New Accelerated Conjugate Gradient Algorithms as modification of Dai-Yuan's computational scheme for 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. New accelerated nonlinear conjugate gradient algorithms which are mainly modifications of the Dai and Yuan's for unconstrained optimization are proposed. Using the exact line search, the algorithm reduces to the Dai and Yuan conjugate gradient computational scheme. For inexact line search the algorithm satisfies the sufficient descent condition. Since the step lengths in conjugate gradient algorithms may differ from 1 by two order of magnitude and tend to vary in a very unpredictable manner, the algorithms are equipped with an acceleration scheme able to improve the efficiency of the algorithms. Computational results for a set consisting of 750 unconstrained optimization test problems show that these new conjugate gradient algorithms substantially outperform the Dai-Yuan conjugate gradient algorithm and its hybrid variants.

Keywords: Unconstrained optimization; conjugate gradient method; sufficient descent condition; conjugacy condition; Newton direction; numerical comparisons.
AMS 2000 Mathematics Subject Classification: 49M07, 49M10, 90C06, 65K05

1. Introduction

Conjugate gradient methods represent an important class of unconstrained optimization algorithms with strong local and global convergence properties and modest memory requirements. A survey on their definition including 40 conjugate gradient algorithms for unconstrained optimization is given by Andrei [6]. A discussion of development of different versions of nonlinear conjugate gradient methods, with special attention to global convergence properties is presented by Hager and Zhang [19].

In this paper we suggest new nonlinear conjugate gradient algorithms which are mainly modifications of the Dai and Yuan [15] conjugate gradient computational scheme. In these algorithms the direction d_{k+1} is computed as a linear combination between $-g_{k+1}$ and s_k, i.e. $d_{k+1} = -\theta_{k+1} g_{k+1} + \beta_k^N s_k$, where $s_k = x_{k+1} - x_k$. The parameter θ_k is computed in such a way that the direction d_{k+1} is the Newton direction or it satisfies the conjugacy condition. On the other hand, β_k^N is a proper modification of the Dai and Yuan's computational scheme in such a way that the direction d_{k+1} at every iteration satisfies the sufficient descent condition. For the exact line search the proposed algorithms reduce to the Dai and Yuan conjugate gradient computational scheme.

The paper has the following structure. In Section 2 we present the development of the conjugate gradient algorithms with sufficient descent condition as modifications of the Dai-Yuan computational scheme, while in section 3 we prove the global convergence of these algorithms under strong Wolfe line search conditions. In Section 4 we present the accelerated algorithms, showing their global convergence and in Section 5 we compare the computational performance of the new conjugate gradient schemes against the Dai and Yuan method and its hybrid variants [16] using 750 unconstrained optimization test problems from the CUTE [11] library along with some other large-scale unconstrained optimization problems presented in [8]. Using the Dolan and Moré performance profiles [18] we prove these new accelerated conjugate gradient algorithms outperform the Dai-Yuan algorithm as well as its hybrid variants.

2. Modifications of the Dai-Yuan conjugate gradient algorithm

For solving the unconstrained optimization problem

$$min\left\{ f(x) : x \in R^n \right\}, \tag{2.1}$$

where $f : R^n \to R$ is continuously differentiable and bounded below we consider a nonlinear conjugate gradient algorithm:

$$x_{k+1} = x_k + \alpha_k d_k, \tag{2.2}$$

where the stepsize α_k is positive and the directions d_k are computed by the rule:

$$d_{k+1} = -\theta_{k+1} g_{k+1} + \beta_k^N s_k, \quad d_0 = -g_0, \tag{2.3}$$

where

$$\beta_k^N = \frac{\|g_{k+1}\|^2}{y_k^T s_k} - \frac{\|g_{k+1}\|^2 (s_k^T g_{k+1})}{(y_k^T s_k)^2}, \tag{2.4}$$

and θ_{k+1} is a parameter which follows to be determined. Here $g_k = \nabla f(x_k)$ and $y_k = g_{k+1} - g_k$, $s_k = x_{k+1} - x_k$.

