Another accelerated conjugate gradient algorithm with guaranteed descent and conjugacy conditions for large-scale unconstrained optimization

Neculai Andrei
Research Institute for Informatics,
Center for Advanced Modeling and Optimization,
8-10, Averescu Avenue, Bucharest 1, Romania,
E-mail: nandrei@ici.ro

Abstract. In this paper we suggest another accelerated conjugate gradient algorithm that for all $k \geq 0$ both the descent and the conjugacy conditions are guaranteed. The search direction is selected as $d_{k+1} = -\theta_k g_{k+1} + (y_k^T g_{k+1} / y_k^T s_k) s_k - t_k (s_k^T g_{k+1} / y_k^T s_k) s_k$, where $g_{k+1} = \nabla f(x_{k+1})$, $s_k = x_{k+1} - x_k$. The coefficients θ_k and t_k in this linear combination are selected in such a way that both the descent and the conjugacy condition are satisfied at every iteration. It is shown that both for uniformly convex functions and for general nonlinear functions the algorithm with strong Wolfe line search generates directions bounded away from infinity. The algorithm uses an acceleration scheme modifying the steplength α_k in such a manner as to improve the reduction of the function values along the iterations. Numerical comparisons with some conjugate gradient algorithms using a set of 75 unconstrained optimization problems with different dimensions, some of them from the CUTE library, show that the computational scheme outperform the known conjugate gradient algorithms like Hestenes and Stiefel; Polak, Ribière and Polyak; Dai and Yuan or the hybrid Dai and Yuan; CG_DESCENT with Wolfe line search by Hager and Zhang, as well as the quasi-Newton L-BFGS by Liu and Nocedal.

Keywords: Conjugate gradient, Wolfe line search, descent condition, conjugacy condition, unconstrained optimization.
AMS subject classifications: 49M20, 65K05, 90C30

1. Introduction

For solving the unconstrained optimization problems

$$\min_{x \in R^n} f(x), \tag{1.1}$$

where $f : R^n \to R$ is a continuously differentiable function, bounded from below, one of the most elegant and probably the simplest methods are the conjugate gradient methods. For solving this problem, starting from an initial guess $x_0 \in R^n$, a nonlinear conjugate gradient method, generates a sequence $\{x_k\}$ as:

$$x_{k+1} = x_k + \alpha_k d_k, \tag{1.2}$$

where $\alpha_k > 0$ is obtained by line search, and the directions d_k are generated as:

$$d_{k+1} = -g_{k+1} + \beta_k d_k, \quad d_0 = -g_0. \tag{1.3}$$

In (1.3) β_k is known as the conjugate gradient parameter, $s_k = x_{k+1} - x_k$ and $g_k = \nabla f(x_k)$. The search direction d_k, assumed to be a descent one, plays the main role in these methods. On the other hand, the stepsize α_k guarantees the global convergence in some cases and is

crucial in efficiency. Different conjugate gradient algorithms correspond to different choices for the scalar parameter β_k. Plenty of conjugate gradient methods are known and an excellent survey of these methods with a special attention on their global convergence is given by Hager and Zhang [26]. Line search in the conjugate gradient algorithms often is based on the standard Wolfe conditions [41, 42]

$$f(x_k + \alpha_k d_k) - f(x_k) \le \rho \alpha_k g_k^T d_k, \tag{1.4}$$

$$g(x_k + \alpha_k d_k)^T d_k \ge \sigma g_k^T d_k, \tag{1.5}$$

where d_k is supposed to be a descent direction and $0 < \rho \le 1/2 < \sigma < 1$.

A numerical comparison of conjugate gradient algorithms (1.2) and (1.3) with Wolfe line search, for different formulae of parameter β_k computation, including the Dolan and Moré performance profile [19], is given in [6].

If the initial direction d_0 is selected as $d_0 = -g_0$, and the objective function to be minimized is a convex quadratic function

$$f(x) = \frac{1}{2} x^T A x + b^T x + c \tag{1.6}$$

and the exact line searches are used, that is

$$\alpha_k = \arg\min_{\alpha > 0} f(x_k + \alpha d_k), \tag{1.7}$$

then the conjugacy condition

$$d_i^T A d_j = 0 \tag{1.8}$$

holds for all $i \ne j$. This relation (1.8) is the original condition used by Hestenes and Stiefel [27] to derive the conjugate gradient algorithms, mainly for solving symmetric positive-definite systems of linear equations. Using (1.3) and (1.6)-(1.8) it can be shown that x_{k+1} is the minimum of the quadratic function (1.6) in the subspace $x_k + span\{g_1, g_2, \ldots, g_k\}$ and the gradients $g_1, g_2, \ldots, g_k$ are mutually orthogonal unless that $g_k = 0$ [20]. It follows that for convex quadratic functions the solution will be found after at most n iterations. Powell [38] shown that if the initial search direction is not g_0 then even for quadratic functions (1.6) the conjugate gradient algorithms does not terminate within a finitely number of iterations. It is well known that the conjugate gradient algorithm converges at least linearly [34]. An upper bound for the rate of convergence of conjugate gradient algorithms was given by Yuan [43].

Let us denote $y_k = g_{k+1} - g_k$. For a general nonlinear twice differential function f, by the mean value theorem, there exists some $\xi \in (0,1)$ such that

$$d_{k+1}^T y_k = \alpha_k d_{k+1}^T \nabla^2 f(x_k + \xi \alpha_k d_k) d_k. \tag{1.9}$$

Therefore, it seems reasonable to replace (1.8) with the following conjugacy condition

$$d_{k+1}^T y_k = 0. \tag{1.10}$$

In order to accelerate the conjugate gradient algorithm Perry [33] (see also Shanno [39]) extended the conjugacy condition by incorporating the second order information. He used the secant condition $H_{k+1} y_k = s_k$, where H_k is a symmetric approximation to the inverse Hessian. Since for quasi-Newton method the search direction d_{k+1} is computed as $d_{k+1} = -H_{k+1} g_{k+1}$, it follows that

$$d_{k+1}^T y_k = -(H_{k+1} g_{k+1})^T y_k = -g_{k+1}^T (H_{k+1} y_k) = -g_{k+1}^T s_k,$$

thus obtaining a new conjugacy condition. Recently, Dai and Liao [15] extended this condition and suggested the following new conjugacy condition

$$d_{k+1}^T y_k = -v g_{k+1}^T s_k, \tag{1.11}$$

where $v \ge 0$ is a scalar.

Conjugate gradient algorithms are based on the conjugacy condition. To minimize a convex quadratic function in a subspace spanned by a set of mutually conjugate directions is equivalent to minimize this function along each conjugate direction in turn. This is a very good idea, but the performance of these algorithms is dependent on the accuracy of the line search. However, in conjugate gradient algorithms we always use inexact line search. Hence, when the line search is not exact, the "pure" conjugacy condition (1.10) may have disadvantages. Therefore, it seems more reasonable to consider in conjugate gradient algorithms the conjugacy condition (1.11). When the algorithm is convergent observe that $g_{k+1}^T s_k$ tends to zero along the iterations, and therefore conjugacy condition (1.11) tends to the pure conjugacy condition (1.10).

Conjugate gradient algorithm (1.2) and (1.3) with exact line search always satisfy the condition $g_{k+1}^T d_{k+1} = -\|g_{k+1}\|^2$ which is in a direct connection with the sufficient descent condition

$$g_{k+1}^T d_{k+1} \le -w\|g_{k+1}\|^2 \tag{1.12}$$

for some positive constant $w > 0$. The sufficient descent condition has been used often in the literature to analyze the global convergence of the conjugate gradient algorithms with inexact line search based on the strong Wolfe conditions. The sufficient descent condition is not needed in the convergence analyses of the Newton or quasi-Newton algorithms. However, it is necessary for the global convergence of conjugate gradient algorithms [18].

Using (1.11) Dai and Liao [15] obtained a new conjugate gradient algorithm

$$\beta_k^{DL} = \frac{g_{k+1}^T (y_k - v s_k)}{y_k^T s_k}. \tag{1.13}$$

For an exact line search we see that g_{k+1} is orthogonal to s_k. Therefore, for an exact line search, the DL method reduces to the HS method. Observe that due to the Powell's example, the DL method may not converge for an exact line search. To overcome this and to ensure convergence Dai and Liao modified their formula as

$$\beta_k^{DL+} = \max\left\{\frac{g_{k+1}^T y_k}{y_k^T s_k}, 0\right\} - v\frac{g_{k+1}^T s_k}{y_k^T s_k}. \tag{1.14}$$

If the level set $S = \left\{x \in R^n : f(x) \le f(x_0)\right\}$ is bounded and the gradient $\nabla f(x)$ is Lipschitz continuous on S, and if d_k satisfies the sufficient descent condition (1.12), it is shown in [15] that DL+ implemented with a strong Wolfe line search is globally convergent. Numerical results are reported in [15] for $v = 0.1$ and $v = 1$. However, for different choices of v, the numerical results are quite different.

In this paper we suggest a new conjugate gradient algorithm that for all $k > 0$ both the descent and the conjugacy conditions are guaranteed. In section 2 we present the search direction, as well as the main ingredients for its computation. The search direction is selected as a linear combination of $-g_{k+1}$ and s_k, where the coefficients in this linear combination are selected in such a way that both the descent and the conjugacy condition to be satisfied at every iteration. In section 3 we prove the convergence of the algorithm. It is shown that both for uniformly convex functions and for general nonlinear functions the corresponding algorithm with strong Wolfe line search generates directions bounded away from infinity. Section 4 is devoted to present the algorithm in its accelerated version. The idea of this computational scheme is to take advantage that the step lengths α_k in conjugate gradient algorithms are very different from 1. Therefore, we suggest we modify α_k in such a manner as to improve the reduction of the function values along the iterations. In section 5 some numerical experiments and performance profiles of Dolan-Moré corresponding to this new conjugate gradient algorithm are given. The performance profiles correspond to a set of 75 unconstrained optimization problems presented in [1]. Each problem was tested 10 times for a

gradually increasing number of variables: $n = 1000, 2000, \ldots, 10000$. It is shown that this new conjugate gradient algorithm outperforms the classical Hestenes and Stiefel [27], Dai and Yuan [17], Polak, Ribière and Polyak [35, 36], hybrid Dai and Yuan [17] conjugate gradient algorithms, the CG_DESCENT conjugate gradient algorithm with Wolfe line search by Hager and Zhang [25] and also L-BFGS by Liu and Nocedal [29].

2. Conjugate gradient algorithm with guaranteed descent and conjugacy conditions

For solving the minimization problem (1.1) let us consider the following conjugate gradient algorithm

$$x_{k+1} = x_k + \alpha_k d_k, \tag{2.1}$$

where $\alpha_k > 0$ is obtained by the Wolfe line search, and the directions d_k are generated as:

$$d_{k+1} = -\theta_k g_{k+1} + \beta_k s_k, \tag{2.2}$$

$$\beta_k = \frac{y_k^T g_{k+1} - t_k s_k^T g_{k+1}}{y_k^T s_k}, \tag{2.3}$$

$d_0 = -g_0$, where θ_k and t_k are scalar parameters which follows to be determined. Observe that in d_{k+1}, given by (2.2), g_{k+1} is scaled by parameter θ_k and the parameter t_k is changed at every iteration. Algorithms of this form, or variations of them, have been studied by many authors. For example, Andrei [3,4,5] considers a preconditioned conjugate gradient algorithm where the preconditioner is a scaled memoryless BFGS matrix and the parameter scaling the gradient is selected as the spectral gradient. On the other hand Birgin and Martínez [11] suggested a spectral conjugate gradient method, where $\theta_k = s_k^T s_k / s_k^T y_k$. Yuan and Stoer [44] studied the conjugate gradient algorithm on a subspace, where the search direction d_{k+1} at the $k-$th iteration ($k \geq 1$) is taken from the subspace $span\{g_{k+1}, d_k\}$. Observe that if for every $k \geq 1$, $\theta_k = 1$ and $t_k = v$, then (2.2) reduces to the Dai and Liao direction (1.13).

