J Optim Theory Appl (2018) 178:860–884
https://doi.org/10.1007/s10957-018-1324-3

New Hybrid Conjugate Gradient and Broyden–Fletcher–Goldfarb–Shanno Conjugate Gradient Methods

Predrag S. Stanimirović[1] · **Branislav Ivanov**[2] · **Snežana Djordjević**[3] · **Ivona Brajević**[1]

Received: 5 March 2018 / Accepted: 26 May 2018 / Published online: 11 June 2018

Abstract Three hybrid methods for solving unconstrained optimization problems are introduced. These methods are defined using proper combinations of the search directions and included parameters in conjugate gradient and quasi-Newton methods. The convergence of proposed methods with the underlying backtracking line search is analyzed for general objective functions and particularly for uniformly convex objective functions. Numerical experiments show the superiority of the proposed methods with respect to some existing methods in view of the Dolan and Moré's performance profile.

Keywords Global convergence · Backtracking line search · Unconstrained optimization · Conjugate gradient methods · Quasi-Newton methods

Mathematics Subject Classification 90C30

Communicated by Ilio Galligani.

✉ Predrag S. Stanimirović
pecko@pmf.ni.ac.rs

Branislav Ivanov
ivanov.branislav@gmail.com

Snežana Djordjević
snezanadjordjevicle@gmail.com

Ivona Brajević
ivona.brajevic@googlemail.com

[1] Faculty of Sciences and Mathematics, University of Niš, Višegradska 33, 18000 Nis, Serbia

[2] Technical Faculty in Bor, University of Belgrade, Vojske Jugoslavije 12, 19210 Bor, Serbia

[3] Faculty of Technology in Leskovac, University of Niš, Bulevar Oslobodjenja 124, 16000 Leskovac, Serbia

1 Introduction

We study the conjugate gradient methods for unconstrained optimization problems. It is well known that Fletcher–Reeves (*FR*) [1], Conjugate Descent (*CD*) [2] and Dai–Yuan (*DY*) [3] conjugate gradient methods have strong convergence properties, but they may not perform well in practice due to jamming. On the other hand, Hestenes–Stiefel (*HS*) [4], Polak–Ribiére–Polyak (*PRP*) [5,6], and Liu–Storey (*LS*) [7] conjugate gradient methods may not converge in general, but they often perform better than *FR*, *CD* and *DY*. To combine the best numerical performances of the *LS* method and the global convergence properties of the *CD* method, Yang et al. [8] proposed a hybrid *LS-CD* method. Dai and Liao [9] proposed an efficient conjugate gradient-type method (Dai–Liao method). Later, some more efficient Dai–Liao-type conjugate gradient methods, known as *DHSDL* and *DLSDL*, were designed and studied in [10].

Continuing previous results, we propose two efficient hybridizations of conjugate gradient (*CG*) methods for solving unconstrained optimization problems and a hybridization of the Broyden–Fletcher–Goldfarb–Shanno (*BFGS*) method with the *CG* methods. The convergence analysis of the proposed methods is considered. Numerical experiments show that our methods outperform the existing ones from [10].

The rest of the paper is organized as follows. In Sect. 2, we elaborate various possibilities to determine the step size and the search direction which can be used in defining various conjugate gradient methods and their combinations. Here, a hybridization of the *CG* method and the *BFGS* method will also be presented. A modification of the *LSCD* method from [8], termed as *MLSCD*, is proposed in Sect. 3. Also, the global convergence of the *MLSCD* method for non-convex functions with the backtracking line search is approved. In Sect. 4, we combine conjugate gradient parameters of *DHSDL* and *DLSDL* methods from [10] and propose the modification of these methods, termed as *MMDL*. The global convergence of the *MMDL* method, supported by the backtracking line search, is proved. A new hybrid *BFGS-CG* search direction, that combines the search direction of the quasi-Newton *BFGS* method and *CG* methods, is proposed in Sect. 5. The method will be termed as *H-BFGS-CG*1. The global convergence of the *H-BFGS-CG*1 method with the backtracking line search is proved. In Sect. 6, we report some numerical results and compare the performance of the proposed methods with some existing methods. Some final conclusions are given in the last section.

2 Preliminaries

Unconstrained optimization problems consider the problem of minimizing an objective function $f : \mathbb{R}^n \to \mathbb{R}$

$$\min f(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^n, \tag{1}$$

that depends on real variables $\mathbf{x} = \{x_1, \ldots, x_n\}$ without any restrictions on their values. There are many methods for solving (1). Usually, the objective f is a continuously differentiable function with its gradient $\mathbf{g}$. The general CG iterative scheme is given by

$$\mathbf{x}_{k+1} = \mathbf{x}_k + t_k \mathbf{d}_k, \quad k = 0, 1, \ldots, \tag{2}$$

where $\mathbf{x}_k$ is the current iterate, t_k is the step size found by one of the line search methods, $\mathbf{d}_k$ is the search direction given by next relations

$$\mathbf{d}_k := \mathfrak{d}_k := \mathfrak{d}(\beta_k, \mathbf{g}_k, \mathbf{d}_{k-1}) = \begin{cases} -\mathbf{g}_0, & k = 0, \\ -\mathbf{g}_k + \beta_k \mathbf{d}_{k-1}, & k \geq 1, \end{cases} \tag{3}$$

where $\mathbf{g}_k = \mathbf{g}(\mathbf{x}_k)$ and β_k is an appropriately defined real scalar, known as the conjugate gradient parameter. A number of conjugate gradient iterative methods have been defined using various modifications of the conjugate gradient direction $\mathbf{d}_k$ and the parameter β_k. The most popular conjugate gradient parameters β_k are surveyed by next formulas:

$$\begin{aligned}
&\beta_k^{FR} = \frac{\|\mathbf{g}_k\|^2}{\|\mathbf{g}_{k-1}\|^2}\ [1], \quad \beta_k^{CD} = -\frac{\|\mathbf{g}_k\|^2}{\mathbf{g}_{k-1}^{\mathrm{T}}\mathbf{d}_{k-1}}\ [2], \quad \beta_k^{DY} = \frac{\|\mathbf{g}_k\|^2}{\mathbf{y}_{k-1}^{\mathrm{T}}\mathbf{d}_{k-1}}\ [3], \\
&\beta_k^{HS} = \frac{\mathbf{y}_{k-1}^{\mathrm{T}}\mathbf{g}_k}{\mathbf{y}_{k-1}^{\mathrm{T}}\mathbf{d}_{k-1}}\ [4], \quad \beta_k^{PRP} = \frac{\mathbf{y}_{k-1}^{\mathrm{T}}\mathbf{g}_k}{\|\mathbf{g}_{k-1}\|^2}\ [5,6], \quad \beta_k^{LS} = -\frac{\mathbf{y}_{k-1}^{\mathrm{T}}\mathbf{g}_k}{\mathbf{g}_{k-1}^{\mathrm{T}}\mathbf{d}_{k-1}}\ [7], \\
&\beta_k^{DHSDL} = \frac{\|\mathbf{g}_k\|^2 - \frac{\|\mathbf{g}_k\|}{\|\mathbf{g}_{k-1}\|}\left|\mathbf{g}_k^{\mathrm{T}}\mathbf{g}_{k-1}\right|}{\mu\left|\mathbf{g}_k^{\mathrm{T}}\mathbf{d}_{k-1}\right| + \mathbf{d}_{k-1}^{\mathrm{T}}\mathbf{y}_{k-1}} - t_k\frac{\mathbf{g}_k^{\mathrm{T}}\mathbf{s}_{k-1}}{\mathbf{k}_{k-1}^{\mathrm{T}}\mathbf{y}_{k-1}}\ [10], \\
&\beta_k^{DLSDL} = \frac{\|\mathbf{g}_k\|^2 - \frac{\|\mathbf{g}_k\|}{\|\mathbf{g}_{k-1}\|}\left|\mathbf{g}_k^{\mathrm{T}}\mathbf{g}_{k-1}\right|}{\mu\left|\mathbf{g}_k^{\mathrm{T}}\mathbf{d}_{k-1}\right| - \mathbf{k}_{k-1}^{\mathrm{T}}\mathbf{g}_{k-1}} - t_k\frac{\mathbf{g}_k^{\mathrm{T}}\mathbf{s}_{k-1}}{\mathbf{d}_{k-1}^{\mathrm{T}}\mathbf{y}_{k-1}}\ [10],
\end{aligned} \tag{4}$$

where $\mathbf{y}_{k-1} = \mathbf{g}_k - \mathbf{g}_{k-1}$, $\mathbf{s}_{k-1} = \mathbf{x}_k - \mathbf{x}_{k-1}$ and $\|\cdot\|$ stands for the Euclidean vector norm.

As usual, the step size t_k is determined using the backtracking line search procedure from [11]. It starts from $t = 1$, and it reduces the objective function sufficiently in each iteration. Therefore, we use the following Algorithm 1 from [12] for the inexact line search which determines t_k.

Algorithm 1 The backtracking line search starting from $t = 1$.

Require: Objective function $f(\mathbf{x})$, the direction $\mathbf{d}_k$ of the search at the point $\mathbf{x}_k$ and numbers $0 < \sigma < 0.5$ and $\beta \in]0, 1[$.

1: $t = 1$.
2: While $f(\mathbf{x}_k + t\mathbf{d}_k) > f(\mathbf{x}_k) + \sigma t \mathbf{g}_k^{\mathrm{T}}\mathbf{d}_k$, take $t := t\beta$.
3: Return $t_k = t$.

To combine the best numerical performances of the *PRP* method and the global convergence properties of the *FR* method, Touati–Ahmed and Storey [13] proposed a hybrid *PRP–FR* method, which is called the *H*1 method in [14], using the conjugate gradient parameter defined as

$$\beta_k^{H1} = \max\left\{0, \min\left\{\beta_k^{PRP}, \beta_k^{FR}\right\}\right\}. \tag{5}$$

Gilbert and Nocedal in [15] modified (5) to

$$\beta_k = \max\left\{-\beta_k^{FR}, \min\left\{\beta_k^{PRP}, \beta_k^{FR}\right\}\right\}.$$

A hybrid *HS–DY* conjugate gradient method was proposed by Dai and Yuan in [16]. This method is termed as the $H2$ method in [14]. The $H2$ method is defined by the conjugate gradient parameter

$$\beta_k^{H2} = \max\left\{0, \min\left\{\beta_k^{HS}, \beta_k^{DY}\right\}\right\}. \tag{6}$$

Numerical results derived in [13,16,17] show better performances of $H1$ and $H2$ methods with respect to the *PRP* method.