Observe that if f is a quadratic function and α_k is selected to achieve the exact minimum of f in the direction d_k, then $s_k^T g_{k+1} = 0$ and the formula (2.4) for β_k^N reduces to the Dai and Yuan computational scheme [15]. However, in this paper we refer to general nonlinear functions and inexact line search.

We were led to this computational scheme by modifying the Dai and Yuan algorithm

$$\beta_k^{DY} = \frac{g_{k+1}^T g_{k+1}}{y_k^T s_k},$$

in order to have the sufficient descent condition, as well as some other properties for an efficient conjugate gradient algorithm. Using a standard Wolfe line search, the Dai and Yuan method always generates descent directions and under Lipschitz assumption it is globally convergent. In [12] Dai established a remarkable property relating the descent directions to the sufficient descent condition, showing 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 $\le j$. In our algorithm the parameter β_k is selected in such a manner that the sufficient descent condition is satisfied at every iteration. As we know, despite the strong convergence theory that has been developed for the Dai and Yuan method, it is susceptible to jamming, that is it begins to take small steps without making significant progress to the minimum. When iterates jam, y_k becomes tiny while $\|g_k\|$ is bounded away from zero. Therefore, β_k^N is a proper modification of the β_k^{DY}.

Theorem 2.1. *If* $\theta_{k+1} \ge 1/4$, *then the direction* $d_{k+1} = -\theta_{k+1} g_{k+1} + \beta_k^N s_k$, *(*$d_0 = -g_0$*), where* β_k^N *is given by (2.4) satisfies the sufficient descent condition*

$$g_{k+1}^T d_{k+1} \le -\left(\theta_{k+1} - \frac{1}{4} \right) \|g_{k+1}\|^2. \tag{2.5}$$

Proof. Since $d_0 = -g_0$, we have $g_0^T d_0 = -\|g_0\|^2$, which satisfy (2.5). Multiplying (2.3) by g_{k+1}^T, we have

$$g_{k+1}^T d_{k+1} = -\theta_{k+1}\|g_{k+1}\|^2 + \frac{(g_{k+1}^T g_{k+1})(g_{k+1}^T s_k)}{y_k^T s_k} - \frac{\|g_{k+1}\|^2 (s_k^T g_{k+1})^2}{(y_k^T s_k)^2}. \tag{2.6}$$

Now, using the inequality $u^T v \le \frac{1}{2}(\|u\|^2 + \|v\|^2)$, where $u, v \in R^n$, we have:

$$\begin{aligned}\frac{(g_{k+1}^T g_{k+1})(g_{k+1}^T s_k)}{y_k^T s_k} &= \frac{\left[(y_k^T s_k) g_{k+1}/\sqrt{2}\right]^T \left[\sqrt{2}(g_{k+1}^T s_k) g_{k+1}\right]}{(y_k^T s_k)^2} \\ &\le \frac{\frac{1}{2}\left[\frac{1}{2}(y_k^T s_k)^2 \|g_{k+1}\|^2 + 2(g_{k+1}^T s_k)^2 \|g_{k+1}\|^2\right]}{(y_k^T s_k)^2} \\ &= \frac{1}{4}\|g_{k+1}\|^2 + \frac{(g_{k+1}^T s_k)^2 \|g_{k+1}\|^2}{(y_k^T s_k)^2}.\end{aligned} \tag{2.7}$$

Using (2.7) in (2.6) we get (2.5). ■

To conclude, the sufficient descent condition from (2.5), the quantity $\theta_{k+1} - 1/4$ is required to be nonnegative. Supposing $\theta_{k+1} - 1/4 > 0$, then the direction given by (2.3) and (2.4) is a descent direction. Dai and Yuan [15, 16] present conjugate gradient schemes with the property that $g_k^T d_k < 0$ when $y_k^T s_k > 0$. If f is strongly convex or the line search satisfies the Wolfe conditions, then $y_k^T s_k > 0$ and the Dai and Yuan scheme yield descent. In our algorithm observe that, if for all k, $\theta_{k+1} \ge 1/4$, and the line search satisfies the Wolfe conditions, then for all k the search direction (2.3) and (2.4) satisfy the sufficient descent condition. Note that in (2.5) we bound $g_{k+1}^T d_{k+1}$ by $-(\theta_{k+1} - 1/4)\|g_{k+1}\|^2$, while for scheme of Dai and Yuan only the non-negativity of $g_{k+1}^T d_{k+1}$ is established.
To determine the parameter θ_{k+1} in (2.3) we suggest the following two procedures.