In our algorithm for all $k \geq 0$ the scalar parameters θ_k and t_k in (2.2) are determined in such a way that both the descent and the conjugacy conditions are satisfied. Therefore, from the *descent condition* (1.12) we have

$$-\theta_k \|g_{k+1}\|^2 + \frac{(y_k^T g_{k+1})(s_k^T g_{k+1})}{y_k^T s_k} - t_k \frac{(s_k^T g_{k+1})^2}{y_k^T s_k} = -w\|g_{k+1}\|^2 \tag{2.4}$$

and from the *conjugacy condition* (1.11)

$$-\theta_k y_k^T g_{k+1} + y_k^T g_{k+1} - t_k s_k^T g_{k+1} = -v(s_k^T g_{k+1}), \tag{2.5}$$

where $v > 0$ and $w > 0$ are known scalar parameters. Observe that in (2.4) we modified the classical sufficient descent condition (1.12) with equality. It is worth saying that the main condition in any conjugate gradient algorithm is the descent condition $g_k^T d_k < 0$ or the sufficient descent condition (1.12). In our algorithm we have selected w close to 1. This is enough a reasonable value. For example, Hager and Zhang [25] show that in their CG_DESCENT algorithm $w = 7/8$. On the other hand, the conjugacy condition (1.10) or its modification (1.11) is not so stringent. In fact very few conjugate gradient algorithms satisfy this condition. For example, the Hestenes and Stiefel (HS) algorithm has this property that the pure conjugacy condition always holds, independent of the line search.

If $v = 0$, then (2.5) is the "pure" conjugacy condition. However, in our algorithm in order to accelerate the algorithm and to incorporate the second order information we take $v > 0$.

Now, let us define

$$\bar{\Delta}_k = (y_k^T g_{k+1})(s_k^T g_{k+1}) - \|g_{k+1}\|^2 (y_k^T s_k), \tag{2.6}$$

$$\Delta_k = (s_k^T g_{k+1})\bar{\Delta}_k, \tag{2.7}$$

$$a_k = v(s_k^T g_{k+1}) + y_k^T g_{k+1}, \tag{2.8}$$

$$b_k = w\|g_{k+1}\|^2 (y_k^T s_k) + (y_k^T g_{k+1})(s_k^T g_{k+1}). \tag{2.9}$$

Supposing that $\Delta_k \neq 0$ and $y_k^T g_{k+1} \neq 0$, then from the linear algebraic system given by (2.4) and (2.5) we get

$$t_k = \frac{b_k (y_k^T g_{k+1}) - a_k (y_k^T s_k)\|g_{k+1}\|^2}{\Delta_k}, \tag{2.10}$$

$$\theta_k = \frac{a_k - t_k (s_k^T g_{k+1})}{y_k^T g_{k+1}}, \tag{2.11}$$

with which the parameter β_k and the direction d_{k+1} can immediately be computed.

Proposition 2.1. *If*

$$\frac{1}{2} < \sigma \leq \frac{\|g_{k+1}\|^2}{|y_k^T g_{k+1}| + \|g_{k+1}\|^2}, \tag{2.12}$$

then for all $k \geq 1$, $\bar{\Delta}_k < 0$.

Proof. Observe that

$$s_k^T g_{k+1} = s_k^T y_k + s_k^T g_k < s_k^T y_k. \tag{2.13}$$

The Wolfe condition (1.5) gives

$$g_{k+1}^T s_k \geq \sigma g_k^T s_k = -\sigma y_k^T s_k + \sigma g_{k+1}^T s_k. \tag{2.14}$$

Since $\sigma < 1$, we can rearrange (2.14) to obtain

$$g_{k+1}^T s_k \geq \frac{-\sigma}{1-\sigma} y_k^T s_k. \tag{2.15}$$

Now, combining this lower bound for $g_{k+1}^T s_k$ with the upper bound (2.13) we get

$$|g_{k+1}^T s_k| \leq |y_k^T s_k| \max\left\{1, \frac{\sigma}{1-\sigma}\right\}. \tag{2.16}$$

Since $\sigma > 1/2$, from (2.16) we can write

$$|g_{k+1}^T s_k| < \frac{\sigma}{1-\sigma} |y_k^T s_k|. \tag{2.17}$$

If (2.12) is true, then

$$\frac{\sigma}{1-\sigma} |y_k^T g_{k+1}| \leq \|g_{k+1}\|^2.$$

Again, observe that the Wolfe condition gives $y_k^T s_k > 0$ (if $g_k \neq 0$). Therefore,

$$\frac{\sigma}{1-\sigma} |y_k^T s_k| |g_{k+1}^T y_k| \leq |y_k^T s_k| \|g_{k+1}\|^2. \tag{2.18}$$

From (2.17) we can write

$$|s_k^T g_{k+1}| |y_k^T g_{k+1}| < \frac{\sigma}{1-\sigma} |y_k^T s_k| |y_k^T g_{k+1}| \leq |y_k^T s_k| \|g_{k+1}\|^2, \tag{2.19}$$

i.e. $\bar{\Delta}_k < 0$ for all $k \geq 1$. ■

Some remarks are in order.

1) Suppose that $g_k \neq 0$ for all $k \geq 1$, otherwise a stationary point is obtained. From (2.4) we can write

$$\beta_k (s_k^T g_{k+1}) = (\theta_k - w) \|g_{k+1}\|^2 . \tag{2.20}$$

Since $s_k^T g_{k+1}$ tends to zero (d_k is a descent direction) it follows that θ_k tends to $w > 0$, and hence $\theta_k > 0$. Therefore, there exists a constant $c_1 > 0$ such that $0 < \theta_k < c_1$.

2) From (2.12) observe that $\sigma < 1$. Besides, since for all k, $\bar{\Delta}_k < 0$ then there exists a positive constant $c_2 > 0$ such that $|\bar{\Delta}_k| > c_2$. Also, in order to have $\bar{\Delta}_k < 0$ the parameter σ in the second Wolfe condition (1.5) is modified as in (2.12). Observe that since $g_k^T d_k = -w\|g_k\|^2 < 0$, i.e. d_k is a descent direction, it follows that $|g_{k+1}^T y_k| \to \|g_{k+1}\|^2$. Therefore $\sigma \to 1/2$, i.e. $0 < \rho < \sigma < 1$, since usually ρ is selected enough small to ensure the reduction of function values along the iterations.

3) By the Wolfe conditions we have $y_k^T s_k = (g_{k+1} - g_k)^T s_k \geq (\sigma - 1) g_k^T s_k$. But from the descent condition (2.4) it follows that $g_k^T s_k = \alpha_k g_k^T d_k = -\alpha_k w \|g_k\|^2$. It is easy to prove that there exist a positive scalar ω such that $\alpha_k \geq \omega > 0$ (see [25], Lemma 2.1). Hence,

$$y_k^T s_k \geq (\sigma - 1) g_k^T s_k = -\alpha_k (\sigma - 1) w \|g_k\|^2 \geq \omega w (1 - \sigma) \|g_k\|^2 > 0.$$

Therefore, if $g_k \neq 0$, then by Wolfe conditions, for all $k \geq 0$, $y_k^T s_k > 0$. On the other hand, $w > 0$, and since $s_k^T g_{k+1}$ tends to zero, from (2.19) it follows that

$$w \|g_{k+1}\|^2 (y_k^T s_k) > |y_k^T g_{k+1}| |s_k^T g_{k+1}| .$$

Therefore, $b_k > 0$ for all $k \geq 0$ and is bounded away from zero.

3. Convergence analysis

In this section we analyze the convergence of the algorithm (2.1) and (2.2), where θ_k and β_k are given by (2.11) and (2.3) respectively, and $d_0 = -g_0$. In the following we consider that $g_k \neq 0$ for all $k \geq 1$, otherwise a stationary point is obtained. Assume that:

(i) *The level set* $S = \{x \in R^n : f(x) \leq f(x_0)\}$ *is bounded, i.e. there exists a positive constant* $B > 0$ *such that for all* $x \in S$, $\|x\| \leq B$.

(ii) *In a neighborhood* N *of* S, *the function* f *is continuously differentiable and its gradient is Lipschitz continuous, i.e. there exists a constant* $L > 0$ *such that* $\|\nabla f(x) - \nabla f(y)\| \leq L \|x - y\|$, *for all* $x, y \in N$.

Under these assumptions on f there exists a constant $\Gamma \geq 0$ such that $\|\nabla f(x)\| \leq \Gamma$ for all $x \in S$. In order to prove the global convergence, we assume that the step size α_k in (2.1) is obtained by the strong Wolfe line search, that is,

$$f(x_k + \alpha_k d_k) - f(x_k) \leq \rho \alpha_k g_k^T d_k , \tag{3.1}$$

$$|g(x_k + \alpha_k d_k)^T d_k| \leq \sigma g_k^T d_k . \tag{3.2}$$

where ρ and σ are positive constants such that $0 < \rho \leq \sigma < 1$.

For the conjugate gradient algorithm (2.2) where θ_k and β_k are given by (2.11) and (2.3) respectively, with strong Wolfe line search, the following Lemmas can be proved. The first two Lemmas were established by Zoutendijk [45] and Wolfe [41, 42], but for completeness we present them here (see also [28]).

Lemma 3.1. *Suppose that the assumptions (i) and (ii) hold. Consider that* α_k *is obtained by the strong Wolfe line search (3.1) and (3.2) and the descent condition hold. Then*

$$\sum_{k=0}^{\infty} -\alpha_k g_k^T d_k < \infty. \tag{3.3}$$

Proof. From (3.1) and the descent condition (2.4) we have that

$$f_{k+1} - f_k \leq \rho \alpha_k g_k^T d_k \leq 0. \tag{3.4}$$

Therefore, $\{f_k\}$ is a decreasing sequence. Since f is bounded below there exist a constant f^* such that

$$\lim_{k\to\infty} f_k = f^*. \tag{3.5}$$

From (3.5) it follows that

$$\sum_{k=0}^{\infty}(f_k - f_{k+1}) = \lim_{n\to\infty}\sum_{k=0}^{n}(f_k - f_{k+1}) = \lim_{n\to\infty}(f_0 - f_{n+1}) = f_0 - f^*.$$

Hence, $\sum_{k=0}^{\infty}(f_k - f_{k+1}) < +\infty$. From (3.4) it follows (3.3). ■

Lemma 3.2. *Consider the conjugate gradient algorithm (2.2) where* θ_k *and* β_k *are given by (2.11) and (2.3) respectively and* α_k *is obtained by the strong Wolfe line search (3.1) and (3.2). Suppose that the assumptions (i) and (ii), as well as the descent condition hold. Then*

$$\sum_{k=0}^{\infty} \frac{(g_k^T d_k)^2}{\|d_k\|^2} < +\infty. \tag{3.6}$$

Proof. From the strong Wolfe line search and the assumptions (i) and (ii), we get

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

Therefore,

$$\alpha_k \geq \frac{-(1-\sigma) g_k^T d_k}{L\|d_k\|^2}. \tag{3.7}$$

We know that for all k, $g_k^T d_k < 0$. Hence, using Lemma 3.1 we get

$$\sum_{k=0}^{\infty} \frac{(g_k^T d_k)^2}{\|d_k\|^2} \leq \frac{L}{1-\sigma}\sum_{k=0}^{\infty}(-\alpha_k g_k^T d_k) < +\infty. \quad ■$$

Observe that (3.6), known as the Zoutendijk condition, is obtained under the assumptions that the strong Wolfe line search hold and that d_k is a descent direction, independent by its form.