We consider hybrid CG algorithms where the search direction $\mathbf{d}_k := \mathfrak{d}_k$, $k \geq 1$, from (3) is modified using one of the following two rules:

$$\mathbf{d}_k := \mathfrak{D}(\beta_k, \mathbf{g}_k, \mathbf{d}_{k-1}) = -\left(1 + \beta_k \frac{\mathbf{g}_k^{\mathrm{T}}\mathbf{d}_{k-1}}{\|\mathbf{g}_k\|^2}\right)\mathbf{g}_k + \beta_k \mathbf{d}_{k-1}, \tag{7}$$

$$\mathbf{d}_k := \mathfrak{D}_1(\beta_k, \mathbf{g}_k, \mathbf{d}_{k-1}) = -B_k\mathbf{g}_k + \mathfrak{D}(\beta_k, \mathbf{g}_k, \mathbf{d}_{k-1}), \tag{8}$$

and the conjugate gradient parameter β_k is defined using some proper combinations of the parameters β_k from (4) and already defined hybridizations of these parameters.

Following the idea from the descent three term *PRP* method from [18], Zhang et al. in [19,20] proposed a modification to the *FR* method, termed as the *MFR* method, using the search direction

$$\mathbf{d}_k := \mathfrak{D}\left(\beta_k^{FR}, \mathbf{g}_k, \mathbf{d}_{k-1}\right). \tag{9}$$

Zhang in [20] also proposed a modified *DY* method, which is known as the *MDY* method and which is defined by the gradient direction

$$\mathbf{d}_k := \mathfrak{D}\left(\beta_k^{DY}, \mathbf{g}_k, \mathbf{d}_{k-1}\right). \tag{10}$$

The *MFR* and *MDY* methods possess very useful property

$$\mathbf{g}_k^{\mathrm{T}}\mathbf{d}_k = -\|\mathbf{g}_k\|^2. \tag{11}$$

Further, the *MFR* and the *MDY* methods reduce to the *FR* method and the *DY* method, respectively, if the exact line search is used. The *MFR* method has proven to be globally convergent for non-convex functions with the Wolfe line search or the Armijo line search, and it is very efficient in real computations [19].

However, it is not known whether the *MDY* method converges globally. So, in the paper [14], the authors replaced β_k^{FR} in (9) and β_k^{DY} in (10) by β_k^{H1} from (5) and β_k^{H2} from (6), respectively. Then, they defined new hybrid *PRP–FR* and *HS–DY* methods,

which they call the *NH*1 method and the *NH*2 method, respectively. These methods are based upon the search directions

$$\text{NH1:} \quad \mathbf{d}_k := \mathfrak{D}\left(\beta_k^{H1}, \mathbf{g}_k, \mathbf{d}_{k-1}\right), \tag{12}$$

$$\text{NH2:} \quad \mathbf{d}_k := \mathfrak{D}\left(\beta_k^{H2}, \mathbf{g}_k, \mathbf{d}_{k-1}\right). \tag{13}$$

It is easy to see that the *NH*1 and the *NH*2 methods still satisfy (11), which shows that they are descent methods.

On the other hand, the search direction $\mathbf{d}_k$ in quasi-Newton methods is obtained as a solution of the linear algebraic system

$$B_k \mathbf{d}_k = -\mathbf{g}_k, \tag{14}$$

where B_k is an approximation of the Hessian. The initial approximation is the identity matrix $B_0 = I$, and subsequent updates B_k are defined by an appropriate formula. Here we are interested in the *BFGS* update formula, defined by

$$B_{k+1} = B_k - \frac{B_k \mathbf{s}_k \mathbf{s}_k^{\mathrm{T}} B_k}{\mathbf{s}_k^{\mathrm{T}} B_k \mathbf{s}_k} + \frac{\mathbf{y}_k \mathbf{y}_k^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}} \mathbf{y}_k}, \tag{15}$$

where $\mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$, $\mathbf{y}_k = \mathbf{g}_{k+1} - \mathbf{g}_k$. The next secant equation must hold:

$$B_{k+1} \mathbf{s}_k = \mathbf{y}_k, \tag{16}$$

which is possible only if the curvature condition

$$\mathbf{s}_k^{\mathrm{T}} \mathbf{y}_k > 0 \tag{17}$$

is satisfied.

The three-term hybrid *BFGS* conjugate gradient method was proposed in [21]. That method uses best properties of both *BFGS* and *CG* methods and defines a hybrid *BFGS-CG* method for solving some selected unconstrained optimization problems, resulting in improvement in the total number of iterations and the CPU time.

3 A Mixed LS-CD Conjugate Gradient Method

We consider the modification of *LSCD* method, defined in [8] by

$$\begin{aligned} \beta_k^{LSCD} &= \max\left\{0, \min\left\{\beta_k^{LS}, \beta_k^{CD}\right\}\right\}, \\ \mathbf{d}_k &= \mathfrak{d}\left(\beta_k^{LSCD}, \mathbf{g}_k, \mathbf{d}_{k-1}\right) \end{aligned} \tag{18}$$

and define the *MLSCD* method with the search direction

$$\mathbf{d}_k := \mathfrak{D}\left(\beta_k^{LSCD}, \mathbf{g}_k, \mathbf{d}_{k-1}\right). \tag{19}$$

In general, our idea is to replace $\mathfrak{d}_k(\beta_k^{LSCD}, \mathbf{g}_k, \mathbf{d}_{k-1})$ from [8] with $\mathbf{d}_k = \mathfrak{D}(\beta_k^{LSCD}, \mathbf{g}_k, \mathbf{d}_{k-1})$.

Now we give the algorithm of this method.

Algorithm 2 MLSCD method.

Require: A starting point x_0, parameters $0 < \epsilon < 1$, $0 < \delta < 1$.

1: Set $k = 0$ and compute $\mathbf{d}_0 = -\mathbf{g}_0$.

2: If

$$\|\mathbf{g}_k\| \leq \epsilon \quad \text{and} \quad \frac{|f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k)|}{1 + |f(\mathbf{x}_k)|} \leq \delta,$$

STOP; else go to Step 3.

3: (Backtracking) Find the step size $t_k \in]0, 1]$ using Algorithm 1.

4: Compute $\mathbf{x}_{k+1} = \mathbf{x}_k + t_k \mathbf{d}_k$.

5: Calculate $\mathbf{g}_{k+1}$, $\mathbf{y}_k = \mathbf{g}_{k+1} - \mathbf{g}_k$ and go to Step 6.

6: Calculate

$$\beta_{k+1}^{LS} = \frac{\mathbf{y}_k^{\mathrm{T}} \mathbf{g}_{k+1}}{-\mathbf{g}_k^{\mathrm{T}} \mathbf{d}_k}, \quad \beta_{k+1}^{CD} = \frac{\|\mathbf{g}_{k+1}\|^2}{-\mathbf{g}_k^{\mathrm{T}} \mathbf{d}_k},$$
$$\beta_{k+1}^{LSCD} = \max\left\{0, \min\left\{\beta_{k+1}^{LS}, \beta_{k+1}^{CD}\right\}\right\}.$$

7: Compute the search direction $\mathbf{d}_{k+1} = \mathfrak{D}(\beta_{k+1}^{LSCD}, \mathbf{g}_{k+1}, \mathbf{d}_k)$.

8: Let $k := k + 1$, and go to Step 2.

3.1 Convergence of the MLSCD Conjugate Gradient Method

First, it is easy to prove the next theorem.

Theorem 3.1 *Let β_k be any CG parameter. Then, the search direction $\mathbf{d}_k := \mathfrak{D}(\beta_k, \mathbf{g}_k, \mathbf{d}_{k-1})$ satisfies*

$$\mathbf{g}_k^{\mathrm{T}} \mathbf{d}_k = -\|\mathbf{g}_k\|^2. \tag{20}$$

We are going to prove the global convergence of the *MLSCD* method under the following assumptions.

Assumption 3.1 (1) *The level set $S = \{\mathbf{x} \in \mathbb{R}^n | \ f(\mathbf{x}) \leq f(\mathbf{x}_0)\}$ is bounded.*

(2) *The function $\mathbf{g}$ is continuously differentiable in some neighborhood $\mathcal{N}$ of S and its gradient is Lipschitz continuous. Namely, there exists a constant $L > 0$, such that*

$$\|\mathbf{g}(\mathbf{x}) - \mathbf{g}(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|, \quad \textit{for all} \quad \mathbf{x}, \mathbf{y} \in \mathbb{N}. \tag{21}$$

It is well known that if Assumption 3.1 holds, then there exists a positive constant Γ, such that

$$\|\mathbf{g}_k\| \leq \Gamma, \quad \text{for all} \quad k. \tag{22}$$

The outcome of the next lemma, often called the Zoutendijk condition, is used to prove the global convergence of nonlinear *CG* methods. Originally, it was given in [22].

