 33-47.

[23] Yao, S. W., Wei, Z. X., & Huang, H. (2007). A note about WYL's conjugate gradient method and its application. *Applied Mathematics and Computation*, *191*, 381-388.

[24] Dolan, E. D., & Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming*, *91*, 201-213.

[25] Alkhazishvili, L., & Gorgadze, L. (2019). *A collection of test functions for unconstrained optimization solvers with serial and parallel C++ implementations* [Preprint]. ResearchGate. https://www.researchgate.net/publication/334226455

[26] Liu, J., & Ma, C. (2013). A nonmonotone trust region method with new inexact line search for unconstrained optimization. *Numerical Algorithms*, *64*, 1-20.

[27] Peng, Z., Wu, D., & Zheng, Q. (2013). A level-value estimation method and stochastic implementation for global optimization. *Journal of Optimization Theory and Applications*, *156*, 493-523.

[28] Wang, C., Liu, K., & Shen, P. (2020). A novel genetic algorithm for global optimization. *Acta Mathematicae Applicatae Sinica (English Series)*, *36*(2), 482-491.

[29] Hu, P., & Liu, X. Q. (2013). A nonmonotone line search slackness technique for unconstrained optimization. *Journal of Optimization Theory and Applications*, *158*, 773-786.

[30] Ali, M. M., Khompatraporn, C., & Zabinsky, Z. B. (2005). A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. *Journal of Global Optimization*, *31*, 635-672.