Lemma 3.3. *Consider the conjugate gradient algorithm (2.2) where* θ_k *and* β_k *are given by (2.11) and (2.3) respectively and* α_k *is obtained by the strong Wolfe line search (3.1) and (3.2). Suppose that the assumptions (i) and (ii) hold, and* $\theta_k \in [0, 2w]$. *Then either*

$$\liminf_{k\to\infty} \|g_k\| = 0 \tag{3.8}$$

or

$$\sum_{k=0}^{\infty}\frac{\|g_k\|^4}{\|d_k\|^2}<\infty. \tag{3.9}$$

Proof. Squaring the both terms of $d_{k+1}+\theta_k g_{k+1}=\beta_k s_k$ we get

$$\|d_{k+1}\|^2+\theta_k^2\|g_{k+1}\|^2+2\theta_k d_{k+1}^T g_{k+1}=\beta_k^2\|s_k\|^2.$$

But, from (2.4) $d_{k+1}^T g_{k+1}=-w\|g_{k+1}\|^2$. Therefore,

$$\|d_{k+1}\|^2=-(\theta_k^2-2\theta_k w)\|g_{k+1}\|^2+\beta_k^2\|s_k\|^2. \tag{3.10}$$

Observe that for $\theta_k\in[0,2w]$, $\theta_k^2-2\theta_k w\le 0$ and is bounded below by $-w^2$. On the other hand from (2.2) we have $g_{k+1}^T d_{k+1}-\beta_k g_{k+1}^T s_k=-\theta_k\|g_{k+1}\|^2$. Using the strong Wolfe line search we get

$$\left|g_{k+1}^T d_{k+1}\right|+\sigma\left|\beta_k\right|\left|g_k^T s_k\right|\ge\theta_k\|g_{k+1}\|^2. \tag{3.11}$$

Now, considering the following inequality $(a+\sigma b)^2\le(1+\sigma^2)(a^2+b^2)$ true for all $a,b,\sigma\ge 0$, with $a=\left|g_{k+1}^T d_{k+1}\right|$ and $b=\left|\beta_k\right|\left|g_k^T s_k\right|$ after some algebra we get

$$(g_{k+1}^T d_{k+1})^2+\beta_k^2(g_k^T s_k)^2\ge e\|g_{k+1}\|^4, \tag{3.12}$$

where $e=\theta_k^2/(1+\sigma^2)$ is a positive constant.

Using (3.10) and (3.12) we can write

$$\begin{aligned}
\frac{(g_{k+1}^T d_{k+1})^2}{\|d_{k+1}\|^2}+\frac{(g_k^T s_k)^2}{\|s_k\|^2}&=\frac{1}{\|d_{k+1}\|^2}\left[(g_{k+1}^T d_{k+1})^2+\frac{\|d_{k+1}\|^2}{\|s_k\|^2}(g_k^T s_k)^2\right]\\
&=\frac{1}{\|d_{k+1}\|^2}\left[(g_{k+1}^T d_{k+1})^2+\frac{(g_k^T s_k)^2}{\|s_k\|^2}\left(-(\theta_k^2-2\theta_k w)\|g_{k+1}\|^2+\beta_k^2\|s_k\|^2\right)\right]\\
&\ge\frac{1}{\|d_{k+1}\|^2}\left[e\|g_{k+1}\|^4-(\theta_k^2-2\theta_k w)\frac{(g_k^T s_k)^2}{\|s_k\|^2}\|g_{k+1}\|^2\right]\\
&=\frac{\|g_{k+1}\|^4}{\|d_{k+1}\|^2}\left[e-(\theta_k^2-2\theta_k w)\frac{(g_k^T s_k)^2}{\|s_k\|^2}\frac{1}{\|g_{k+1}\|^2}\right].
\end{aligned} \tag{3.13}$$

From Lemma 3.2 we know that

$$\lim_{k\to\infty}\frac{(g_k^T s_k)^2}{\|s_k\|^2}=0.$$

On the other hand, for $\theta_k\in[0,2w]$, $\theta_k^2-2\theta_k w$ is finite. Therefore, if (3.8) is not true, then

$$\lim_{k\to\infty}\frac{(g_k^T s_k)^2}{\|s_k\|^2}\frac{(\theta_k^2-2\theta_k w)}{\|g_{k+1}\|^2}=0.$$

Hence,

$$\frac{(g_{k+1}^T d_{k+1})^2}{\|d_{k+1}\|^2}+\frac{(g_k^T s_k)^2}{\|s_k\|^2}\ge e\frac{\|g_{k+1}\|^4}{\|d_{k+1}\|^2}, \tag{3.14}$$

holds for all sufficiently large k. Therefore, by Lemma 3.2 it follows that (3.9) is true. ■

Using Lemma 3.3 we can prove the following proposition which has a crucial role in proving the convergence of our algorithm.

Proposition 3.1. *Consider the conjugate gradient algorithm (2.2) where* θ_k *and* β_k *are given by (2.11) and (2.3) respectively and* α_k *is obtained by the strong Wolfe line search (3.1) and (3.2). Suppose that the assumptions (i) and (ii) hold, and* $\theta_k \in [0, 2w]$. *If*

$$\sum_{k\geq 1} \frac{1}{\|d_k\|^2} = \infty, \tag{3.15}$$

then

$$\liminf_{k\to\infty} \|g_k\| = 0. \tag{3.16}$$

Proof. Suppose by contradiction that there is a positive constant γ such that $\|g_k\| \geq \gamma$ for all $k \geq 1$. Therefore, from Lemma 3.3 it follows that

$$\sum_{k\geq 1} \frac{1}{\|d_k\|^2} \leq \frac{1}{\gamma^4} \sum_{k\geq 1} \frac{\|g_k\|^4}{\|d_k\|^2} < \infty$$

which is in contradiction with (3.15). ■

Therefore, the iteration can fail, in the sense that $\|g_k\| \geq \gamma > 0$ for all k, only if $\|d_k\| \to \infty$ sufficiently rapidly.

Convergence for uniformly convex functions. For uniformly convex functions we can prove that the norm of the direction d_k generated by (2.2), where θ_k and β_k are given by (2.11) and (2.3) respectively, is bounded. Thus by Proposition 3.1 we can prove the following result.

Theorem 3.1. *Suppose that the assumptions (i) and (ii) hold. Consider the method (2.1)-(2.3) and (2.11), where* d_k *is a descent direction and* α_k *is obtained by the strong Wolfe line search. Suppose that there exists the positive constants* c_1 *and* t *such that* $\theta_k < c_1$ *and* $|t_k| < t$ *for all* $k \geq 1$. *If there exists a constant* $\mu > 0$ *such that*

$$(\nabla f(x) - \nabla f(y))^T (x - y) \geq \mu \|x - y\|^2 \tag{3.17}$$

for all $x, y \in S$, *then*

$$\lim_{k\to\infty} g_k = 0. \tag{3.18}$$

Proof. From (3.17) it follows that f is a uniformly convex function in S and therefore

$$y_k^T s_k \geq \mu \|s_k\|^2. \tag{3.19}$$

Again, by Lipschitz continuity $\|y_k\| \leq L\|s_k\|$. Now, from (2.3) we have that

$$\begin{aligned} |\beta_k| &= \left| \frac{y_k^T g_{k+1}}{y_k^T s_k} - t_k \frac{s_k^T g_{k+1}}{y_k^T s_k} \right| \leq \frac{\|y_k\| \|g_{k+1}\|}{\mu \|s_k\|^2} + |t_k| \frac{\|s_k\| \|g_{k+1}\|}{\mu \|s_k\|^2} \\ &\leq \frac{L\|s_k\| \|g_{k+1}\|}{\mu \|s_k\|^2} + t \frac{\|s_k\| \|g_{k+1}\|}{\mu \|s_k\|^2} = \frac{L+t}{\mu} \frac{\Gamma}{\|s_k\|}. \end{aligned} \tag{3.20}$$

Hence,

$$\|d_{k+1}\| \le c_1\Gamma + \frac{L+t}{\mu}\frac{\Gamma}{\|s_k\|}\|s_k\| = \left(c_1 + \frac{L+t}{\mu}\right)\Gamma. \tag{3.21}$$

Which implies that (3.15) is true. Therefore, by Proposition 3.1 we have (3.16), which for uniformly convex functions is equivalent to (3.18). ■

Observe that in Lemma 3.3 and in Proposition 3.1 θ_k is bounded as $0 \le \theta_k \le 2w$. We know that θ_k tends to w. Therefore, in Theorem 3.1 the positive constant c_1 is bounded as $c_1 \le 2w$.

Convergence for general nonlinear functions. Firstly we prove that in very mild conditions the direction d_k generated by (2.2), where θ_k and β_k are given by (2.11) and (2.3) respectively, is bounded. Again, by Proposition 3.1 we can prove the following result.

Theorem 3.2. *Suppose that the assumptions (i) and (ii) hold. Consider the conjugate gradient algorithm (2.1), where the direction d_{k+1} is given by (2.2) and (2.3), and the step length α_k is obtained by the strong Wolfe line search (3.1) and (3.2). Assume that for all $k \ge 0$ there exist positive constants $c > 0$ and $c_1 > 0$ such that $|y_k^T g_{k+1}| \le c/\|s_k\|$ and $\theta_k < c_1$ respectively, then* $\liminf_{k\to\infty} \|g_k\| = 0$.

Proof. From (2.3) using (2.10) after some algebra we get

$$\beta_k = \frac{y_k^T g_{k+1}}{y_k^T s_k}\left(1 - \frac{b_k}{\bar{\Delta}_k}\right) + a_k \frac{\|g_{k+1}\|^2}{\bar{\Delta}_k}. \tag{3.22}$$

Suppose that $g_k \ne 0,$ otherwise a stationary point is obtained. By the Wolfe line search $y_k^T s_k > 0$. Since d_k is a descent direction for all $k \ge 0,$ it follows that $\|s_k\|$ tends to zero. Hence, there exists a positive constant $c_3 > 0$ such that

$$\left|\frac{y_k^T g_{k+1}}{y_k^T s_k}\right| \le \frac{c_3}{\|s_k\|}. \tag{3.23}$$

Now, observe that since for all $k \ge 0$, $b_k > 0$ and $\bar{\Delta}_k < 0,$ it follows that $-b_k/\bar{\Delta}_k > 0$. Besides, from (2.6) and (2.9) we can write

$$-\frac{b_k}{\bar{\Delta}_k} = w + (1+w)\frac{(y_k^T g_{k+1})(s_k^T g_{k+1})}{-\bar{\Delta}_k}. \tag{3.24}$$

Since $-\bar{\Delta}_k > 0$ and $s_k^T g_{k+1}$ tends to zero along the iterations, it follows that $-b_k/\bar{\Delta}_k$ tends to $w > 0$. Therefore, there exists a positive constant $c_4 > 0$ such that $1 < 1 - b_k/\bar{\Delta}_k \le c_4$.

Again observe that if $g_k \ne 0$ from the Wolfe line search $y_k^T s_k > 0$. Hence, there exists a positive constant $c_5 > 0$ such that $0 < y_k^T s_k \le c_5/\|s_k\|$ for all $k \ge 0$.