Lemma 3.1 [22,23] *Let the conditions in Assumption 3.1 be satisfied. Let the sequence* $\{\mathbf{x}_k\}$ *be generated by the MLSCD method with the backtracking line search. Then it holds that*

$$\sum_{k=0}^{\infty} \frac{\|\mathbf{g}_k\|^4}{\|\mathbf{d}_k\|^2} < +\infty. \tag{23}$$

Proof Consider the case where the backtracking line search is used. If $t_k \neq \beta$, then it is not difficult to show that there is a constant $c > 0$ such that

$$t_k \geq c \frac{-\mathbf{g}_k^{\mathrm{T}} \mathbf{d}_k}{\|\mathbf{d}_k\|^2} = c \frac{\|\mathbf{g}_k\|^2}{\|\mathbf{d}_k\|^2}. \tag{24}$$

This together with backtracking line search implies that there exists a constant $M > 0$ such that

$$\frac{\|\mathbf{g}_k\|^4}{\|\mathbf{d}_k\|^2} \leq M(f(\mathbf{x}_k) - f(\mathbf{x}_{k+1})). \tag{25}$$

On the other hand, if $t_k = \zeta$ then it follows that

$$\frac{\|\mathbf{g}_k\|^4}{\|\mathbf{d}_k\|^2} \leq \|\mathbf{g}_k\|^2 \leq \|\mathbf{d}_k\|^2 \leq \delta^{-1}\zeta^{-2}(f(\mathbf{x}_k) - f(\mathbf{x}_{k+1})).$$

This also implies (25) with some $M = \delta^{-1}\zeta^{-2}$. □

Theorem 3.2 *Let the conditions of Assumption 3.1 hold. Then the sequence* $\{\mathbf{x}_k\}$, *generated by the MLSCD method with the backtracking line search satisfies*

$$\liminf_{k \to \infty} \|\mathbf{g}_k\| = 0. \tag{26}$$

Proof In order to gain the contradiction, let us suppose that (26) does not hold. Then, there exists a constant $c > 0$ such that

$$\|\mathbf{g}_k\| \geq c, \quad \text{for all} \quad k \geq 0. \tag{27}$$

Clearly, (19) can be rewritten into the form

$$\mathbf{d}_k = -l_k \mathbf{g}_k + \beta_k^{LSCD} \mathbf{d}_{k-1}, \quad l_k = 1 + \beta_k^{LSCD} \frac{\mathbf{g}_k^{\mathrm{T}} \mathbf{d}_{k-1}}{\|\mathbf{g}_k\|^2}. \tag{28}$$

Now, from (28), it follows that

$$\mathbf{d}_k + l_k \mathbf{g}_k = \beta_k^{LSCD}\, \mathbf{d}_{k-1},$$

which further implies

$$(\mathbf{d}_k + l_k \mathbf{g}_k)^2 = \left(\beta_k^{LSCD}\, \mathbf{d}_{k-1}\right)^2,$$
$$\|\mathbf{d}_k\|^2 + 2l_k \mathbf{d}_k^{\mathrm{T}} \mathbf{g}_k + l_k^2 \|\mathbf{g}_k\|^2 = \left(\beta_k^{LSCD}\right)^2 \|\mathbf{d}_{k-1}\|^2,$$

and subsequently

$$\|\mathbf{d}_k\|^2 = \left(\beta_k^{LSCD}\right)^2 \|\mathbf{d}_{k-1}\|^2 - 2l_k \mathbf{d}_k^{\mathrm{T}} \mathbf{g}_k - l_k^2 \|\mathbf{g}_k\|^2. \tag{29}$$

Notice that

$$\beta_k^{LSCD} = \max\left\{0, \min\left\{\beta_k^{LS}, \beta_k^{CD}\right\}\right\} \le |\beta_k^{CD}|. \tag{30}$$

Dividing both sides of (29) by $(\mathbf{g}_k^{\mathrm{T}} \mathbf{d}_k)^2$, we get from (30), (20), (27) and the definition of β_k^{CD} that

$$\begin{aligned}
\frac{\|\mathbf{d}_k\|^2}{\|\mathbf{g}_k\|^4} = \frac{\|\mathbf{d}_k\|^2}{(\mathbf{g}_k^{\mathrm{T}} \mathbf{d}_k)^2} &= \left(\beta_k^{LSCD}\right)^2 \frac{\|\mathbf{d}_{k-1}\|^2}{(\mathbf{g}_k^{\mathrm{T}} \mathbf{d}_k)^2} - \frac{2l_k \mathbf{d}_k^{\mathrm{T}} \mathbf{g}_k}{(\mathbf{g}_k^{\mathrm{T}} \mathbf{d}_k)^2} - l_k^2 \frac{\|\mathbf{g}_k\|^2}{(\mathbf{g}_k^{\mathrm{T}} \mathbf{d}_k)^2} \\
&\le \left(\beta_k^{CD}\right)^2 \frac{\|\mathbf{d}_{k-1}\|^2}{(\mathbf{g}_k^{\mathrm{T}} \mathbf{d}_k)^2} - \frac{2l_k}{\mathbf{g}_k^{\mathrm{T}} \mathbf{d}_k} - l_k^2 \frac{\|\mathbf{g}_k\|^2}{(\mathbf{g}_k^{\mathrm{T}} \mathbf{d}_k)^2} \\
&= \left(\frac{\|\mathbf{g}_k\|^2}{-\mathbf{g}_{k-1}^{\mathrm{T}} \mathbf{d}_{k-1}}\right)^2 \frac{\|\mathbf{d}_{k-1}\|^2}{(\mathbf{g}_k^{\mathrm{T}} \mathbf{d}_k)^2} - \frac{2l_k}{\mathbf{g}_k^{\mathrm{T}} \mathbf{d}_k} - l_k^2 \frac{\|\mathbf{g}_k\|^2}{(\mathbf{g}_k^{\mathrm{T}} \mathbf{d}_k)^2}.
\end{aligned} \tag{31}$$

Now, applying (20) in (31), one can verify

$$\begin{aligned}
\frac{\|\mathbf{d}_k\|^2}{\|\mathbf{g}_k\|^4} &= \frac{\|\mathbf{g}_k\|^4}{\|\mathbf{g}_{k-1}\|^4} \frac{\|\mathbf{d}_{k-1}\|^2}{\|\mathbf{g}_k\|^4} + \frac{2l_k}{\|\mathbf{g}_k\|^2} - l_k^2 \frac{\|\mathbf{g}_k\|^2}{\|\mathbf{g}_k\|^4} \\
&= \frac{\|\mathbf{d}_{k-1}\|^2}{\|\mathbf{g}_{k-1}\|^4} + \frac{2l_k}{\|\mathbf{g}_k\|^2} - l_k^2 \frac{1}{\|\mathbf{g}_k\|^2} \\
&= \frac{\|\mathbf{d}_{k-1}\|^2}{\|\mathbf{g}_{k-1}\|^4} - \frac{(l_k - 1)^2}{\|\mathbf{g}_k\|^2} + \frac{1}{\|\mathbf{g}_k\|^2} \\
&\le \frac{\|\mathbf{d}_{k-1}\|^2}{\|\mathbf{g}_{k-1}\|^4} + \frac{1}{\|\mathbf{g}_k\|^2} \\
&\le \sum_{j=0}^{k} \frac{1}{\|\mathbf{g}_j\|^2} \\
&\le \frac{k+1}{c^2}.
\end{aligned}$$

The last inequalities imply

$$\sum_{k\geq 1}\frac{\|\mathbf{g}_k\|^4}{\|\mathbf{d}_k\|^2}\geq c^2\sum_{k\geq 1}\frac{1}{k+1}=\infty, \tag{32}$$

which contradicts to (23). The proof is thus complete. □

4 A Mixed DHSDL-DLSDL Conjugate Gradient Method

In this section, we propose the hybrid *MMDL* method which is defined by the search direction $\mathbf{d}_k$ as follows:

$$\beta_k^{MMDL}=\max\left\{0,\min\left\{\beta_k^{DHSDL},\beta_k^{DLSDL}\right\}\right\}, \tag{33}$$

$$\mathbf{d}_k=\mathfrak{D}\left(\beta_k^{MMDL},\mathbf{g}_k,\mathbf{d}_{k-1}\right). \tag{34}$$

The computational algorithm of this method is presented in Algorithm 3.

Algorithm 3 MMDL method.

Require: A starting point x_0, parameters $0<\epsilon<1$, $0<\delta<1$, $\mu>1$.

1: Set $k=0$ and compute $d_0=-g_0$.

2: If

$$\|\mathbf{g}_k\|\leq\epsilon \quad\text{and}\quad \frac{|f(\mathbf{x}_{k+1})-f(\mathbf{x}_k)|}{1+|f(\mathbf{x}_k)|}\leq\delta,$$

STOP; else go to Step 3.

3: (Backtracking) Find the step size $t_k\in]0,1]$ using Algorithm 1.

4: Compute $\mathbf{x}_{k+1}=\mathbf{x}_k+t_k\mathbf{d}_k$.

5: Calculate $\mathbf{g}_{k+1}$, $\mathbf{y}_k=\mathbf{g}_{k+1}-\mathbf{g}_k$, $\mathbf{s}_k=\mathbf{x}_{k+1}-\mathbf{x}_k$ and go to Step 6.

6: Calculate

$$\beta_{k+1}^{DHSDL}=\frac{\|\mathbf{g}_{k+1}\|^2-\frac{\|\mathbf{g}_{k+1}\|}{\|\mathbf{g}_k\|}\left|\mathbf{g}_{k+1}^{\mathrm{T}}\mathbf{g}_k\right|}{\mu\left|\mathbf{g}_{k+1}^{\mathrm{T}}\mathbf{d}_k\right|+\mathbf{d}_k^{\mathrm{T}}\mathbf{y}_k}-t_k\frac{\mathbf{g}_{k+1}^{\mathrm{T}}\mathbf{s}_k}{\mathbf{d}_k^{\mathrm{T}}\mathbf{y}_k},$$

$$\beta_{k+1}^{DLSDL}=\frac{\|\mathbf{g}_{k+1}\|^2-\frac{\|\mathbf{g}_{k+1}\|}{\|\mathbf{g}_k\|}\left|\mathbf{g}_{k+1}^{\mathrm{T}}\mathbf{g}_k\right|}{\mu\left|\mathbf{g}_{k+1}^{\mathrm{T}}\mathbf{d}_k\right|-\mathbf{d}_k^{\mathrm{T}}\mathbf{g}_k}-t_k\frac{\mathbf{g}_{k+1}^{\mathrm{T}}\mathbf{s}_k}{\mathbf{d}_k^{\mathrm{T}}\mathbf{y}_k},$$

$$\beta_{k+1}^{MMDL}=\max\left\{0,\min\left\{\beta_{k+1}^{DHSDL},\beta_{k+1}^{DLSDL}\right\}\right\}.$$

7: Compute the search direction $\mathbf{d}_{k+1}=\mathfrak{D}(\beta_{k+1}^{MMDL},\mathbf{g}_{k+1},\mathbf{d}_k)$.