A) Our motivation to get a good algorithm for solving (2.1) is to choose the parameter θ_{k+1} in such a way that for every $k \ge 1$ the direction d_{k+1} given by (2.3) be the Newton direction. Therefore, from the equation

$$-\nabla^2 f(x_{k+1})^{-1} g_{k+1} = -\theta_{k+1} g_{k+1} + \beta_k^N s_k \tag{2.8}$$

after some algebra we get

$$\theta_{k+1} = \frac{1}{s_k^T \nabla^2 f(x_{k+1}) g_{k+1}} \left[\frac{\|g_{k+1}\|^2}{y_k^T s_k}\left(1 - \frac{s_k^T g_{k+1}}{y_k^T s_k}\right) s_k^T \nabla^2 f(x_{k+1}) s_k + s_k^T g_{k+1}\right]. \tag{2.9}$$

The salient point in this formula for θ_{k+1} is the presence of the Hessian. For large-scale problems, choices for the update parameter that do not require the evaluation of the Hessian matrix are often preferred in practice to the methods that require the Hessian in each iteration. Therefore, in order to have an algorithm for solving large-scale problems we assume that in (2.8) we use an approximation B_{k+1} of the true Hessian $\nabla^2 f(x_{k+1})$ and let B_{k+1} satisfy the quasi-Newton equation $B_{k+1} s_k = y_k$. This leads us to:

$$\theta_{k+1} = \frac{1}{y_k^T g_{k+1}} \left[\|g_{k+1}\|^2 - \frac{\|g_{k+1}\|^2 (s_k^T g_{k+1})}{y_k^T s_k} + s_k^T g_{k+1} \right]. \tag{2.10}$$

Observe that if θ_{k+1} given by (2.10) is greater than or equal to $1/4$, then according to Theorem 2.1 the direction (2.3) satisfies the sufficient descent condition (2.5). On the other hand, if in (2.10) $\theta_{k+1} < 1/4$, then we take *ex abrupto* $\theta_{k+1} = 1$ in (2.3).

B) The second procedure is based on the conjugacy condition. Dai and Liao [13] introduced the conjugacy condition $y_k^T d_{k+1} = -t s_k^T g_{k+1}$, where $t \geq 0$ is a scalar. This is indeed very reasonable since in real computation the inexact line search is generally used. However, this condition is very dependent on the nonnegative parameter t, for which we do not know any formula to choose in an optimal manner. Therefore, even if in our developments we use the inexact line search we adopt here a more conservative approach and consider the conjugacy condition $y_k^T d_{k+1} = 0$. This leads us to:

$$\theta_{k+1} = \frac{1}{y_k^T g_{k+1}} \left[\|g_{k+1}\|^2 - \frac{\|g_{k+1}\|^2 (s_k^T g_{k+1})}{y_k^T s_k} \right]. \tag{2.11}$$

As above, if θ_{k+1} given by (2.11) is greater than or equal to $1/4$, then according to Theorem 2.1 the direction (2.3) satisfies the sufficient descent condition (2.5). On the other hand, if in (2.11) $\theta_{k+1} < 1/4$, then we take $\theta_{k+1} = 1$ in (2.3).

The line search in the conjugate gradient algorithms for α_k computation is often based on the standard Wolfe conditions:

$$f(x_k + \alpha_k d_k) - f(x_k) \leq \rho \alpha_k g_k^T d_k, \tag{2.12}$$

$$g_{k+1}^T d_k \geq \sigma g_k^T d_k, \tag{2.13}$$

where d_k is a descent direction and $0 < \rho \leq \sigma < 1$.