Now, from (2.8) and (2.16) we have

$$\begin{aligned} |a_k| &= \left|v(s_k^T g_{k+1}) + (y_k^T g_{k+1})\right| \le v\left|s_k^T g_{k+1}\right| + \left|y_k^T g_{k+1}\right| \\ &\le v\left|y_k^T s_k\right| \max\left\{1, \frac{\sigma}{1-\sigma}\right\} + \left|y_k^T g_{k+1}\right| \le v\frac{c_5}{\|s_k\|}\max\left\{1, \frac{\sigma}{1-\sigma}\right\} + \frac{c}{\|s_k\|} \end{aligned}$$

$$= \left(vc_5 \max\left\{1, \frac{\sigma}{1-\sigma}\right\} + c \right) \frac{1}{\|s_k\|}. \tag{3.25}$$

Hence, since $|\bar{\Delta}_k| > c_2$ from (3.25) it follows that

$$|a_k| \frac{\|g_{k+1}\|^2}{|\bar{\Delta}_k|} \le \left(vc_5 \max\left\{1, \frac{\sigma}{1-\sigma}\right\} + c \right) \frac{\Gamma^2}{c_2} \frac{1}{\|s_k\|}. \tag{3.26}$$

With these, from (3.22) we can write

$$\begin{aligned} |\beta_k| &\le \left| \frac{y_k^T g_{k+1}}{y_k^T s_k} \right| \left| 1 - \frac{b_k}{\bar{\Delta}_k} \right| + |a_k| \frac{\|g_{k+1}\|^2}{|\bar{\Delta}_k|} \\ &\le \frac{c_3}{\|s_k\|} c_4 + \left(vc_5 \max\left\{1, \frac{\sigma}{1-\sigma}\right\} + c \right) \frac{\Gamma^2}{c_2} \frac{1}{\|s_k\|} \\ &= \left[c_3 c_4 + \left(vc_5 \max\left\{1, \frac{\sigma}{1-\sigma}\right\} + c \right) \frac{\Gamma^2}{c_2} \right] \frac{1}{\|s_k\|}. \end{aligned} \tag{3.27}$$

From (2.2) we have

$$\begin{aligned} \|d_{k+1}\| &\le |\theta_k| \|g_{k+1}\| + |\beta_k| \|s_k\| \\ &\le c_1 \Gamma + \left[c_3 c_4 + \left(vc_5 \max\left\{1, \frac{\sigma}{1-\sigma}\right\} + c \right) \frac{\Gamma^2}{c_2} \right] \frac{1}{\|s_k\|} \|s_k\| \equiv E, \end{aligned} \tag{3.28}$$

where E is a positive constant. Therefore, for all $k \ge 0$, $\|d_k\| \le E$, which implies (3.15). Therefore, by Proposition 3.1, since d_k is a descent direction, we have $\liminf_{k\to\infty} \|g_k\| = 0$. ■

Observe that if for every $k \ge 1$, $\theta_k = 1$ and $t_k = 0$, then (2.2) reduces to the Hestenes and Stiefel direction. For an exact line search the HS algorithm reduces to that of Polak-Ribière and Polyak (PRP). Therefore, the convergence properties of the HS method should be similar to the convergence properties of the PRP method. In particular, for a general nonlinear function by the Powell's example, the HS method with an exact line search may not converge. Hence, our method (2.1)-(2-3) need not converge for general functions. Therefore, like in Gilbert and Nocedal [22], who proved the global convergence of the PRP method with the restriction that $\beta_k^{PRP} \ge 0$, we replace (2.3) by

$$\beta_k = \max\left\{ \frac{y_k^T g_{k+1}}{y_k^T s_k}, 0 \right\} - t_k \frac{s_k^T g_{k+1}}{y_k^T s_k} \tag{3.29}$$

and prove the global convergence of this modification of the algorithm for general functions. Firstly, we prove the following results.

Lemma 3.4. *Suppose that the assumptions (i) and (ii) hold. Consider the conjugate gradient algorithm (2.2), where θ_k and β_k are given by (2.11) and (3.29) respectively and α_k is obtained by the strong Wolfe line search. Suppose that there exists the positive constants c_1 and t such that $\theta_k < c_1$ and $|t_k| < t$ for all $k \ge 1$. If there exists a positive constant $\gamma > 0$ such that*

$$\|g_k\| \ge \gamma \tag{3.30}$$

for all $k \ge 0$, then $d_k \ne 0$ and

$$\sum_{k\ge 1} \|u_{k+1} - u_k\|^2 < \infty, \tag{3.31}$$

where $u_k = d_k / \|d_k\|$.

Proof. First, we note that $d_k \neq 0$, otherwise the descent condition (2.4) is not true. Therefore, u_k is well defined. Besides, by (3.30) and the Proposition 3.1 we have

$$\sum_{k\geq 0} \frac{1}{\|d_k\|} < \infty, \tag{3.32}$$

otherwise (3.16) is true, contradicting (3.30)

Now, as usual (see [15]) we can consider $\beta_k = \beta_k^1 + \beta_k^2$, where

$$\beta_k^1 = \max\left\{\frac{y_k^T g_{k+1}}{y_k^T s_k}, 0\right\} \tag{3.33}$$

$$\beta_k^2 = -t_k \frac{s_k^T g_{k+1}}{y_k^T s_k}. \tag{3.34}$$

Define

$$v_{k+1} = -\theta_k g_{k+1} + \beta_k^2 s_k, \tag{3.35}$$

$$r_{k+1} = \frac{v_{k+1}}{\|d_{k+1}\|}, \tag{3.36}$$

$$\delta_k = \beta_k^1 \frac{\|d_k\|}{\|d_{k+1}\|} \geq 0. \tag{3.37}$$

With these we have

$$u_{k+1} = r_{k+1} + \alpha_k \delta_k u_k. \tag{3.38}$$

But, $\|u_k\| = \|u_{k+1}\| = 1$ and therefore from (3.38) we obtain

$$\|r_{k+1}\| = \|u_{k+1} - \alpha_k \delta_k u_k\| = \|\alpha_k \delta_k u_{k+1} - u_k\|. \tag{3.39}$$

Now, using the condition $\delta_k \geq 0$, the triangle inequality and (3.39) we have

$$\begin{aligned} \|u_{k+1} - u_k\| &= \|(1+\alpha_k\delta_k)u_{k+1} - (1+\alpha_k\delta_k)u_k\| \\ &\leq \|u_{k+1} - \alpha_k\delta_k u_k\| + \|\alpha_k\delta_k u_{k+1} - u_k\| = 2\|r_{k+1}\|. \end{aligned} \tag{3.40}$$

On the other hand, from the strong Wolfe line search and the descent condition it follows that

$$\left|\frac{s_k^T g_{k+1}}{y_k^T s_k}\right| \leq \max\left\{1, \frac{\sigma}{1-\sigma}\right\}. \tag{3.41}$$

Hence, from the definition of v_{k+1} given by (3.35), (3.41) and the assumptions (i) and (ii), i.e. $\|x_k\| \leq B$ and $\|g_k\| \leq \Gamma$ for all $k \geq 0$, we obtain

$$\begin{aligned} \|v_{k+1}\| &\leq \theta_k \|g_{k+1}\| + |t_k| \left|\frac{s_k^T g_{k+1}}{y_k^T s_k}\right| \|s_k\| \\ &\leq c_1\Gamma + t \max\left\{1, \frac{\sigma}{1-\sigma}\right\} 2B. \end{aligned} \tag{3.42}$$

Therefore,

$$\|u_{k+1} - u_k\| \leq 2\|r_{k+1}\| = 2\frac{\|v_{k+1}\|}{\|d_{k+1}\|} \leq \frac{2}{\|d_{k+1}\|}\left(c_1\Gamma + t\max\left\{1, \frac{\sigma}{1-\sigma}\right\}2B\right),$$

which completes the proof. ■

This lemma shows that asymptotically the search directions generated by the algorithm (2.2), where θ_k and β_k are computed as in (2.11) and (3.29) respectively, change slowly.

Lemma 3.5. *Suppose that the assumptions (i) and (ii) hold. Consider the conjugate gradient algorithm (2.2), where θ_k and β_k are given by (2.11) and (3.29) respectively and α_k is obtained by the strong Wolfe line search and for all $k \geq 0$, $\alpha_k \geq \omega$. Suppose that there exist the positive constants t and γ such that for all $k \geq 1$, $|t_k| < t$ and $\|g_k\| > \gamma$, respectively. Then there exist the constants $b > 1$ and $\lambda > 0$ such that for all $k \geq 1$*

$$|\beta_k| \leq b \tag{3.43}$$

and

$$\|s_k\| \leq \lambda \text{ implies } |\beta_k| \leq \frac{1}{b}. \tag{3.44}$$

Proof. We have

$$y_k^T s_k \geq (\sigma - 1) s_k^T g_k = (\sigma - 1)\alpha_k d_k^T g_k = -(\sigma - 1)\alpha_k w \|g_k\|^2 \geq (1 - \sigma)\omega w \gamma^2.$$

Therefore

$$|\beta_k| \leq \left| \frac{y_k^T g_{k+1}}{y_k^T s_k} \right| + |t_k| \left| \frac{s_k^T g_{k+1}}{y_k^T s_k} \right| \leq \frac{\|y_k\| \|g_{k+1}\|}{(1-\sigma)\omega w \gamma^2} + t \frac{\|s_k\| \|g_{k+1}\|}{(1-\sigma)\omega w \gamma^2}$$

$$\leq \frac{L\|s_k\|\Gamma + t\|s_k\|\Gamma}{(1-\sigma)\omega w \gamma^2} \leq \frac{2(L+t)B\Gamma}{(1-\sigma)\omega w \gamma^2} \equiv b. \tag{3.45}$$

Without loss of generality we can define b such that $b > 1$. Let us define

$$\lambda \equiv \frac{(1-\sigma)\omega w \gamma^2}{2(L+t)\Gamma b}. \tag{3.46}$$

Obviously, if $\|s_k\| \leq \lambda$, then from the third inequality in (3.45) we have

$$|\beta_k| \leq \frac{(L+t)\Gamma\lambda}{(1-\sigma)\omega w \gamma^2} = \frac{1}{b}. \tag{3.47}$$

Therefore, for b and λ defined in (3.45) and (3.46) respectively, it follows that the relations (3.43) and (3.44) hold. ■

The property presented in Lemma 3.5, which is similar to but slightly different from Property (*) in [22], can be used to show that if the gradients are bounded away from zero and (3.43) and (3.44) hold, then a finite number of steps s_k cannot be too small. Therefore, the algorithm makes a rapid progress to the optimum. Indeed, for $\lambda > 0$ and a positive integer J let us define the set of index

$$K_{k,J}^{\lambda} = \left\{ i \in N^* : k \leq i \leq k + J - 1, \|s_k\| > \lambda \right\}, \tag{3.48}$$

where N^* is the set of positive integers. The following Lemma is similar to Lemma 3.5 in [15] and Lemma 4.2 in [22].

Lemma 3.6. *Suppose that all assumptions of Lemma 3.5 are satisfied. Then there exists a $\lambda > 0$ such that for any $J \in N^*$ and any index k_0, there is a greater index $k \geq k_0$ such that $\left|K_{k,J}^{\lambda}\right| > J/2$.*

Using Lemma 3.4 and Lemma 3.6 we can prove the global convergence of the conjugate gradient algorithm (2.2) where θ_k and β_k are given by (2.11) and (3.29) respectively and α_k is obtained by the strong Wolfe line search. The following Theorem is similar to Theorem 3.6

in Dai and Liao [15] or to Theorem 3.2 in Hager and Zhang [25] and the proof is omitted here.

Theorem 3.3. *Suppose that the assumptions (i) and (ii) hold. Consider the conjugate gradient algorithm (2.2), where* θ_k *and* β_k *are given by (2.11) and (3.29) respectively and* α_k *is obtained by the strong Wolfe line search. Then we have* $\liminf_{k\to\infty} \|g_k\| = 0.$

4. DLDC algorithm

We know that in conjugate gradient algorithms the search directions tend to be poorly scaled and as a consequence the line search must perform more function evaluations in order to obtain a suitable steplength α_k. Therefore, the research effort was directed to design procedures for direction computation which takes the second order information. For example, the algorithms implemented in SCALCG by Andrei [3-5], or CONMIN by Shanno and Phua [40] use the BFGS preconditioning with remarkable results.