8: Let $k:=k+1$, and go to Step 2.

4.1 Convergence of the MMDL Conjugate Gradient Method

We now prove the global convergence of the *MMDL* method for arbitrary objective functions.

Theorem 4.1 *Let the conditions in Assumption 3.1 hold. Then the sequence* $\{\mathbf{x}_k\}$ *generated by the MMDL method with the backtracking line search satisfies*

$$\liminf_{k\to\infty} \|\mathbf{g}_k\| = 0. \tag{35}$$

Proof Assume, on the contrary, that (35) does not hold. Then, there exists a constant $c > 0$ such that

$$\|\mathbf{g}_k\| \geq c, \quad \text{for all} \quad k. \tag{36}$$

Denote

$$l_k = 1 + \beta_k^{MMDL} \frac{\mathbf{g}_k^{\mathrm{T}}\mathbf{d}_{k-1}}{\|\mathbf{g}_k\|^2}. \tag{37}$$

Then we can write

$$\mathbf{d}_k + l_k\mathbf{g}_k = \beta_k^{MMDL}\mathbf{d}_{k-1}, \tag{38}$$

and further

$$(\mathbf{d}_k + l_k\mathbf{g}_k)^2 = \left(\beta_k^{MMDL}\mathbf{d}_{k-1}\right)^2,$$

$$\|\mathbf{d}_k\|^2 + 2l_k\mathbf{d}_k^{\mathrm{T}}\mathbf{g}_k + l_k^2\|\mathbf{g}_k\|^2 = \left(\beta_k^{MMDL}\right)^2\|\mathbf{d}_{k-1}\|^2.$$

Thus,

$$\|\mathbf{d}_k\|^2 = \left(\beta_k^{MMDL}\right)^2\|\mathbf{d}_{k-1}\|^2 - 2l_k\mathbf{d}_k^{\mathrm{T}}\mathbf{g}_k - l_k^2\|\mathbf{g}_k\|^2. \tag{39}$$

Having in view $\mu > 1$ as well as $\mathbf{d}_k^{\mathrm{T}}\mathbf{g}_k < 0$ and applying the extended conjugacy condition $\mathbf{d}_k^{\mathrm{T}}\mathbf{y}_{k-1} = t\mathbf{g}_k^{\mathrm{T}}\mathbf{s}_{k-1}$, $t > 0$, which was exploited in [9,10], we get

$$\begin{aligned}
\beta_{k+1}^{DHSDL} &= \frac{\|\mathbf{g}_{k+1}\|^2 - \frac{\|\mathbf{g}_{k+1}\|}{\|\mathbf{g}_k\|}\left|\mathbf{g}_{k+1}^{\mathrm{T}}\mathbf{g}_k\right|}{\mu\left|\mathbf{g}_{k+1}^{\mathrm{T}}\mathbf{d}_k\right| + \mathbf{d}_k^{\mathrm{T}}\mathbf{y}_k} - t_k\frac{\mathbf{g}_{k+1}^{\mathrm{T}}\mathbf{s}_k}{\mathbf{d}_k^{\mathrm{T}}\mathbf{y}_k} \\
&\leq \frac{\|\mathbf{g}_{k+1}\|^2 - \frac{\|\mathbf{g}_{k+1}\|}{\|\mathbf{g}_k\|}\left|\mathbf{g}_{k+1}^{\mathrm{T}}\mathbf{g}_k\right|}{\mu\left|\mathbf{g}_{k+1}^{\mathrm{T}}\mathbf{d}_k\right| + \mathbf{d}_k^{\mathrm{T}}\mathbf{y}_k} \\
&= \frac{\|\mathbf{g}_{k+1}\|^2 - \frac{\|\mathbf{g}_{k+1}\|}{\|\mathbf{g}_k\|}\left|\mathbf{g}_{k+1}^{\mathrm{T}}\mathbf{g}_k\right|}{\mu\left|\mathbf{g}_{k+1}^{\mathrm{T}}\mathbf{d}_k\right| + \mathbf{d}_k^{\mathrm{T}}(\mathbf{g}_{k+1} - \mathbf{g}_k)} \\
&= \frac{\|\mathbf{g}_{k+1}\|^2 - \frac{\|\mathbf{g}_{k+1}\|}{\|\mathbf{g}_k\|}\left|\mathbf{g}_{k+1}^{\mathrm{T}}\mathbf{g}_k\right|}{\mu\left|\mathbf{g}_{k+1}^{\mathrm{T}}\mathbf{d}_k\right| + \mathbf{d}_k^{\mathrm{T}}\mathbf{g}_{k+1} - \mathbf{d}_k^{\mathrm{T}}\mathbf{g}_k} \\
&\leq \frac{\|\mathbf{g}_{k+1}\|^2}{\mu\left|\mathbf{g}_{k+1}^{\mathrm{T}}\mathbf{d}_k\right| + \mathbf{d}_k^{\mathrm{T}}\mathbf{g}_{k+1} - \mathbf{d}_k^{\mathrm{T}}\mathbf{g}_k} \\
&\leq \frac{\|\mathbf{g}_{k+1}\|^2}{-\mathbf{d}_k^{\mathrm{T}}\mathbf{g}_k}.
\end{aligned}$$

Further,

$$\begin{aligned}\beta_{k+1}^{DLSDL} &= \frac{\|\mathbf{g}_{k+1}\|^2 - \frac{\|\mathbf{g}_{k+1}\|}{\|\mathbf{g}_k\|}\left|\mathbf{g}_{k+1}^{\mathrm{T}}\mathbf{g}_k\right|}{\mu\left|\mathbf{g}_{k+1}^{\mathrm{T}}\mathbf{d}_k\right| - \mathbf{d}_k^{\mathrm{T}}\mathbf{g}_k} - t_k\frac{\mathbf{g}_{k+1}^{\mathrm{T}}\mathbf{s}_k}{\mathbf{d}_k^{\mathrm{T}}\mathbf{y}_k}\\ &\leq \frac{\|\mathbf{g}_{k+1}\|^2 - \frac{\|\mathbf{g}_{k+1}\|}{\|\mathbf{g}_k\|}\left|\mathbf{g}_{k+1}^{\mathrm{T}}\mathbf{g}_k\right|}{\mu\left|\mathbf{g}_{k+1}^{\mathrm{T}}\mathbf{d}_k\right| - \mathbf{d}_k^{\mathrm{T}}\mathbf{g}_k}\\ &\leq \frac{\|\mathbf{g}_{k+1}\|^2}{\mu\left|\mathbf{g}_{k+1}^{\mathrm{T}}\mathbf{d}_k\right| - \mathbf{d}_k^{\mathrm{T}}\mathbf{g}_k}\\ &\leq \frac{\|\mathbf{g}_{k+1}\|^2}{-\mathbf{d}_k^{\mathrm{T}}\mathbf{g}_k}.\end{aligned}$$

Now, we easily conclude

$$\beta_k^{MMDL} = \max\left\{0, \min\left\{\beta_k^{DHSDL}, \beta_k^{DLSDL}\right\}\right\} \leq \frac{\|\mathbf{g}_k\|^2}{-\mathbf{d}_{k-1}^{\mathrm{T}}\mathbf{g}_{k-1}}. \tag{40}$$

Next, dividing both sides of (39) by $(\mathbf{g}_k^{\mathrm{T}}\mathbf{d}_k)^2$, we get from (20), (40) and (36) that

$$\begin{aligned}\frac{\|\mathbf{d}_k\|^2}{\|\mathbf{g}_k\|^4} = \frac{\|\mathbf{d}_k\|^2}{(\mathbf{g}_k^{\mathrm{T}}\mathbf{d}_k)^2} &= \left(\beta_k^{MMDL}\right)^2\frac{\|\mathbf{d}_{k-1}\|^2}{(\mathbf{g}_k^{\mathrm{T}}\mathbf{d}_k)^2} - \frac{2l_k\mathbf{d}_k^{\mathrm{T}}\mathbf{g}_k}{(\mathbf{g}_k^{\mathrm{T}}\mathbf{d}_k)^2} - l_k^2\frac{\|\mathbf{g}_k\|^2}{(\mathbf{g}_k^{\mathrm{T}}\mathbf{d}_k)^2}\\ &= \left(\beta_k^{MMDL}\right)^2\frac{\|\mathbf{d}_{k-1}\|^2}{(\mathbf{g}_k^{\mathrm{T}}\mathbf{d}_k)^2} - \frac{2l_k}{\mathbf{g}_k^{\mathrm{T}}\mathbf{d}_k} - l_k^2\frac{\|\mathbf{g}_k\|^2}{(\mathbf{g}_k^{\mathrm{T}}\mathbf{d}_k)^2}\\ &\leq \left(\frac{\|\mathbf{g}_k\|^2}{-\mathbf{g}_{k-1}^{\mathrm{T}}\mathbf{d}_{k-1}}\right)^2\frac{\|\mathbf{d}_{k-1}\|^2}{(\mathbf{g}_k^{\mathrm{T}}\mathbf{d}_k)^2} - \frac{2l_k}{\mathbf{g}_k^{\mathrm{T}}\mathbf{d}_k} - l_k^2\frac{\|\mathbf{g}_k\|^2}{(\mathbf{g}_k^{\mathrm{T}}\mathbf{d}_k)^2}\\ &= \frac{\|\mathbf{g}_k\|^4}{\|\mathbf{g}_{k-1}\|^4}\frac{\|\mathbf{d}_{k-1}\|^2}{\|\mathbf{g}_k\|^4} + \frac{2l_k}{\|\mathbf{g}_k\|^2} - l_k^2\frac{\|\mathbf{g}_k\|^2}{\|\mathbf{g}_k\|^4}\\ &= \frac{\|\mathbf{d}_{k-1}\|^2}{\|\mathbf{g}_{k-1}\|^4} + \frac{2l_k}{\|\mathbf{g}_k\|^2} - l_k^2\frac{1}{\|\mathbf{g}_k\|^2}\\ &= \frac{\|\mathbf{d}_{k-1}\|^2}{\|\mathbf{g}_{k-1}\|^4} - \frac{(l_k-1)^2}{\|\mathbf{g}_k\|^2} + \frac{1}{\|\mathbf{g}_k\|^2}\\ &\leq \frac{\|\mathbf{d}_{k-1}\|^2}{\|\mathbf{g}_{k-1}\|^4} + \frac{1}{\|\mathbf{g}_k\|^2}\\ &\leq \sum_{j=0}^{k}\frac{1}{\|\mathbf{g}_j\|^2}\\ &\leq \frac{k+1}{c^2}.\end{aligned} \tag{41}$$