In [16] Dai and Yuan proved the global convergence of a conjugate gradient algorithm for which $\beta_k = \beta_k^{DY} t_k$, where $t_k \in [-c, 1]$ with $c = (1-\sigma)/(1+\sigma)$. Our algorithm is a modification of the Dai and Yuan's with the following property.

Observe that

$$\beta_k^N = \frac{\|g_{k+1}\|^2}{y_k^T s_k} \left[1 - \frac{s_k^T g_{k+1}}{y_k^T s_k} \right] = \beta_k^{DY} r_k, \tag{2.14}$$

where

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

From the second Wolfe condition it follows that $s_k^T g_{k+1} \geq \sigma s_k^T g_k = -\sigma y_k^T s_k + \sigma s_k^T g_{k+1}$, i.e.

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

Since by the Wolfe condition $y_k^T s_k > 0$, it follows that $\dfrac{s_k^T g_{k+1}}{y_k^T s_k} \geq \dfrac{-\sigma}{1-\sigma}$. Hence $r_k \leq \dfrac{1}{1-\sigma}$.

Therefore, $\beta_k^N \leq \beta_k^{DY} \dfrac{1}{1-\sigma}$.

3. Convergence analysis

In this section we analyze the convergence of the algorithm (2.2), (2.3), (2.4) and (2.10) or (2.11) where $d_0 = -g_0$. 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.*

(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 (2.2) is obtained by the strong Wolfe line search, that is,

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

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

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

For any conjugate gradient algorithm with strong Wolfe line search, we have the following results given by lemma 3.1 and lemma 3.2, which were first proved by Zoutendijk [28] and Wolfe [26, 27]. For completeness, we present them here without proofs.

Lemma 3.1. *Let* α_k *be obtained by the strong Wolfe line search (3.1) and (3.2). Suppose that the assumptions (i) and (ii) and* d_k *is a descent direction. Then*

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

Lemma 3.2. *Let* α_k *be obtained by the strong Wolfe line search (3.1) and (3.2). Suppose that the assumptions (i) and (ii) and* d_k *is a descent direction. Then the so-called Zoutendijk condition holds*

$$\sum_{k=0}^{\infty} \frac{(g_k^T d_k)^2}{\|d_k\|^2} < \infty. \ \blacksquare \tag{3.4}$$

Based on these results, for conjugate gradient method (2.2) where

$$d_{k+1} = -\theta_{k+1} g_{k+1} + \beta_k^N d_k \tag{3.5}$$

and $\theta_k > 1/4$, with strong Wolfe line search, we can prove the following lemma and its corollary which are essential for the convergence of our algorithms. Lemma 3.3 is a variant of the Theorem 2.3 of Dai *et al.* [17].

Lemma 3.3. *Suppose that the assumptions (i) and (ii) hold. Consider the conjugate gradient method (2.2) and (3.5) where* $\theta_k > 1/4$, *with strong Wolfe line search (3.1) and (3.2). Then either*

$$\liminf_{k \to \infty} \|g_k\| = 0, \tag{3.6}$$

or

$$\sum_{k=0}^{\infty} \frac{\|g_k\|^4}{\|d_k\|^2} < \infty. \tag{3.7}$$

Proof. Since for any $k \ge 0$, $\theta_k > 1/4$, it follows that d_k is a descent direction. From (3.5) since for all $k \ge 0$, $g_k^T d_k < 0$ we have

$$\|d_k\|^2 \ge (\beta_{k-1}^N)^2 \|d_{k-1}\|^2 - \theta_k^2 \|g_k\|^2 . \tag{3.8}$$

On the other hand, from (3.5) we get

$$g_k^T d_k - \beta_{k-1}^N g_k^T d_{k-1} = -\theta_k \|g_k\|^2 .$$

Therefore, from the strong Wolfe condition we have that

$$\left|g_k^T d_k\right| + \sigma \left|\beta_{k-1}^N\right| \left|g_{k-1}^T d_{k-1}\right| \ge \theta_k \|g_k\|^2 . \tag{3.9}$$