In conjugate gradient methods the step lengths computed by means of the Wolfe line search (1.4) and (1.5) may differ from 1 in a very unpredictable manner [32]. They can be larger or smaller than 1 depending on how the problem is scaled. This is in very sharp contrast to the Newton and quasi-Newton methods, including the limited memory quasi-Newton methods, which accept the unit steplength most of the time along the iterations, and therefore usually they require only few function evaluations per search direction. Numerical comparisons between conjugate gradient method and limited memory quasi Newton method by Liu and Nocedal [29] showed that the latter is more successful [6]. One partial explanation of the efficiency of this limited memory quasi-Newton method is given by its ability to accept unity step lengths along the iterations. In this section we take advantage of this behavior of conjugate gradient algorithms and consider an acceleration scheme we have presented in [7] (see also [2]). Basically the acceleration scheme modifies the step length α_k in a multiplicative manner to improve the reduction of the function values along the iterations. In accelerated algorithm instead of (1.2) the new estimation of the minimum point is computed as

$$x_{k+1} = x_k + \xi_k \alpha_k d_k, \tag{4.1}$$

where

$$\xi_k = -\frac{\bar{a}_k}{\bar{b}_k}, \tag{4.2}$$

$\bar{a}_k = \alpha_k g_k^T d_k$, $\bar{b}_k = -\alpha_k (g_k - g_z)^T d_k$, $g_z = \nabla f(z)$ and $z = x_k + \alpha_k d_k$. Hence, if $\bar{b}_k \neq 0$, then the new estimation of the solution is computed as $x_{k+1} = x_k + \xi_k \alpha_k d_k$, otherwise $x_{k+1} = x_k + \alpha_k d_k$. Observe that since ρ in (1.4) is enough small (usually $\rho = 0.0001$), the Wolfe line search leads to very small reductions in function's values along the iterations. The acceleration scheme (4.1) emphasizes the reduction of function's values, since in conjugate gradient algorithms often $\alpha_k > 1$ along the iterations (see [7]). Therefore, using the definitions of g_k, s_k, y_k and the above acceleration scheme (4.1) and (4.2) we can present the following conjugate gradient algorithm.

DLDC algorithm

Step 1. Select a starting point $x_0 \in dom\, f$ and compute: $f_0 = f(x_0)$ and $g_0 = \nabla f(x_0)$. Select some positive values for ρ and σ, and for ν and w. Set $d_0 = -g_0$ and $k = 0$.

Step 2. Test a criterion for stopping the iterations. If the test is satisfied, then stop; otherwise continue with step 3.

Step 3. Determine the steplength α_k by using the Wolfe line search conditions (1.4) - (1.5)

Step 4. Acceleration scheme. Compute: $z = x_k + \alpha_k d_k$, $g_z = \nabla f(z)$ and $y_k = g_k - g_z$.

Step 5. Compute: $\bar{a}_k = \alpha_k g_k^T d_k$, and $\bar{b}_k = -\alpha_k y_k^T d_k$.

Step 6. If $\bar{b}_k \neq 0$, then compute $\xi_k = -\bar{a}_k / \bar{b}_k$ and update the variables as $x_{k+1} = x_k + \xi_k \alpha_k d_k$, otherwise update the variables as $x_{k+1} = x_k + \alpha_k d_k$. Compute f_{k+1} and g_{k+1}. Compute $y_k = g_{k+1} - g_k$ and $s_k = x_{k+1} - x_k$.

Step 7. Compute Δ_k as in (2.7).

Step 8. If $|\Delta_k| \geq \varepsilon_m$, then determine θ_k and β_k as in (2.11) and (3.29) respectively. Else, set $\theta_k = 1$ and $\beta_k = y_k^T g_{k+1} / y_k^Y s_k$.

Step 9. Compute the search direction as: $d_{k+1} = -\theta_k g_{k+1} + \beta_k s_k$.

Step 10. Compute $\sigma = \|g_{k+1}\|^2 / (|y_k^T g_{k+1}| + \|g_{k+1}\|^2)$. If $\sigma < \rho$, then set $\sigma = 0.8$.

Step 11. Restart criterion. If $|g_{k+1}^T g_k| > 0.2\|g_{k+1}\|^2$ then set $d_{k+1} = -g_{k+1}$.

Step 12. Consider $k = k+1$ and go to step 2. ■

It is well known that if f is bounded along the direction d_k then there exists a stepsize α_k satisfying the Wolfe line search conditions (1.4) and (1.5). In our algorithm when the Powell restart condition is satisfied, then we restart the algorithm with the negative gradient $-g_{k+1}$. More sophisticated reasons for restarting the algorithms have been proposed in the literature [16], but we are interested in the performance of a conjugate gradient algorithm that uses this restart criterion associated to a direction satisfying both the descent and the conjugacy conditions. Under reasonable assumptions, the Wolfe conditions and the Powell restart criterion are sufficient to prove the global convergence of the algorithm. The first trial of the step length crucially affects the practical behavior of the algorithm. At every iteration $k \geq 1$ the starting guess for the step α_k in the line search is computed as $\alpha_{k-1}\|d_{k-1}\| / \|d_k\|$. This selection was used for the first time by Shanno and Phua in CONMIN [40] and in SCALCG by Andrei [3-5]. Observe that in the line search procedure (step 3) the steplength α_k is computed using the updated value of the parameter σ, computed as in step 10. For uniformly convex functions, we can prove the linear convergence of the acceleration scheme [7].

5. Numerical results and comparisons

In this section we report some numerical results obtained with an implementation of the DLDC algorithm. The code is written in Fortran and compiled with f77 (default compiler settings) on a Workstation Intel Pentium 4 with 1.8 GHz. DLDC uses the loop unrolling to a depth of 5. We selected a number of 75 large-scale unconstrained optimization test functions in generalized or extended form [1] (some from CUTE library [12]). For each test function we have taken ten numerical experiments with the number of variables $n = 1000, 2000, ..., 10000$. The algorithm implements the Wolfe line search conditions with $\rho = 0.0001$, $\sigma = \|g_{k+1}\|^2 / (|y_k^T g_{k+1}| + \|g_{k+1}\|^2)$, and the same stopping criterion $\|g_k\|_\infty \leq 10^{-6}$, where $\|.\|_\infty$ is the maximum absolute component of a vector. If $\sigma < \rho$, then we set $\sigma = 0.8$. If $|\Delta_k| \geq \varepsilon_m$, where ε_m is epsilon machine, then θ_k and β_k are computed as in (2.11) and (3.29), respectively. Otherwise, set $\theta_k = 1$ and $\beta_k = y_k^T g_{k+1} / y_k^T s_k$, i.e. the Hestenes-Stiefel conjugate gradient algorithm [27] is considered. In DLDC we set $w = 7/8$ and $v = 0.05$. In our numerical experiments θ_k is not restricted in the interval $[0, 2w]$. In all

the algorithms we considered in this numerical study the maximum number of iterations is limited to 10000.

The comparisons of algorithms are given in the following context. Let f_i^{ALG1} and f_i^{ALG2} be the optimal value found by ALG1 and ALG2, for problem $i = 1,\ldots,750,$ respectively. We say that, in the particular problem $i,$ the performance of ALG1 was better than the performance of ALG2 if:

$$\left| f_i^{ALG1} - f_i^{ALG2} \right| < 10^{-3} \tag{5.1}$$

and the number of iterations (#iter), or the number of function-gradient evaluations (#fg), or the CPU time of ALG1 was less than the number of iterations, or the number of function-gradient evaluations, or the CPU time corresponding to ALG2, respectively.

In the first set of numerical experiments we compare DLDC versus Dai and Liao $(v = 1)$ conjugate gradient algorithm (1.13). Figure 1 shows the Dolan and Moré CPU performance profile of DLDC versus DL $(v = 1)$. In a performance profile plot, the top curve corresponds to the method that solved the most problems in a time that was within a factor τ of the best time. The percentage of the test problems for which a method is the fastest is given on the left axis of the plot. The right side of the plot gives the percentage of the test problems that were successfully solved by these algorithms, respectively. Mainly, the right side is a measure of the robustness of an algorithm.

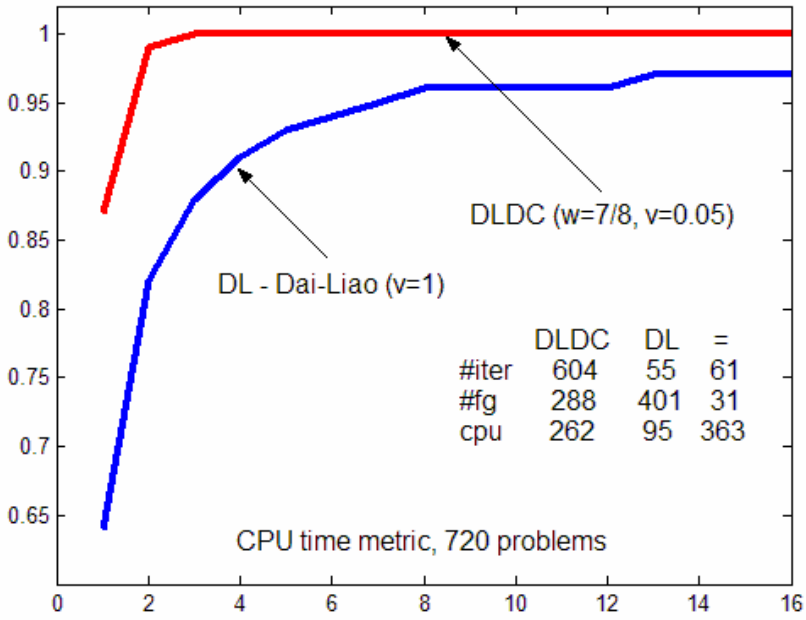

Fig. 1. DLDC ($w = 7/8, v = 0.05$) versus DL $(v = 1)$.

When comparing DLDC with DL $(v = 1)$ conjugate gradient algorithm subject to CPU time metric we see that DLDC is top performer, i.e. the accelerated Dai and Liao conjugate gradient algorithm with guaranteed descent and conjugacy conditions is more successful and more robust than the Dai and Liao conjugate gradient algorithms with $v = 1.$ Comparing DLDC with DL $(v = 1)$ (see Figure 1), subject to the number of iterations, we see that DLDC was better in 604 problems (i.e. it achieved the minimum number of iterations in 604 problems). DL $(v = 1)$ was better in 55 problems and they achieved the same number of iterations in 61 problems, etc. Out of 750 problems, only for 720 problems does the criterion (5.1) hold. Therefore, DLDC appears to generate the best search direction and the best steplength, on average.

In the second set of numerical experiments we compare DLDC versus Hestenes and Stiefel (HS) ($\beta_k^{HS} = \frac{y_k^T g_{k+1}}{y_k^T s_k}$) [27], versus Dai and Yuan (DY) ($\beta_k^{DY} = \frac{g_{k+1}^T g_{k+1}}{y_k^T s_k}$) [17] and versus Polak-Ribière-Polyak (PRP) ($\beta_k^{PRP} = \frac{y_k^T g_{k+1}}{g_k^T g_k}$) [35, 36], conjugate gradient algorithms. Figures 2-4 present the Dolan and Moré CPU performance profile of DLDC versus HS, DY and PRP, respectively.

An attractive feature of the Hestenes and Stiefel conjugate gradient algorithm is that the pure conjugacy condition $y_k^T d_{k+1} = 0$ always is satisfied, independent of the line search. However, for an exact line search the convergence properties of the HS method are similar to the convergence properties of the PRP method. Therefore, by Powell's example [37], the HS method with exact line search may not converge for a general nonlinear function. Both the HS and PRP methods possess a built-in restart feature that addresses directly to the jamming phenomenon. When the step $x_{k+1} - x_k$ is small, the factor $y_k = g_{k+1} - g_k$ in the numerator of β_k tends to zero. Therefore, β_k becomes small and the new search direction d_{k+1} essentially becomes the steepest descent direction $-g_{k+1}$. Hence, both HS and PRP methods automatically adjust β_k to avoid jamming. The performance of these methods is better than the performance of DY. On the other hand, the DY method always generates descent directions, and in [14] Dai established a remarkable property for the DY conjugate gradient algorithm, relating the descent directions to the sufficient descent condition. It is shown that if there exist constants γ_1 and γ_2 such that $\gamma_1 \le \|g_k\| \le \gamma_2$ for all k, then for any $p \in (0,1)$, there exists a constant $c > 0$ such that the sufficient descent condition $g_i^T d_i \le -c\|g_i\|^2$ holds for at least $\lfloor pk \rfloor$ indices $i \in [0,k]$, where $\lfloor j \rfloor$ denotes the largest integer $\le j$. However, the DY method does not satisfy the conjugacy condition. In contrast, observe that in DLDC the search directions are always descent directions and the conjugacy condition always is satisfied independent of the accuracy of the line search.