The inequalities in (41) imply

$$\sum_{k\geq 1}\frac{\|\mathbf{g}_k\|^4}{\|\mathbf{d}_k\|^2}\geq c^2\sum_{k\geq 1}\frac{1}{k+1}=\infty. \tag{42}$$

Therefore, $\|\mathbf{g}_k\|\geq c$ causes a contradiction to (23). Consequently, (35) is verified. □

5 Hybrid Three-Term Broyden–Fletcher–Goldfarb–Shanno Conjugate Gradient Methods in Solving Unconstrained Optimization Problems

It is known that conjugate gradient methods are better compared to the quasi-Newton method in terms of the CPU time. In addition, *BFGS* is more costly in terms of the memory storage requirements than *CG*. On the other hand, the quasi-Newton methods are better in terms of the number of iterations and the number of function evaluations. For this purpose, various hybridizations of quasi-Newton methods and *CG* methods have been proposed by previous researchers. The new hybrid method which solves the system of nonlinear equations combining the quasi-Newton method with chaos optimization was proposed in [24]. In [25], the authors defined a combination of a quasi-Newton and the Cauchy descent method for solving unconstrained optimization problems, which is known as the quasi-Newton SD method. In [21], the authors proposed a hybrid search direction that combines the quasi-Newton and *CG* methods. It yields a search direction of the hybrid method which is known as the *BFGS-CG* method in [21]. The search direction of the method from [21] is defined by

$$\mathbf{d}_k=\begin{cases}-B_k\mathbf{g}_k, & k=0,\\ -B_k\mathbf{g}_k+\eta(-\mathbf{g}_k+\beta_k\mathbf{d}_{k-1}), & k\geq 1,\end{cases}$$

where $\eta>0$ and $\beta_k=\dfrac{\mathbf{g}_k^{\mathrm{T}}\mathbf{g}_{k-1}}{\mathbf{g}_k^{\mathrm{T}}\mathbf{d}_{k-1}}$. A hybrid direction search between *BFGS* update of the Hessian matrix and the conjugate coefficient β_k was proposed and investigated in [26,27]. A Hybrid *DFP-CG* method for solving unconstrained optimization problems was presented in [28].

5.1 Three-Term H-BFGS-CG1 Method

Our idea is to consider a three-term hybrid *BFGS-CG* method (called *H-BFGS-CG*1), defined by the search direction

$$\mathbf{d}_k:=\begin{cases}-B_k\mathbf{g}_k, & k=0,\\ \mathfrak{D}_1\left(\beta_k^{LSCD},\mathbf{g}_k,\mathbf{d}_{k-1}\right), & k\geq 1.\end{cases}$$

Algorithm 4 defines the corresponding computational procedure of the *H-BFGS-CG*1 method.

Algorithm 4 H-BFGS-CG1 algorithm.

Require: A starting point $\mathbf{x}_0$, parameters $0 < \epsilon < 1$, $0 < \delta < 1$.

1: Set $k = 0$ and compute $\mathbf{g}_0$, $B_0 = I$, $\mathbf{d}_0 = -B_0\mathbf{g}_0$.

2: If

$$\|\mathbf{g}_k\| \leq \epsilon \quad \text{and} \quad \frac{|f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k)|}{1 + |f(\mathbf{x}_k)|} \leq \delta,$$

STOP; else go to Step 3.

3: (Backtracking) Find the step size $t_k \in]0, 1]$ using Algorithm 1.

4: Compute $\mathbf{x}_{k+1} = \mathbf{x}_k + t_k\mathbf{d}_k$.

5: Calculate $\mathbf{g}_{k+1}, \mathbf{y}_k = \mathbf{g}_{k+1} - \mathbf{g}_k, \mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$ and go to Step 6.

6: Calculate

$$\beta_{k+1}^{LS} = \frac{\mathbf{y}_k^{\mathrm{T}}\mathbf{g}_{k+1}}{-\mathbf{g}_k^{\mathrm{T}}\mathbf{d}_k}, \quad \beta_{k+1}^{CD} = \frac{\|\mathbf{g}_{k+1}\|^2}{-\mathbf{g}_k^{\mathrm{T}}\mathbf{d}_k},$$
$$\beta_{k+1}^{LSCD} = \max\left\{0, \min\left\{\beta_{k+1}^{LS}, \beta_{k+1}^{CD}\right\}\right\}.$$

7: Compute B_{k+1} using (15).

8: Compute the search direction $d_{k+1} = \mathfrak{D}_1(\beta_{k+1}^{LSCD}, \mathbf{g}_{k+1}, \mathbf{d}_k)$.

9: Let $k := k + 1$, and go to Step 2.

5.2 Convergence Analysis of H-BFGS-CG1 Method

The following assumptions are required in this section.

Assumption 5.1 *H*1: *The objective function f is twice continuously differentiable.*

*H*2: *The level set S is convex. Moreover, there exist positive constants c_1 and c_2 such that*

$$c_1\|\mathbf{z}\|^2 \leq \mathbf{z}^{\mathrm{T}}F(\mathbf{x})\mathbf{z} \leq c_2\|\mathbf{z}\|^2,$$

for all $\mathbf{z} \in \mathbb{R}^n$ and $\mathbf{x} \in S$, where $F(\mathbf{x})$ is the Hessian of f.

*H*3: *The gradient $\mathbf{g}$ is Lipschitz continuous at the point $\mathbf{x}^*$, that is, there exists the positive constant c_3 satisfying*

$$\|\mathbf{g}(\mathbf{x}) - g(\mathbf{x}*)\| \leq c_3\|\mathbf{x} - \mathbf{x}^*\|,$$

for all $\mathbf{x}$ in a neighborhood of $\mathbf{x}^*$.

Theorem 5.1 [29] *Let $\{B_k\}$ be generated by the BFGS update formula (15), where $\mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$, $\mathbf{y}_k = \mathbf{g}_{k+1} - \mathbf{g}_k$. Assume that the matrix B_k is symmetric and positive definite and satisfies (16) and (17) for all k. Furthermore, assume that $\{\mathbf{s}_k\}$ and $\{\mathbf{y}_k\}$ satisfy the inequality*

$$\frac{\|\mathbf{y}_k - G_*\mathbf{s}_k\|}{\|\mathbf{s}_k\|} \leq \epsilon_k,$$

for some symmetric and positive matrix G_* and for some sequence $\{\epsilon_k\}$ possessing the property

$$\sum_{k=1}^{\infty} \epsilon_k < \infty.$$

Then

$$\lim_{k\to\infty} \frac{\|(B_k - G_*)\mathbf{s}_k\|}{\|\mathbf{s}_k\|} = 0$$

and the sequences $\{\|B_k\|\}$, $\{\|B_k^{-1}\|\}$ are bounded.

Theorem 5.2 (Sufficient descent and global convergence.) *Consider Algorithm 4. Assume that the conditions $H1$, $H2$ and $H3$ in Assumption 5.1 are satisfied as well as conditions of Theorem 5.1. Then*

$$\lim_{k\to\infty} \|\mathbf{g}_k\|^2 = 0.$$

Proof It holds

$$\begin{aligned} \mathbf{g}_k^{\mathrm{T}}\mathbf{d}_k &= -\mathbf{g}_k^{\mathrm{T}} B_k \mathbf{g}_k - \mathbf{g}_k^{\mathrm{T}}\mathbf{g}_k - \beta_k^{LSCD}\mathbf{g}_k^{\mathrm{T}}\mathbf{d}_{k-1} + \beta_k^{LSCD}\mathbf{g}_k^{\mathrm{T}}\mathbf{d}_{k-1} \\ &\le -c_1\|\mathbf{g}_k\|^2 - \|\mathbf{g}_k\|^2 \\ &= -(c_1+1)\|\mathbf{g}_k\|^2 \le \|\mathbf{g}_k\|^2, \;\; c_1 + 1 > 0, \end{aligned} \tag{43}$$

wherefrom we conclude that the sufficient descent holds.

Further, from backtracking line search condition and (43), it holds

$$f(\mathbf{x}_k) - f(\mathbf{x}_k + t_k\mathbf{d}_k) \le -\sigma t_k \mathbf{g}_k^{\mathrm{T}}\mathbf{d}_k \le \sigma t_k \|\mathbf{g}_k\|^2. \tag{44}$$

Since $f(\mathbf{x}_k)$ is decreasing and the sequence $f(\mathbf{x}_k)$ is bounded below by $H2$, we have that

$$\lim_{k\to\infty} (f(\mathbf{x}_k) - f(\mathbf{x}_k + t_k\mathbf{d}_k)) = 0. \tag{45}$$

Hence, (44) and (45) imply

$$\lim_{k\to\infty} \sigma t_k \|\mathbf{g}_k\|^2 = 0.$$

Now, from $t_k > 0$ and $\sigma > 0$, we have

$$\lim_{k\to\infty} \|\mathbf{g}_k\|^2 = 0,$$

which was our initial intention. □

6 Numerical Experiments

In this section, we present numerical results obtained by testing *DHSDL*, *DLSDL*, *MMDL*, *MLSCD* and *H-BFGS-CG*1 methods. The codes used in the tests were written in the *Matlab* programming language, and the tests were performed on the computer Workstation Intel Core i3 2.0 GHz. The number of iterations, number of function evaluations and CPU time in all tested methods are analyzed.