But for any $a, b, \sigma \ge 0$ the following inequality $(a+\sigma b)^2 \le (1+\sigma^2)(a^2+b^2)$ holds. Considering $a = \left|g_k^T d_k\right|$ and $b = \left|\beta_{k-1}^N\right| \left|g_{k-1}^T d_{k-1}\right|$, then (3.9) yields to

$$(g_k^T d_k)^2 + (\beta_{k-1}^N)^2 (g_{k-1}^T d_{k-1})^2 \ge c \|g_k\|^4 , \tag{3.10}$$

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

$$\begin{aligned}
\frac{(g_k^T d_k)^2}{\|d_k\|^2} + \frac{(g_{k-1}^T d_{k-1})^2}{\|d_{k-1}\|^2} &= \frac{1}{\|d_k\|^2}\left[(g_k^T d_k)^2 + \frac{\|d_k\|^2}{\|d_{k-1}\|^2}(g_{k-1}^T d_{k-1})^2\right] \\
&\ge \frac{1}{\|d_k\|^2}\left[(g_k^T d_k)^2 + (\beta_{k-1}^N)^2 (g_{k-1}^T d_{k-1})^2 - \frac{(g_{k-1}^T d_{k-1})^2}{\|d_{k-1}\|^2}\|g_k\|^2\right] \\
&\ge \frac{1}{\|d_k\|^2}\left[c\|g_k\|^4 - \frac{(g_{k-1}^T d_{k-1})^2}{\|d_{k-1}\|^2}\|g_k\|^2\right] \\
&= \frac{\|g_k\|^4}{\|d_k\|^2}\left[c - \frac{(g_{k-1}^T d_{k-1})^2}{\|d_{k-1}\|^2}\frac{1}{\|g_k\|^2}\right].
\end{aligned} \tag{3.11}$$

From lemma 3.2 we know that

$$\lim_{k\to\infty} \frac{(g_{k-1}^T d_{k-1})^2}{\|d_{k-1}\|^2} = 0.$$

Therefore, if (3.6) is not true, then

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

Therefore, from (3.11) we get that

$$\frac{(g_k^T d_k)^2}{\|d_k\|^2} + \frac{(g_{k-1}^T d_{k-1})^2}{\|d_{k-1}\|^2} \ge \frac{c}{2}\frac{\|g_k\|^4}{\|d_k\|^2}$$

holds for all sufficiently large k. Hence, the inequality (3.7) follows from Zoutendijk condition (3.4) in lemma 3.2. ■

Corollary 3.1. *Suppose that the assumptions (i) and (ii) hold and consider any conjugate gradient method (2.2) and (3.5), where d_k is a descent direction, i.e. $\theta_k > 1/4$, and α_k is obtained by the strong Wolfe line search (3.1) and (3.2). If*

$$\sum_{k\ge 1} \frac{1}{\|d_k\|^2} = \infty , \tag{3.12}$$

then

$$\liminf_{k\to\infty} \|g_k\| = 0. \tag{3.13}$$

Proof. Suppose that there is a positive constant γ such that $\|g_k\| \geq \gamma$ for all $k \geq 0$. Then, from lemma 3.3 we have

$$\sum_{k\geq 0} \frac{1}{\|d_k\|^2} \leq \frac{1}{\gamma^4} \sum_{k\geq 0} \frac{\|g_k\|^4}{\|d_k\|^2} < \infty.$$

However, this contradicts (3.12) from the corollary 3.1, i.e. the corollary 3.1 is true. ■

Theorem 3.1. *Suppose that the assumptions (i) and (ii) hold and consider the algorithm (2.2), (2.3), (2.4) and (2.10) or (2.11), where* d_{k+1} *is a descent direction and* α_k *is obtained by the strong Wolfe line search (3.1) and (3.2). If there exists a constant* $\gamma \geq 0$ *such* $\gamma \leq \|\nabla f(x)\|$, $1/4 \leq \theta_k \leq \tau$, *where* τ *is a positive constant and the angle* φ_k *between* g_k *and* d_k *is bounded, i.e.* $\cos\varphi_k \leq \xi \leq 0$ *for all* $k = 0,1,\ldots$, *then the algorithm satisfies* $\liminf_{k\to\infty} g_k = 0$.