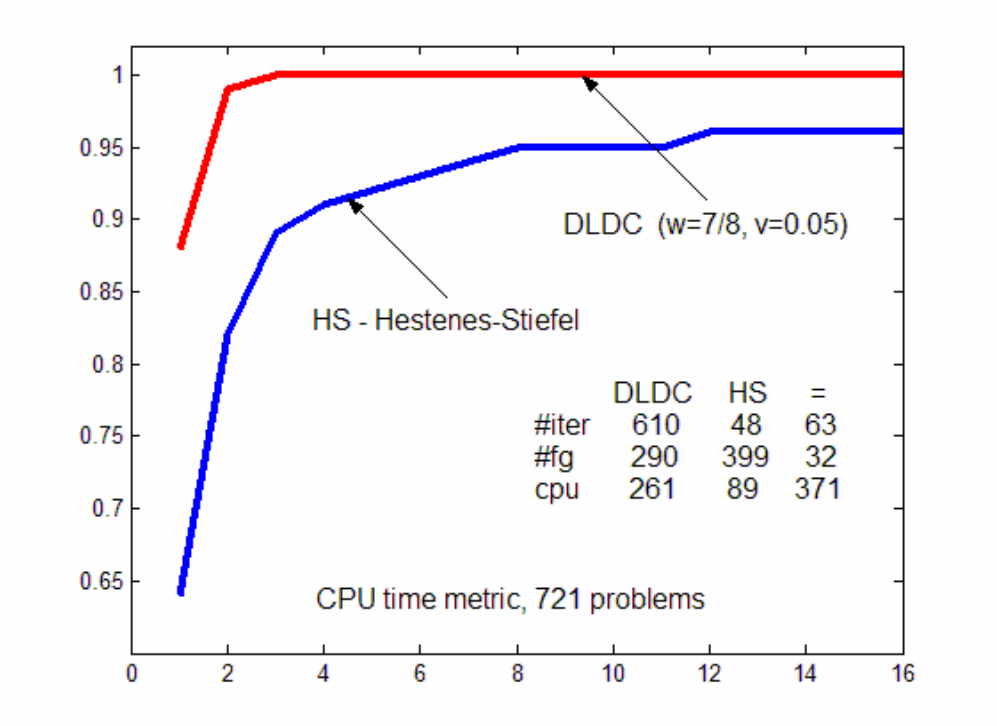

Fig. 2. DLDC ($w = 7/8$, $v = 0.05$) versus Hestenes-Stiefel.

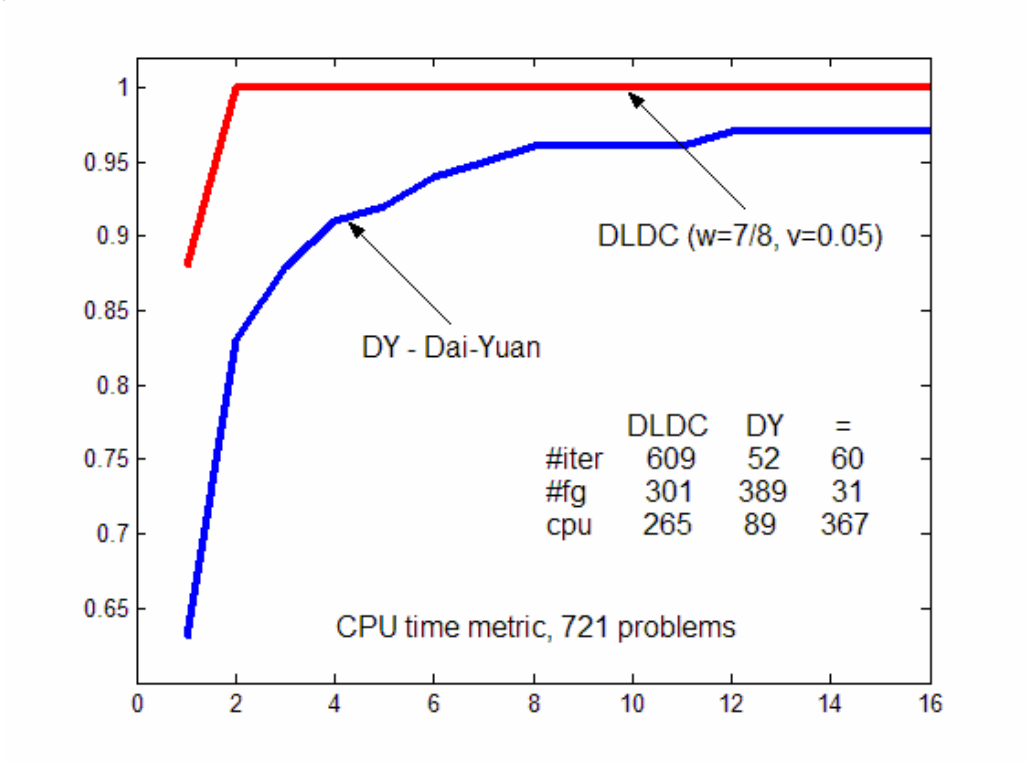

Fig. 3. DLDC ($w = 7/8, v = 0.05$) versus Dai-Yuan.

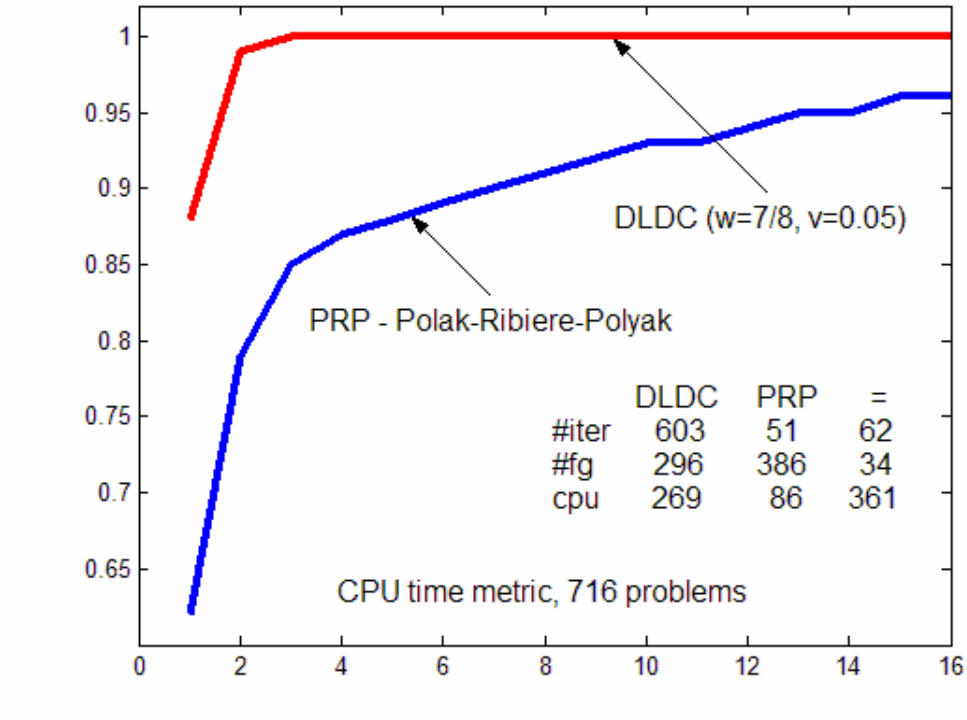

Fig. 4. DLDC ($w = 7/8, v = 0.05$) versus Polak-Ribière-Polyak.

In the third set of numerical experiments we compare DLDC versus hybrid Dai-Yuan ($\beta_k^{hDY} = \max\left\{-c\beta_k^{DY}, \min\left\{\beta_k^{HS}, \beta_k^{DY}\right\}\right\}, c = (1-\sigma)/(1+\sigma), \quad \sigma = 0.8$) [17]. The hDY method reduces to the Fletcher and Reeves method [21] if f is a strictly convex quadratic

function and the line search is exact. For a standard Wolfe line search, Dai and Yuan [17] proved that it produces descent directions at every iteration and they established the global convergence of their hybrid conjugate gradient algorithm when the Lipschitz assumption holds. However, the hDY conjugate gradient algorithm does not satisfy the conjugacy condition. Figure 5 presents the Dolan and Moré CPU time performance profile of DLDC versus hDY. The best performance, relative to the CPU time metric, again was obtained by DLDC, the top curve in Figure 5.

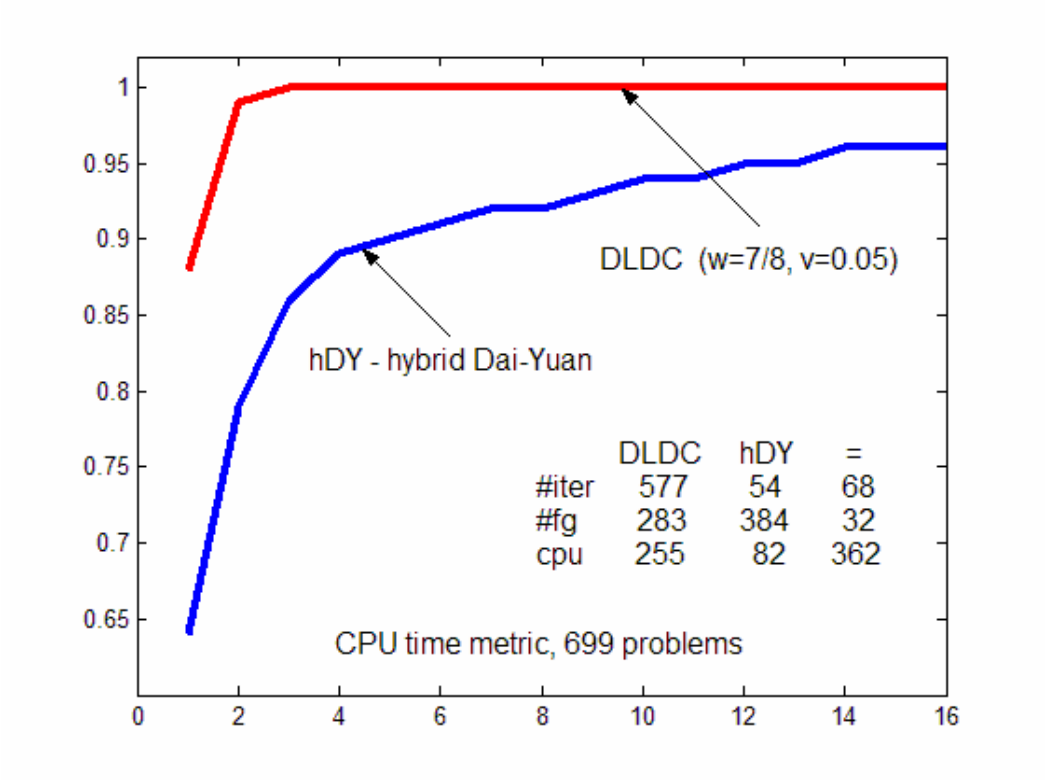

Fig. 5. DLDC ($w = 7/8$, $v = 0.05$) versus hybrid Dai-Yuan.

In the fourth set of numerical experiments we compare DLDC versus CG_DESCENT by Hager and Zhang [25]. Presently CG_DESCENT is the practical conjugate gradient algorithm with more reputation. CG_DESCENT is a modification of HS and was devised in order to ensure sufficient descent, independent of the accuracy of the line search. Hager and Zhang [25] proved that the direction d_k in their algorithm satisfies the sufficient descent condition $g_k^T d_k \le -(7/8)\|g_k\|^2$. This is the main reason we considered $w = 7/8$ in all our numerical experiments. CG_DESCENT has a very advanced line search procedure that utilizes the "approximate Wolfe conditions" which provides a more accurate way to check the usual Wolfe conditions when the iterates are near a local minimum of the function f. However, in CG_DESCENT the conjugacy condition (1.11) holds approximately. CG_DESCENT like DLDC uses the loop unrolling to a depth of 5. Figure 6 presents the Dolan and Moré CPU time performance profile of DLDC versus CG_DESCENT with Wolfe line search. Again, the best performance, relative to the CPU time metric, was obtained by DLDC, the top curve in Figure 6.