In the first part, we compare *DHSDL*, *DLSDL*, *MMDL* and *MLSCD* methods. The tested 33 functions are selected from [30]. We have considered 10 different numerical experiments with the number of variables equal to 100, 500, 1000, 2000, 3000, 5000, 7000, 8000, 10,000 and 15,000, for each function in the Tables 1, 2 and 3. Summary numerical results for *DHSDL*, *DLSDL*, *MMDL* and *MLSCD*, tested on 33 large-scale test functions, are presented in Tables 1, 2 and 3.

The stopping conditions for all algorithms are

$$\|\mathbf{g}_k\| \leq 10^{-6} \quad \text{and} \quad \frac{|f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k)|}{1 + |f(\mathbf{x}_k)|} \leq 10^{-16}.$$

The backtracking parameters for all algorithms are $\sigma = 0.0001$ and $\beta = 0.8$, which means that we accept a small decrease in f predicted by a linear approximation at the current point.

Performance profiles of a given metric, proposed in [31], is a widely used tool for benchmarking and comparing the performance of an optimization software on a large test set. As usual, the number of iterations, number of function evaluations and the computing time (CPU time) are used as performance measures. Figure 1(left) shows the performances of compared methods relative to the number of iterations. Figure 1(right) illustrates the performance of these methods relative to the number of function evaluations. The top curve corresponds to the method that exhibits the best performances with respect to the chosen performance profile.

Figure 2 shows the performance of the considered methods relative to the CPU time.

In Fig. 1(left), it is observable that *DHSDL*, *DLSDL*, *MMDL* and *MLSCD* methods successfully solve all the problems, and the *MLSCD* method is best in 82% of the test problems compared to other methods.

In Fig. 1(right), it is observable that *DHSDL*, *DLSDL*, *MMDL* and *MLSCD* methods successfully solve all the problems, and the *MLSCD* method is best in 78% of the test problems compared to the other methods.

Graphs in Fig. 2 show that *DHSDL*, *DLSDL*, *MMDL* and *MLSCD* methods successfully solve all the problems, and the *MLSCD* method is best in 73% of the test problems compared to the other methods.

Consequently, Figs. 1 and 2 show that the *MLSCD* method generates the best result with respect to all three considered criteria.

Table 4 contains the average number of iterations, the CPU time, and the number of function evaluations for all 330 numerical experiments.

Based on the results arranged in Table 4, it is observable that the *MLSCD* method gives better results compared to *DHSDL*, *DLSDL* and *MMDL* methods. This conclu-

Table 1 Summary numerical results for the number of iterations

Test function	No. of iterations			
	DHSDL	*DLSDL*	*MMDL*	*MLSCD*
Extended penalty function	1215	1234	1205	1189
Perturbed quadratic function	270,275	221,969	259,193	6776
Raydan 1 function	53,186	49,828	49,015	3321
Raydan 2 function	390	390	56	56
Diagonal 1 function	29,328	27,623	32,947	3548
Diagonal 3 function	57,572	46,091	52,521	4477
Hager function	3185	3205	2962	1614
Generalized tridiagonal 1 function	622	625	622	1385
Extended tridiagonal 1 function	986,304	857	1,250,116	3939
Extended TET function	3320	3416	2005	1042
Diagonal 4 function	8184	8046	7778	1439
Diagonal 5 function	25,070	25,070	40	40
Extended Himmelblau function	1348	1348	1345	1614
Perturbed quadratic diagonal function	1,314,838	832,200	1,162,770	12,650
Quadratic QF1 function	269,831	231,134	262,582	8654
Extended quadratic penalty QP1 function	560	556	585	846
Extended quadratic penalty QP2 function	120,793	115,663	112,874	2863
Quadratic QF2 function	290,961	259,641	280,104	7875
Extended quadratic exponential EP1 function	506	506	501	501
Extended tridiagonal 2 function	1334	1339	1095	748
ARWHEAD function (CUTE)	37,844	34,583	37,508	3159
Almost perturbed quadratic function	273,128	225,250	262,536	6792
LIARWHD function (CUTE)	1,491,307	1,382,551	1,441,296	3498
ENGVAL1 function (CUTE)	558	570	561	1045
Diagonal 6 function	482	486	56	56
COSINE function (CUTE)	6420	13,212	5802	2154
Generalized quartic function	1540	1593	1226	1163
Diagonal 7 function	5163	5163	539	539
Diagonal 8 function	16,366	16,366	577	577
Full Hessian FH3 function	2458	2455	2454	2456
Diagonal 9 function	221,364	158,244	200,892	4654
HIMMELH function (CUTE)	90	90	90	67
Extended Rosenbrock	50	50	50	40

sion is confirmed by performance profiles for the number of iterations, number of function evaluations and the CPU time.

In order to compare all five methods (*DHSDL*, *DLSDL*, *MMDL*, *MLSCD* and *H-BFGS-CG*1), we are forced to reduce the number of variables due to the bad average CPU time of the *H-BFGS-CG*1 method during the testing. For each test problem, we

Table 2 Summary numerical results for the number of function evaluations

Test function	No. of funct. evaluation			
	DHSDL	*DLSDL*	*MMDL*	*MLSCD*
Extended penalty function	44,466	44,600	44,179	42,846
Perturbed quadratic function	11,355,510	9,214,352	10,909,537	271,818
Raydan 1 function	1,435,352	1,334,819	1,329,884	91,573
Raydan 2 function	970	970	122	122
Diagonal 1 function	1,100,988	1,032,762	1,252,937	136,604
Diagonal 3 function	2,158,912	1,709,338	1,986,158	167,563
Hager function	56,222	56,459	52,392	29,503
Generalized tridiagonal 1 function	10,290	10,568	10,387	23,914
Extended tridiagonal 1 function	5,194,192	5092	6,369,195	19,043
Extended TET function	33,240	34,180	20,100	9510
Diagonal 4 function	158,336	155,132	151,761	28,344
Diagonal 5 function	50,310	50,310	90	90
Extended Himmelblau function	25,572	25,572	25,515	29,640
Perturbed quadratic diagonal function	54,423,738	33,793,696	48,110,702	471,146
Quadratic QF1 function	10,439,311	8,879,307	10,198,882	330,341
Extended quadratic penalty QP1 function	10,551	10,221	11,047	14,320
Extended quadratic penalty QP2 function	3,443,064	3,283,726	3,243,403	82,215
Quadratic QF2 function	13,106,020	11,629,683	12,644,902	347,236
Extended quadratic exponential EP1 function	13,968	13,968	13,776	13,776
Extended tridiagonal 2 function	11,261	11,255	9185	6157
ARWHEAD function (CUTE)	1,670,783	1,521,208	1,661,430	127,659
Almost perturbed quadratic function	11,471,846	9,348,191	11,052,096	273,094
LIARWHD function (CUTE)	71,820,981	66,491,445	69,605,196	156,571
ENGVAL1 function (CUTE)	8182	8815	8406	17,065
Diagonal 6 function	1205	1262	122	122
COSINE function (CUTE)	176,032	485,018	120,982	69,682
Generalized quartic function	18,131	18,004	15,666	10,756
Diagonal 7 function	26,630	26,630	3275	3275
Diagonal 8 function	82,669	82,669	3654	3654
Full Hessian FH3 function	86,912	86,866	86,267	86,714
Diagonal 9 function	9,542,988	6,768,095	8,683,251	197,364
HIMMELH function (CUTE)	190	190	190	2803
Extended Rosenbrock	110	110	110	90

considered 10 different numerical experiments with the number of variables as follows: 100, 200, 300, 500, 700, 800, 1000, 1500, 2000 and 3000. The stopping conditions are the same as in previous tests. Also, the values of backtracking parameters for all methods are identical. Summary numerical results for all five methods, tested on 26 large-scale test functions, are presented in Tables 5, 6 and 7. Figures 3 and 4 show the

Table 3 Summary numerical results for the CPU time

Test function	CPU time			
	DHSDL	*DLSDL*	*MMDL*	*MLSCD*
Extended penalty function	11.469	11.844	12.141	12.266
Perturbed quadratic function	4703.016	4348.422	4617.125	92.375
Raydan 1 function	430.313	386.734	402.156	47.875
Raydan 2 function	0.656	0.922	0.313	0.219
Diagonal 1 function	1006.297	920.359	1186.688	90.953
Diagonal 3 function	2926.469	2296.703	2776.672	210.828
Hager function	87.500	88.078	80.781	36.203
Generalized tridiagonal 1 function	9.094	9.359	8.422	17.500
Extended tridiagonal 1 function	14,915.734	10.750	17,806.109	45.438
Extended TET function	20.766	20.797	12.313	5.641
Diagonal 4 function	19.422	17.781	17.406	5.359
Diagonal 5 function	83.000	82.625	0.359	0.297
Extended Himmelblau function	3.672	3.750	4.000	6.016
Perturbed quadratic diagonal function	21,479.750	12,615.844	19,096.688	137.219
Quadratic QF1 function	4994.438	3891.266	4762.094	106.500
Extended quadratic penalty QP1 function	2.922	2.828	3.359	4.313
Extended quadratic penalty QP2 function	513.609	499.391	479.750	31.516
Quadratic QF2 function	4023.453	3586.750	3756.359	96.453
Extended quadratic exponential EP1 function	4.281	4.844	4.172	4.188
Extended tridiagonal 2 function	3.031	3.172	2.641	1.984
ARWHEAD function (CUTE)	791.922	720.609	733.859	29.734
Almost perturbed quadratic function	5771.016	4021.641	5214.688	71.000
LIARWHD function (CUTE)	24,076.406	22,565.406	23,170.625	40.625
ENGVAL1 function (CUTE)	2.438	2.703	2.391	6.531
Diagonal 6 function	1.031	1.031	0.172	0.141
COSINE function (CUTE)	29.609	370.688	113.250	33.828
Generalized quartic function	4.813	5.750	6.000	3.156
Diagonal 7 function	17.922	17.094	2.813	2.547
Diagonal 8 function	50.344	48.484	3.000	2.625
Full Hessian FH3 function	20.938	20.391	21.500	20.094
Diagonal 9 function	2539.172	1570.594	2368.234	107.125
HIMMELH function (CUTE)	0.594	0.609	0.547	5.422
Extended Rosenbrock	0.203	0.234	0.250	0.172

performance of these methods relative to the number of iterations, number of function evaluations and the CPU time, respectively.