Proof. Observe that $y_k^T s_k = g_{k+1}^T s_k - g_k^T s_k \geq (\sigma - 1) g_k^T s_k$. But $g_k^T s_k = \|g_k\| \|s_k\| \cos\varphi_k$. Since d_k is a descent direction it follows that $g_k^T s_k \leq \|g_k\| \|s_k\| \xi \leq 0$ for all $k = 0,1,\ldots$, i.e.

$$y_k^T s_k \geq -(1-\sigma) \|g_k\| \|s_k\| \xi.$$

With these

$$\beta_k^N \leq \frac{\|g_{k+1}\|^2}{y_k^T s_k} \frac{1}{1-\sigma} \leq \frac{\|g_{k+1}\|^2}{-(1-\sigma)^2 \xi \|g_k\| \|s_k\|} \leq \frac{\Gamma^2}{-(1-\sigma)^2 \xi \gamma \|s_k\|} = \frac{\eta}{\|s_k\|},$$

where

$$\eta = \frac{\Gamma^2}{-(1-\sigma)^2 \xi \gamma}.$$

Therefore

$$\|d_{k+1}\| \leq |\theta_{k+1}| \|g_{k+1}\| + |\beta_k^N| \|s_k\| \leq \tau\Gamma + \frac{\eta}{\|s_k\|} \|s_k\| = \tau\Gamma + \eta.$$

This relation shows that

$$\sum_{k\geq 1} \frac{1}{\|d_k\|^2} \geq \frac{1}{(\tau\Gamma + \eta)^2} \sum_{k\geq 1} 1 = \infty.$$

Hence, from corollary 3.1 it follows that $\liminf_{k\to\infty} \|g_k\| = 0$. ■

4. AMDYN and AMDYC Algorithms

Nocedal [21] pointed out that in conjugate gradient methods the step lengths may differ from 1 in a very unpredictable manner. 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 methods and the limited memory quasi Newton method by Liu and Nocedal [20], show that the latter is more successful [7]. One explanation of 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 of the above conjugate gradient algorithms. 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 (see [5] and [9]). In accelerated algorithm instead of (2.2) the new estimation of the minimum point is computed as

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

where

$$\gamma_m = -\frac{a_k}{b_k},$$

$a_k = \alpha_k g_k^T d_k$, $b_k = -\alpha_k (g_k - g_z)^T d_k$, $z = x_k + \alpha_k d_k$ and $g_z = \nabla f(z)$. Hence, if $b_k \neq 0$, then $x_{k+1} = x_k + \gamma_k \alpha_k d_k$, otherwise $x_{k+1} = x_k + \alpha_k d_k$. Therefore, using the definitions of g_k, s_k, y_k and the above acceleration scheme we present the following conjugate gradient algorithms which are accelerated, modified versions of the Dai and Yuan algorithm with Newton direction (AMDYN) or with conjugacy condition (AMDYC).

AMDYN and AMDYC Algorithms

Step 1. Initialization. Select $x_0 \in R^n$ and the parameters $0 < \rho < \sigma < 1$. Compute $f(x_0)$ and g_0. Consider $d_0 = -g_0$ and $\alpha_0 = 1/\|g_0\|$. Set $k = 0$.

Step 2. Test for continuation of iterations. If $\|g_k\|_\infty \leq 10^{-6}$, then stop, else set $k = k+1$.

Step 3. Line search. Compute α_k satisfying the Wolfe line search conditions (2.12) and (2.13).

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

Step 5. Compute: $a_k = \alpha_k g_k^T d_k$, and $b_k = -\alpha_k y_k^T d_k$.

Step 6. Acceleration. If $b_k \neq 0$, then compute $\gamma_k = -a_k / b_k$ and update the variables as $x_{k+1} = x_k + \gamma_k \alpha_k d_k$, 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. θ_{k+1} computation. For the algorithm AMDYN, θ_{k+1} is computed as in (2.10). For the algorithm AMDYC, θ_{k+1} is computed as in (2.11). If $\theta_{k+1} < 1/4$, then we set $\theta_{k+1} = 1$.