Finally we compare DLDC versus L-BFGS (m=3) by Liu and Nocedal [29] as in Figure 7, where m is the number of pairs (s_k, y_k) used. Observe that DLDC is top performer again. The differences are significant. The linear algebra in the L-BFGS code to update the search direction is very different from the linear algebra used in AMDYN. On the other hand the steplength in L-BFGS is determined at each iteration by means of the line search routine MCVSRCH, which is a slight modification of the routine CSRCH written by Moré and Thuente [30].

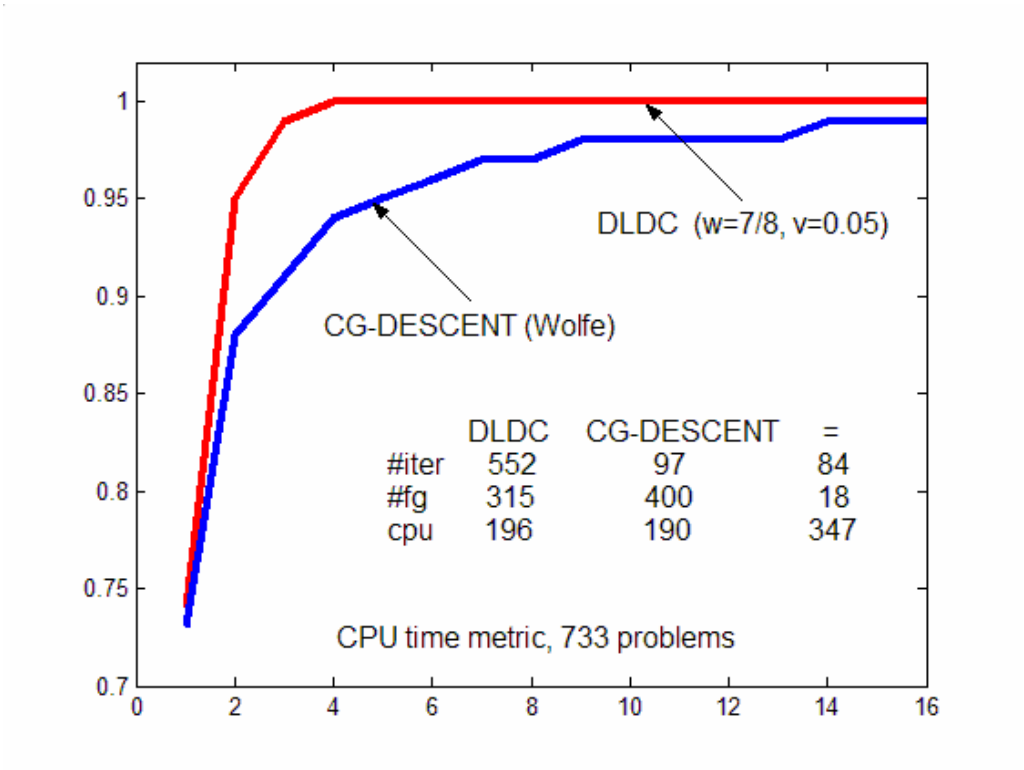

Fig. 6. DLDC ($w = 7/8, v = 0.05$) versus CG_DESCENT by Hager and Zhang.

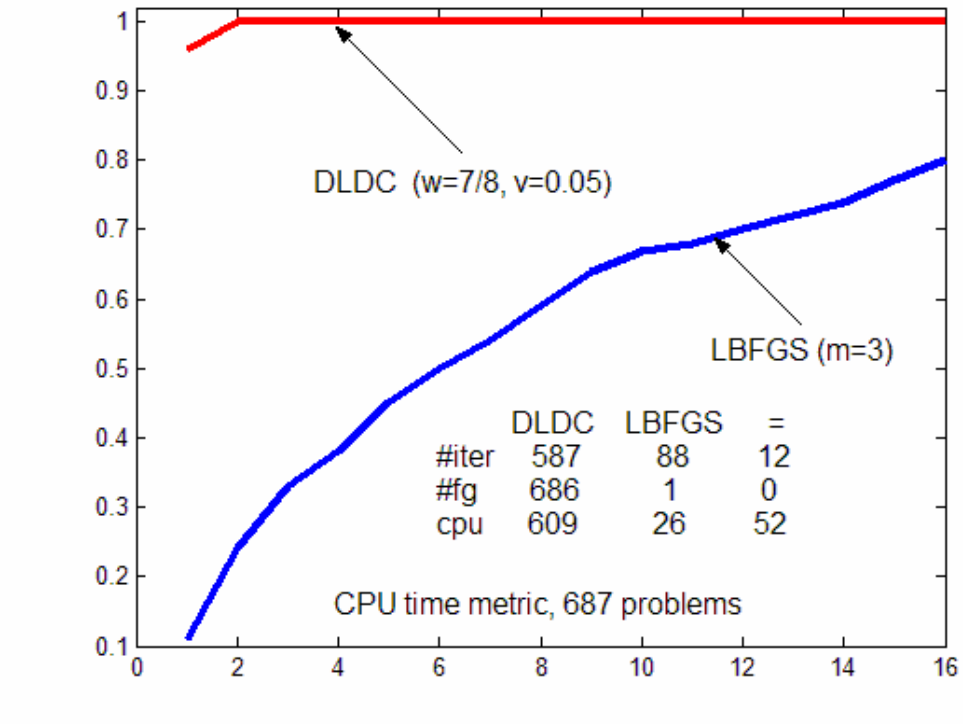

Fig. 7. DLDC ($w = 7/8, v = 0.05$) versus L-BFGS (m=3) by Liu and Nocedal.

In the following, in Figure 8, we present the performance profile of DLDC ($w = 7/8, v = 0.05$) versus HS, PRP, CG_DESCENT and L-BFGS (m=3), subject to cpu time metric. We see that among these algorithms DLDC is top performer. Observe that these algorithms can be classified in three major classes: DLDC and CG_DESCENT, HS and PRP, and finally the limited memory quasi-Newton L-BFGS.

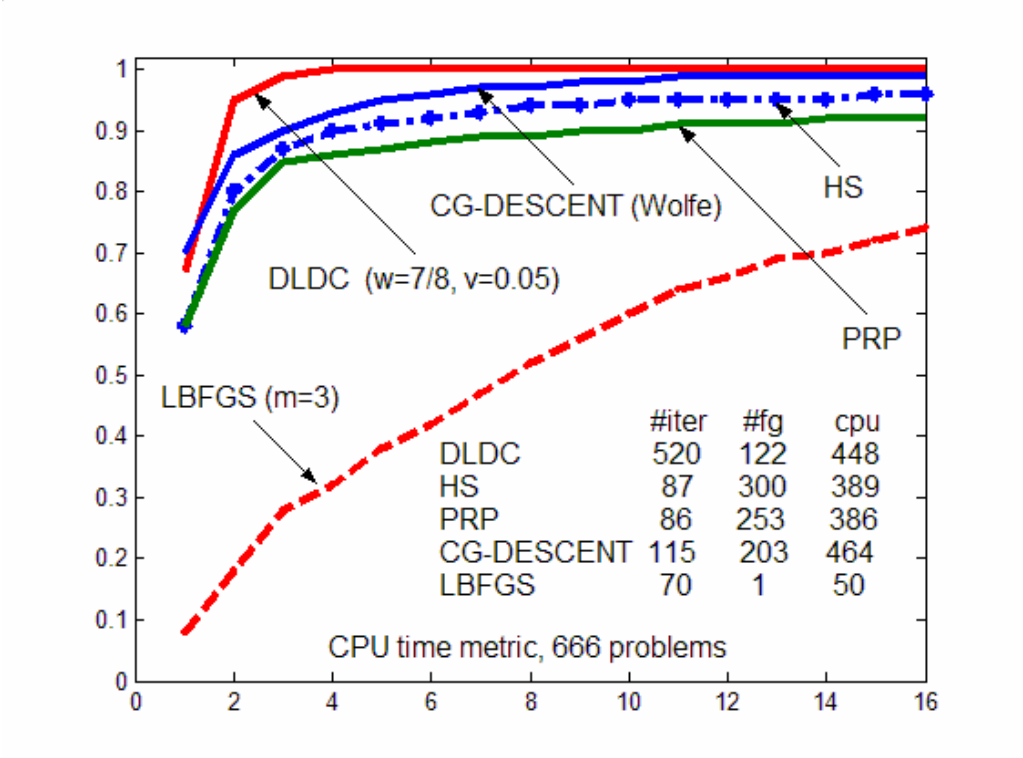

Fig. 8. DLDC ($w = 7/8$, $v = 0.05$) versus HS, PRP, CG_DESCENT and L-BFGS (m=3).

In order to see the performances of the algorithm we present a *sensitivity study* of DLDC subject to the variation of v and w parameters. Both these parameters emphasize the importance of the conjugacy condition and the sufficient descent condition, respectively. From (2.2), (2.3) and (2.6)-(2.11) we have

$$\frac{\partial d_{k+1}}{\partial w} = \frac{(y_k^T s_k)\|g_{k+1}\|^2}{\bar{\Delta}_k}\left(g_{k+1} - \frac{y_k^T g_{k+1}}{y_k^T s_k} s_k\right), \tag{5.2}$$

$$\frac{\partial d_{k+1}}{\partial v} = -\frac{(s_k^T g_{k+1})}{\bar{\Delta}_k}\left((s_k^T g_{k+1}) g_{k+1} - \|g_{k+1}\|^2 s_k\right). \tag{5.3}$$

Observe that if the line search is exact ($s_k^T g_{k+1} = 0$) then from (5.3) we see that the algorithm is not sensitive to the variation of v. However, in our algorithm the line search is not exact.

Table 1 presents the total number of iterations (#itert), the total number of function and its gradient evaluations (#fgt) and the total CPU time (cput) for solving the above set of 750 unconstrained optimization test problems for $w = 7/8$ and for different values of v. For example, for solving the set of 750 problems with $w = 7/8$ and $v = 0$, the total number of iteration is 260792, the total number of function and its gradient evaluations is 654859 and the total CPU time is 308.14 seconds, etc.

In Table 1 we have a computational evidence of the sensitivity of DLDC corresponding to a set of 12 numerical experiments subject to variation of v parameter. The best results corresponding to this set of 12 numerical experiments are obtained for $v = 0.05$. Subject to the CPU time metric the average of the total CPU time corresponding to these 12 numerical experiments, for solving 750 problems in each experiment, is 3742.13/12=311.84 seconds. The largest deviation is of 76.07 seconds and corresponds to the numerical experiment in which $v = 0.2$ Therefore, in all these 12 numerical experiments the maximum deviation is of 76.07/750=0.1 seconds per problem.

Table 1. Sensitivity of the DLDC subject to v. $w = 7/8$.

v	#itert	#fgt	cput
0	260792	654859	308.14
0.001	259291	641597	301.19
0.005	266787	663287	335.99
0.01	268870	691054	376.59
0.02	266274	623824	279.81
0.05	**258153**	**608602**	**277.44**
0.07	260234	641310	298.78
0.1	260116	649701	299.42
0.2	278611	679622	***387.91***
0.5	260216	633033	290.69
0.7	261112	625176	283.76
1	279770	664769	302.41

In the following we present the sensitivity of DLDC subject to the variation of w parameter. Table 2 presents the total number of iterations, the total number of function and its gradient evaluations and the total CPU time for solving the above set of 750 unconstrained optimization test problems for $v = 0.05$ and for 6 different values of w.