In Fig. 3(left), we see that all five methods successfully solve all the problems, and the *MLSCD* method is best in 42% of the test problems compared to the *DHSDL* (7%), *DLSDL* (7%), *MMDL* (23%), *H-BFGS-CG*1 (42%).

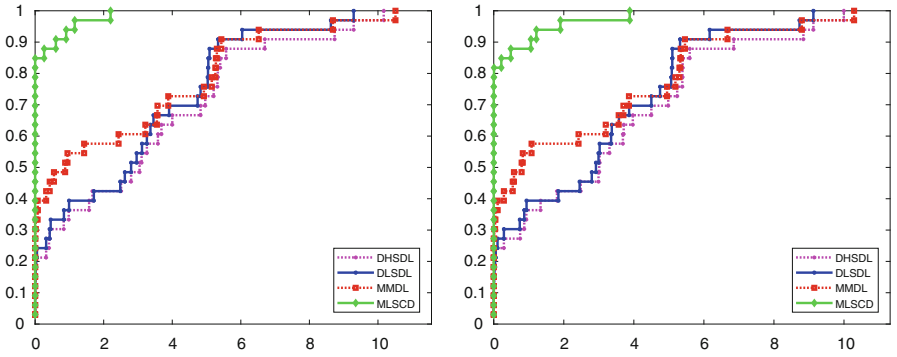

Fig. 1 Performance profiles based on the number of iterations (left) and function evaluations (right)

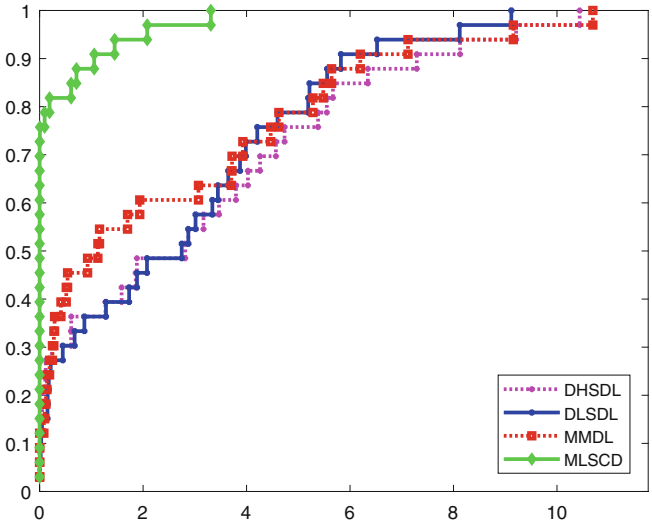

Fig. 2 Performance profile based on CPU time

Table 4 Average numerical outcomes for 33 test functions tested on 10 numerical experiments

Average performances	*DHSDL*	*DLSDL*	*MMDL*	*MLSCD*
Number of iterations	166,533.091	111,253.152	164,663.727	2,750.818
Number of function evaluations	5,999,361.576	4,731,348.879	5,685,599.970	92,866.970
CPU time (s)	2683.191	1762.044	2626.269	38.671

In Fig. 3(right), we see that all five methods successfully solve all test problems, and the *MLSCD* method is best in 38% of the test problems compared to the *DHSDL* (12%), *DLSDL* (15%), *MMDL* (30%), *H-BFGS-CG*1 (34%).

In Fig. 4, we see that all five methods successfully solve all the problems, and the *MLSCD* method is best in 50% of the test problems compared to the *DHSDL* (23%), *DLSDL* (12%), *MMDL* (35%), *H-BFGS-CG*1 (0%).

Table 5 Summary numerical results for the number of iterations

Test function	No. of iterations				
	DHSDL	*DLSDL*	*MMDL*	*MLSCD*	*H-BFGS-CG*1
Extended penalty function	1456	1492	1476	1165	13,055
Raydan 1 function	23,050	23,217	22,238	1976	32,385
Raydan 2 function	390	392	51	51	451
Diagonal 2 function	4758	5209	39,601	88,355	137
Diagonal 3 function	22,477	19,693	19,313	2238	15,706
Hager function	1692	1764	1623	1589	7496
Generalized tridiagonal 1 function	623	606	644	1542	7750
Extended tridiagonal 1 function	596,328	827	754,710	3429	330
Extended TET function	3411	3500	2061	1067	2357
Diagonal 5 function	25,070	25,070	40	40	25,043
Extended Himmelblau function	1306	1306	1301	1572	7201
Extended quadratic penalty QP1 function	563	585	599	917	2087
Extended quadratic exponential EP1 function	504	504	522	522	3187
Extended tridiagonal 2 function	1416	1417	1223	770	40,094
ENGVAL1 function (CUTE)	607	594	589	1126	5486
NONSCOMP function (CUTE)	3,145,116	30,559,018	34,334,110	39,002	8366
Diagonal 6 function	466	470	51	51	514
DIXON3DQ function (CUTE)	23,873,421	20,357,499	23,604,046	70,883	2344
COSINE function (CUTE)	31,671	2875	1389	2487	113
BIGGSB1 function (CUTE)	21,483,005	18,112,456	21,231,110	64,815	2270
Generalized quartic function	1245	1178	979	1310	378
Diagonal 7 function	5454	5454	562	562	43
Diagonal 8 function	17,891	17,891	579	579	13,254
Full Hessian FH3 function	3702	3699	3680	3680	1503
HIMMELH function (CUTE)	90	90	90	68	20
Extended Rosenbrock	50	50	50	40	40

Table 8 contains the average number of iterations, the average CPU time, and the average number of function evaluations for all 260 numerical experiments.

The results in Table 8 show remarkable progress in reducing the number of iterations and the number of function evaluations using the *H-BFGS-CG*1 method. Compared to the *DHSDL*, *DLSDL*, *MMDL* and *MLSCD* methods, the *H-BFGS-CG*1 method achieved a lower average number of iterative steps, and also a lower average number of function evaluations compared to the *DHSDL*, *DLSDL* and *MMDL* methods.

Table 6 Summary numerical results for the number of function evaluations

Test function	No. of function evaluations				
	DHSDL	*DLSDL*	*MMDL*	*MLSCD*	*H-BFGS-CG*1
Extended penalty function	39,354	40,173	40,016	33,458	508,837
Raydan 1 function	462,262	465,590	449,908	39,859	1,394,768
Raydan 2 function	970	975	112	112	1192
Diagonal 2 function	9526	10,428	151,063	176,720	2139
Diagonal 3 function	679,418	589,965	585,640	66,898	939,249
Hager function	23,955	25,156	23,148	24,396	222,414
Generalized tridiagonal 1 function	10,239	10,064	10,728	26,171	260,172
Extended tridiagonal 1 function	3,140,661	4912	3,845,293	16,675	5800
Extended TET function	34,150	35,020	20,660	9726	50,201
Diagonal 5 function	50,310	50,310	90	90	50,298
Extended Himmelblau function	24,774	24,774	24,679	28,878	293,347
Extended quadratic penalty QP1 function	9291	9644	9828	14,096	57,012
Extended quadratic exponential EP1 function	13,734	13,734	14,131	14,131	171,947
Extended tridiagonal 2 function	11,253	11,336	9613	6304	1,220,619
ENGVAL1 function (CUTE)	8520	8769	8258	17,416	140,092
NONSCOMP function (CUTE)	55,575,511	531,637,119	618,136,564	698,515	340,659
Diagonal 6 function	1164	1171	112	112	1372
DIXON3DQ function (CUTE)	194,329,593	161,402,916	193,158,500	597,334	42,659
COSINE function (CUTE)	950,303	105,989	24,834	84,165	9956
BIGGSB1 function (CUTE)	17,4871,424	143,604,806	173,740,005	546,352	41,301
Generalized quartic function	11,696	10,944	10,999	10,978	7574
Diagonal 7 function	27,714	27,714	3370	3370	2209
Diagonal 8 function	89,819	89,819	3238	3238	146,099
Full Hessian FH3 function	120,485	120,291	119,251	119,385	99,571
HIMMELH function (CUTE)	190	190	190	1927	1800
Extended Rosenbrock	110	110	110	90	90

However, if we observe the average CPU time for all five methods, we can conclude that the *H-BFGS-CG*1 method is slow. The conclusion is the same if we look at graphs arranged in Fig. 4.

From the above, we can give a final conclusion that the *MLSCD* method is the most efficient in terms of all three underlying metrics: number of iterations, number of function evaluations and the CPU time.