Table 2. Sensitivity of the DLDC subject to w. $v = 0.05$.

w	#itert	#fgt	cput
0.5	257634	609194	278.85
0.6	261100	628040	288.59
0.7	**256887**	**605325**	**274.09**
0.8	259572	627447	***292.52***
0.9	257143	612870	281.27
1	258642	613259	278.33

The best results corresponding to this set of 6 numerical experiments are obtained for $w = 0.7$. Subject to CPU time metric for solving 750 problems in each of these 6 numerical experiments the total CPU time difference is of 292.52 − 274.09 = 18.43 seconds. Therefore, in all these 6 numerical experiments the maximum deviation is of 18.43/750=0.024 seconds per problem. Observe that the average of the total CPU time corresponding to these 6 numerical experiments is 1693.65/6=282.27 seconds. The largest deviation is of 292.52 − 282.27 = 10.25 seconds. Therefore, in all these 6 numerical experiments the maximum deviation is of 10.25/750=0.013 seconds per problem. Practically, DLDC is very little sensitive to the variation of w.

We now present comparisons between DLDC and CG_DESCENT conjugate gradient algorithms for solving some applications from MINPACK-2 test problem collection [9]. In Table 3 we present these applications, as well as the values of their parameters. The infinite-dimensional version of these problems is transformed into a finite element approximation by triangulation. Thus a finite-dimensional minimization problem is obtained whose variables are the values of the picewise linear function at the vertices of the triangulation. The discretization steps are $nx = 1000$ and $ny = 1000$, thus obtaining minimization problems with 1,000,000 variables.

Table 3. Applications from MINPACK-2 collection.

A1	*Elastic-Plastic Torsion* [23, pp. 41-55], $c = 5$.
A2	*Pressure Distribution in a Journal Bearing* [13], $b = 10$, $\varepsilon = 0.1$.
A3	*Optimal Design with Composite Materials* [24], $\lambda = 0.008$.
A4	*Steady-State Combustion* [8, pp. 292-299], [10], $\lambda = 5$.
A5	*Minimal Surfaces with Enneper conditions* [31, pp. 80-85].

A comparison between DLDC ($v = 0.05,\ w = 0.6,$ Powell restart criterion, $\|\nabla f(x_k)\|_\infty \le 10^{-6},\ \rho = 10^{-4}$) and CG_DESCENT (Wolfe line search, default settings, $\|\nabla f(x_k)\|_\infty \le 10^{-6}$) for solving these applications is given in Table 4.

Table 4. Performance of DLDC and CG_DESCENT.
1,000,000 variables. cpu seconds.

	DLDC			CG_DESCENT		
	#iter	#fg	cpu	#iter	#fg	cpu
A1	1111	2253	**787.09**	1145	2291	1087.83
A2	2833	5694	**2151.86**	3368	6737	3369.77
A3	4777	9595	**5695.39**	4841	9684	8058.66
A4	1413	2864	**2340.41**	1806	3613	4213.00
A5	1279	2580	**1360.88**	1226	2453	1773.95
TOTAL	11413	22986	**12335.63**	12386	24778	18503.21

Form Table 4 we see that subject to the CPU time metric the DLDC algorithm is top performer again, and the difference is significant, about 6167.58 seconds for solving all these 5 applications.

The DLDC and CG_DESCENT algorithms (and codes) are different in many respects. Since both of them use the Wolfe line search (however, implemented in different manners), these codes mainly differ in their choice of the search direction. DLDC appears to generate a better search direction, on average. The direction d_{k+1} used in DLDC is more elaborate, it satisfies both the sufficient descent condition and the conjugacy condition in a restart environment. Although the update formulae (2.2), (2.3) and (2.7)-(2.11) are more complicated, this computational scheme proved to be more efficient and more robust in numerical experiments and applications. However, since each of these codes are different in the number of parameters which can be modified by the user to establish a context of optimization (CG_DESCENT has 26 parameters while DLDC has only 9 parameters) and in the amount of linear algebra required in each iteration, it is quite clear that different codes will be superior in different problem sets.

6. Conclusions

For solving large scale unconstrained optimization problems we have presented an accelerated conjugate gradient algorithm that for all $k \ge 0$ both the descent and the conjugacy conditions are guaranteed. In our algorithm the search direction is selected as a linear combination of $-g_{k+1}$ and s_k, where the coefficients in this linear combination are selected in such a way that both the descent and the conjugacy condition are satisfied at every iteration. The step length is modified by an acceleration scheme which proved to be very efficient in reducing the values of the minimizing function along the iterations. For a test set consisting of 750 problems with dimensions ranging between 1000 and 10,000, the CPU time performance profiles of DLDC was higher than those of HS, PRP, DY, hDY, CG_DESCENT with Wolfe line search and limited memory quasi-Newton method L-BFGS. A number of 5 applications from MINPACK2 test problem collection illustrate the performances of DLDC versus CG_DESCENT. At present, for the above test problems and applications it follows that DLDC is the fastest and the most robust conjugate gradient algorithm.

References

[1] N. Andrei, *An unconstrained optimization test functions collection.* Advanced Modeling and Optimization, 10 (2008), pp. 147-161.

[2] N. Andrei, *An acceleration of gradient descent algorithm with backtracking for unconstrained optimization*, Numerical Algorithms, 42 (2006), pp. 63-73.

[3] N. Andrei, *Scaled conjugate gradient algorithms for unconstrained optimization*. Computational Optimization and Applications, 38 (2007), pp. 401-416.
[4] N. Andrei, *Scaled memoryless BFGS preconditioned conjugate gradient algorithm for unconstrained optimization.* Optimization Methods and Software, 22 (2007), 561-571.
[5] N. Andrei, *A scaled BFGS preconditioned conjugate gradient algorithm for unconstrained optimization*. Applied Mathematics Letters, 20 (2007), 645-650.
[6] N. Andrei, *Numerical comparison of conjugate gradient algorithms for unconstrained optimization.* Studies in Informatics and Control, 16 (2007), pp.333-352.
[7] N. Andrei, *Acceleration of conjugate gradient algorithms for unconstrained optimization.* Applied Mathematics and Computation, 213 (2009), 361-369.
[8] R. Aris, *The mathematical theory of diffusion and reaction in permeable catalysts.* Oxford, 1975.
[9] B.M., Averick, R.G., Carter, J.J., Moré, Xue, G.L. *The MINPACK-2 test problem collection.* Mathematics and Computer Science Division, Argonne National Laboratory, Preprint MCS-P153-0692, June 1992.
[10] J. Bebernes, D. Eberly, *Mathematical problems from combustion theory.* Applied Mathematical Sciences 83, Springer-Verlag, 1989.
[11] E. Birgin, J.M. Martínez, *A spectral conjugate gradient method for unconstrained optimization*, Applied Math. and Optimization, 43, pp.117-128, 2001.
[12] I. Bongartz, A.R. Conn, N.I.M. Gould, P.L. Toint, *CUTE: constrained and unconstrained testing environments*, ACM Trans. Math. Software, 21, pp.123-160, 1995.
[13] G., Cimatti, *On a problem of the theory of lubrication governed by a variational inequality.* Appl. Math. Potim., 3 (1977) 227-242.
[14] Y.H. Dai, *New properties of a nonlinear conjugate gradient method.* Numer. Math., 89 (2001), pp.83-98.
[15] Y.H. Dai, L.Z. Liao, *New conjugacy conditions and related nonlinear conjugate gradient methods*. Applied Mathematical Optimization, 43 (2001), pp. 87-101.
[16] Y.H. Dai, L.Z. Liao, Li Duan, *On restart procedures for the conjugate gradient method.* Numerical Algorithms 35 (2004), pp. 249-260.
[17] Y.H. Dai, Y. Yuan, *An efficient hybrid conjugate gradient method for unconstrained optimization,* Ann. Oper. Res., 103 (2001) 33-47.
[18] Y.H. Dai, Han, J.Y., Liu, G.H., Sun, D.F., Yin, .X., Yuan, Y., *Convergence properties of nonlinear conjugate gradient methods*. SIAM Journal on Optimization 10 (1999), 348-358.
[19] E.D., Dolan, J.J. Moré, *Benchmarking optimization software with performance profiles*, Math. Programming **91**, 201-213 (2002)
[20] R. Fletcher, *Practical Optimization:Vol. 1: Unconstrained optimization.* John Wiley and Sons, Chichester, 1980.
[21] R., Fletcher, Reeves, C., *Function minimization by conjugate gradients*, Comput. J., 7 (1964), pp.149-154.
[22] J.C. Gilbert, J. Nocedal, *Global convergence properties of conjugate gradient methods for optimization*, SIAM J. Optim., 2 (1992), pp. 21-42.
[23] R., Glowinski, *Numerical Methods for Nonlinear Variational Problems.* Springer-Verlag, Berlin, 1984.
[24] J., Goodman, R., Kohn, L., Reyna, *Numerical study of a relaxed variational problem from optimal design.* Comput. Methods Appl. Mech. Engrg., 57, 1986, pp.107-127.
[25] W.W. Hager, H. Zhang, "*A new conjugate gradient method with guaranteed descent and an efficient line search*", SIAM Journal on Optimization, 16 (2005) 170-192.
[26] W.W. Hager, H. Zhang, *A survey of nonlinear conjugate gradient methods.* Pacific journal of Optimization, 2 (2006) 35-58.
[27] M.R. Hestenes, E.L. Stiefel, *Methods of conjugate gradients for solving linear systems,* J. Research Nat. Bur. Standards, 49 (1952) 409-436.

[28] G. Li, C. Tang, Z. Wei, *New conjugacy condition and related new conjugate gradient methods for unconstrained optimization.* Journal of Computational and Applied Mathematics, 202 (2007) 523-539.
[29] D.C. Liu, J. Nocedal, *On the limited memory BFGS method for large scale optimization methods.* Mathematical Programming, 45 (1989), pp. 503-528.
[30] J.J. Moré, D.J. Thuente, *Line search algorithms with guaranteed sufficient decrease.* ACM Transactions on Mathematical Software, 20 (1994) 286-307.
[31] J.C.C. Nitsche, *Lectures on minimal surfaces.* Vol.1, Cambridge University Press, 1989.
[32] J. Nocedal, *Conjugate gradient methods and nonlinear optimization.* In Linear and nonlinear Conjugate Gradient related methods, L. Adams and J.L. Nazareth (eds.), SIAM, 1996, pp.9-23.
[33] A. Perry, *A modified conjugate gradient algorithm.* Operations Research 26 (1978), pp. 1073-1078.
[34] E. Polak, *Computational methods in optimization: A unified approach.* Academic Press, New York, 1971.
[35] E. Polak, G. Ribière, *Note sur la convergence de directions conjuguée,* Rev. Francaise Informat Recherche Operationelle, 3e Année 16 (1969) 35-43.
[36] B.T. Polyak, *The conjugate gradient method in extreme problems.* USSR Comp. Math. Math. Phys., 9 (1969) 94-112.
[37] M.J.D. Powell, *Nonconvex minimization calculations and the conjugate gradient method.* Numerical Analysis (Dundee, 1983), Lecture Notes in Mathematics, vol. 1066, Springer-Verlag, Berlin, 1984, pp. 122-141.
[38] M.J.D. Powell, *Some convergence properties of the conjugate gradient method.* Mathematical Programming, 11 (1976), pp.42-49.
[39] D.F. Shanno, *Conjugate gradient methods with inexact searches.* Mathematics of Operations Research 3 (1978), pp. 244-256.
[40] D.F. Shanno, K.H. Phua, *Algorithm 500. Minimization of unconstrained multivariate functions,* ACM Trans. on Math. Soft., 2 (1976) 87-94.
[41] P. Wolfe, *Convergence conditions for ascent methods,* SIAM Rev. 11 (1969) 226-235.
[42] P. Wolfe, *Convergence conditions for ascent methods II: some corrections*, SIAM Rev. 13 (1971) 185-188.
[43] Y. Yuan, *Analysis on the conjugate gradient method.* Technical Report, Computing Center, Academia Sinica, China, 1990.
[44] Y. Yuan, J. Stoer, *A subspace study on conjugate gradient algorithms.* Z. Angew. Math. Mech., 75 (1995), pp. 69-77.
[45] G. Zoutendijk, *Nonlinear programming computational methods.* In: J. Abadie (Ed.) Integer and Nonlinear Programming, North-Holland, Amsterdam, 1970, pp. 37-86.

January 29, 2010