7 Conclusions

We propose three efficient conjugate gradient methods for unconstrained optimization problems, in which the search directions always satisfy the sufficient descent

Table 7 Summary numerical results for the CPU time

Test function	CPU time				
	DHSDL	*DLSDL*	*MMDL*	*MLSCD*	*H-BFGS-CG*1
Extended penalty function	1.891	1.922	1.938	1.531	7745.109
Raydan 1 function	17.016	17.906	17.031	2.406	28,628.781
Raydan 2 function	0.141	0.219	0.063	0.063	177.156
Diagonal 2 function	1.688	2.063	15.641	30.453	66.547
Diagonal 3 function	99.109	82.953	81.938	7.750	7598.953
Hager function	5.078	5.297	4.734	4.406	5111.313
Generalized tridiagonal 1 function	1.203	1.359	1.203	3.094	2484.188
Extended tridiagonal 1 function	1099.453	1.531	1369.641	5.047	130.000
Extended TET function	2.313	2.500	1.547	0.906	935.063
Diagonal 5 function	9.656	10.094	0.078	0.109	9798.609
Extended Himmelblau function	0.719	0.719	0.828	0.750	2737.547
Extended quadratic penalty QP1 function	0.594	0.500	0.578	0.656	688.172
Extended quadratic exponential EP1 function	0.641	0.766	0.719	0.641	1207.734
Extended tridiagonal 2 function	0.641	0.828	0.813	0.328	12,771.406
ENGVAL1 function (CUTE)	0.469	0.578	0.547	0.984	1830.703
NONSCOMP function (CUTE)	732.359	32,109.703	10,681.109	8.969	1647.484
Diagonal 6 function	0.281	0.344	0.078	0.078	210.125
DIXON3DQ function (CUTE)	10,948.516	9275.313	12,082.484	28.031	1292.516
COSINE function (CUTE)	42.063	5.859	2.375	6.125	61.828
BIGGSB1 function (CUTE)	9884.078	8730.078	10,221.297	24.469	891.500
Generalized quartic function	0.719	1.078	0.609	0.594	159.453
Diagonal 7 function	2.297	2.375	0.438	0.469	20.453
Diagonal 8 function	6.375	6.156	0.438	0.500	5002.109
Full Hessian FH3 function	6.922	7.078	6.688	6.781	1117.875
HIMMELH function (CUTE)	0.328	0.391	0.078	0.500	9.141
Extended Rosenbrock	0.078	0.094	0.297	0.141	16.984

condition. These methods are derived using various modifications on the conjugate gradients direction d_k of the form (7) or (8) or using various combinations of scaling parameters β_k. Comparative criteria are the number of iterative steps, spent CPU time, and the number of the function evaluations. Based on the backtracking line search conditions, we show that our methods are strongly convergent for the uniformly convex functions and globally convergent for general functions. Numerical results illustrate that the proposed methods can outperform the existing ones. Also, these results show that a hybridization of a quasi-Newton method with a *CG* method reduces the number of iterations and the number of function evaluations, but increases the CPU time.

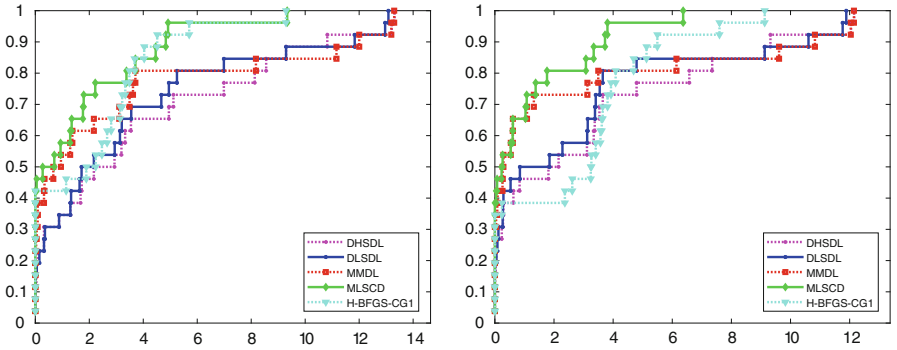

Fig. 3 Performance profiles based on the number of iterations (left) and function evaluations (right)

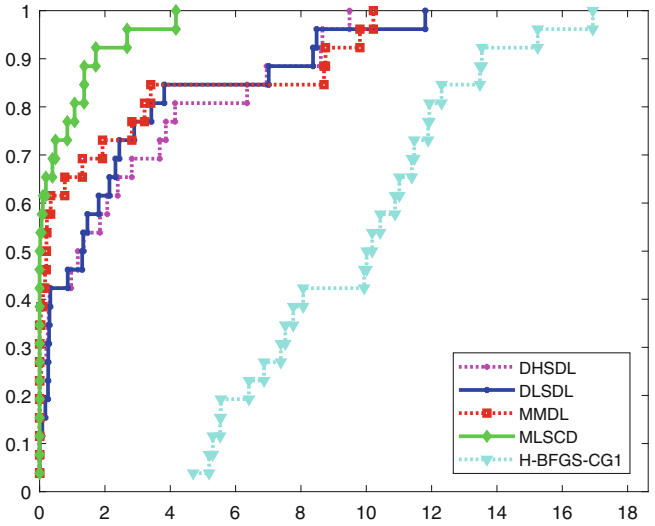

Fig. 4 Performance profile based on CPU time

Table 8 Average numerical outcomes for 26 test functions tested on 10 numerical experiments

Avg. performances	*DHSDL*	*DLSDL*	*MMDL*	*MLSCD*	*H-BFGS-CG*1
Number of iterations	1,894,067.77	2,659,494.46	3,077,793.73	11,147.54	7,369.62
No. of fun.evaluation	16,557,554.85	32,242,381.50	38,091,936.15	97,707.54	231,206.81
CPU time (s)	879.41	1933.37	1326.66	5.22	3551.57

Acknowledgements The first author gratefully acknowledge support from the Research Project 174013 of the Serbian Ministry of Science.

References

1. Fletcher, R., Reeves, C.M.: Function minimization by conjugate gradients. Comput. J. **7**(2), 149–154 (1964)
2. Fletcher, R.: Practical Methods of Optimization. Unconstrained Optimization, vol. 1. Wiley, New York (1987)
3. Dai, Y.H., Yuan, Y.: A nonlinear conjugate gradient method with a strong global convergence property. SIAM J. Optim. **10**(1), 177–182 (1999)
4. Hestenes, M.R., Stiefel, E.L.: Methods of conjugate gradients for solving linear systems. J. Res. Natl. Bur. Stand. **49**(6), 409–436 (1952)
5. Polak, E., Ribiere, G.: Note sur la convergence des mthodes de directions conjuguées. Rev. Francaise Imformat Recherche Opertionelle **16**, 35–43 (1969)
6. Polyak, B.T.: The conjugate gradient method in extreme problems. U.S.S.R. Comput. Math. Math. Phys. **9**, 94–112 (1969)
7. Liu, Y., Storey, C.: Efficient generalized conjugate gradient algorithms, part 1: theory. J. Optim. Theory Appl. **69**(1), 129–137 (1991)
8. Yang, X., Luo, Z., Dai, X.: A global convergence of LS-CD hybrid conjugate gradient method. In: Advances in Numerical Analysis 2013. Hindawi Publishing Corporation, Article ID 517452, 5 pp. (2013)
9. Dai, Y.H., Liao, L.Z.: New conjugacy conditions and related nonlinear conjugate gradient methods. Appl. Math. Optim. **43**(1), 87–101 (2001)
10. Zheng, Y., Zheng, B.: Two new Dai–Liao-type conjugate gradient methods for unconstrained optimization problems. J. Optim. Theory Appl. **175**, 502–509 (2017)
11. Andrei, N.: An acceleration of gradient descent algorithm with backtracking for unconstrained optimization. Numer. Algorithms **42**, 63–73 (2006)
12. Stanimirović, P.S., Miladinović, M.B.: Accelerated gradient descent methods with line search. Numer. Algorithms **54**, 503–520 (2010)
13. Touati-Ahmed, D., Storey, C.: Efficient hybrid conjugate gradient techniques. J. Optim. Theory Appl. **64**(2), 379–397 (1990)
14. Zhang, L., Zhou, W.: Two descent hybrid conjugate gradient methods for optimization. J. Comput. Appl. Math. **216**, 251–264 (2008)
15. Gilbert, J.C., Nocedal, J.: Global convergence properties of conjugate gradient methods for optimization. SIAM J. Optim. **2**(1), 21–42 (1992)
16. Dai, Y.H., Yuan, Y.: An efficient hybrid conjugate gradient method for unconstrained optimization. Ann. Oper. Res. **103**, 33–47 (2001)
17. Hager, W.W., Zhang, H.: A new conjugate gradient method with guaranteed descent and an efficient line search. SIAM J. Optim. **16**, 170–192 (2005)
18. Zhang, L., Zhou, W.J., Li, D.H.: A descent modified Polak–Ribiére–Polyak conjugate gradient method and its global convergence. IMA J. Numer. Anal. **26**, 629–640 (2006)
19. Zhang, L., Zhou, W.J., Li, D.H.: Global convergence of a modified Fletcher–Reeves conjugate method with Armijo-type line search. Numer. Math. **104**, 561–572 (2006)
20. Zhang, L.: Nonlinear Conjugate Gradient Methods for Optimization Problems. Ph.D. Thesis, College of Mathematics and Econometrics, Hunan University, Changsha, China (2006)
21. Ibrahim, M.A.H., Mamat, M., Leong, W.J.: The hybrid BFGS-CG method in solving unconstrained optimization problems. In: Abstract and Applied Analysis 2014. Hindawi Publishing Corporation. Article ID 507102, 6 pp. (2014)
22. Zoutendijk, G.: Nonlinear programming, computational methods. In: Abadie, J. (ed.) Integer and Nonlinear Programming, pp. 37–86. North-Holland, Amsterdam (1970)
23. Cheng, W.: A two-term PRP-based descent method. Numer. Funct. Anal. Optim. **28**(11–12), 1217–1230 (2007)
24. Luo, Y.Z., Tang, G.J., Zhou, L.N.: Hybrid approach for solving systems of nonlinear equations using chaos optimization and quasi-Newton method. Appl. Soft Comput. **8**(2), 1068–1073 (2008)
25. Han, L., Neumann, M.: Combining quasi-Newton and Cauchy directions. Int. J. Appl. Math. **12**(2), 167–191 (2003)
26. Baluch, B., Salleh, Z., Alhawarat, A., Roslan, U.A.M.: A new modified three-term conjugate gradient method with sufficient descent property and its global convergence. J. Math. 2017, Article ID 2715854, 12 pp. (2017)

27. Khanaiah, Z., Hmod, G.: Novel hybrid algorithm in solving unconstrained optimizations problems. Int. J. Novel Res. Phys. Chem. Math. **4**(3), 36–42 (2017)
28. Osman, W.F.H.W., Ibrahim, M.A.H., Mamat, M.: Hybrid DFP-CG method for solving unconstrained optimization problems. J. Phys. Conf. Ser. **890**(2017), 012033 (2017)
29. Byrd, R.H., Nocedal, J.: A tool for the analysis of quasi-Newton methods with application to unconstrained minimization. SIAM J. Numer. Anal. **26**(3), 727–739 (1989)
30. Andrei, N.: An Unconstrained optimization test functions collection. Adv. Model. Optim. **10**(1), 147–161 (2008)
31. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Math. Program. **91**, 201–213 (2002)