Step 8. Direction computation. Compute $d = -\theta_{k+1} g_{k+1} + \beta_k^N s_k$, where β_k^N is computed as in (2.4). If

$$g_{k+1}^T d \leq -10^{-3} \|d\|_2 \|g_{k+1}\|_2, \tag{4.1}$$

then define $d_{k+1} = d$, otherwise set $d_{k+1} = -g_{k+1}$. Compute the initial guess $\alpha_k = \alpha_{k-1} \|d_{k-1}\| / \|d_k\|$, set $k = k+1$ and continue with 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 (2.12) and (2.13). In our algorithm when the angle between d and $-g_{k+1}$ is not acute enough, then we restart the algorithm with the negative gradient $-g_{k+1}$ [10]. More sophisticated reasons for restarting the algorithms have been proposed in the literature [24], but we are interested in the performance of a conjugate gradient algorithm that uses this restart criterion, associated to a direction satisfying the sufficient descent condition. Under reasonable assumptions, conditions (2.12), (2.13) and (4.1) are sufficient to prove the global convergence of the algorithm.

The initial selection of the step length crucially affects the practical behaviour of the algorithm. At every iteration $k \geq 1$ the starting guess for the step α_k in the line search is computed as $\alpha_{k-1} \|d_{k-1}\|_2 / \|d_k\|_2$. This selection, was considered for the first time by Shanno and Phua in CONMIN [25]. It is also considered in the packages: SCG by Birgin and Martínez [10] and in SCALCG by Andrei [1-4, 7].

For uniformly convex functions, like in [9] we can prove that the sequence generated by AMDYN or AMDYC converges linearly to the solution of the problem (2.1).

Proposition 4.1. *Suppose that f is a uniformly convex function on the level set $S=\{x: f(x)\le f(x_0)\}$, and d_k satisfies the sufficient descent condition $g_k^T d_k < -c_1\|g_k\|^2$, where $c_1>0$, and $\|d_k\|^2 \le c_2\|g_k\|^2$, where $c_2>0$. Then the sequence generated by AMDYN or AMDYC converges linearly to x^*, solution to the problem (2.1).* ■

5. Numerical results and comparisons

In this section we present the computational performance of a Fortran implementation of the AMDYN and AMDYC algorithms on a set of 750 unconstrained optimization test problems. We selected 75 large-scale unconstrained optimization problems in extended or generalized form [8]. For each function we have considered ten numerical experiments with the increasing number of variables $n = 1000, 2000, \ldots, 10000$. All algorithms implement the Wolfe line search conditions with $\rho = 0.0001$ and $\sigma = 0.9$, and the same stopping criterion $\|g_k\|_\infty \le 10^{-6}$, where $\|.\|_\infty$ is the maximum absolute component of a vector. 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, or the number of function-gradient evaluations, 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.

All codes are written in double precision Fortran and compiled with f77 (default compiler settings) on an Intel Pentium 4, 1.8GHz workstation. All these codes are authored by Andrei.

In the first set of numerical experiments we compare AMDYN versus AMDYC. In Table 1 we present the number of problems solved by these two algorithms with a minimum number of iterations (#iter), a minimum number of function and its gradient evaluations (#fg) and the minimum cpu time.

Table 1. Performance of AMDYN versus AMDYC. 750 problems.

	AMDYN	AMDYC	=
# iter	83	105	562
# fg	152	147	451
CPU	143	119	488

Both algorithms have similar performances. However, subject to cpu time metric, AMDYN proves to be slightly better.

In the second set of numerical experiments we compare AMDYN and AMDYC algorithms with the Dai and Yuan (DY) algorithm. Figures 1 and 2 present the Dolan-Moré performance profile for these algorithms subject to the cpu time metric. We see that both AMDYN and AMDYC is top performer, being more successful and more robust than the Dai and Yuan algorithm. When comparing AMDYN with the Dai and Yuan algorithm (Figure 1), subject to the number of iterations, we see that AMDYN was better in 619 problems (i.e. it achieved the minimum number of iterations in 619 problems). DY was better in 27 problems and they achieved the same number of iterations in 60 problems, etc. Out of 750 problems, only for 706 of them does the criterion (5.1) hold.

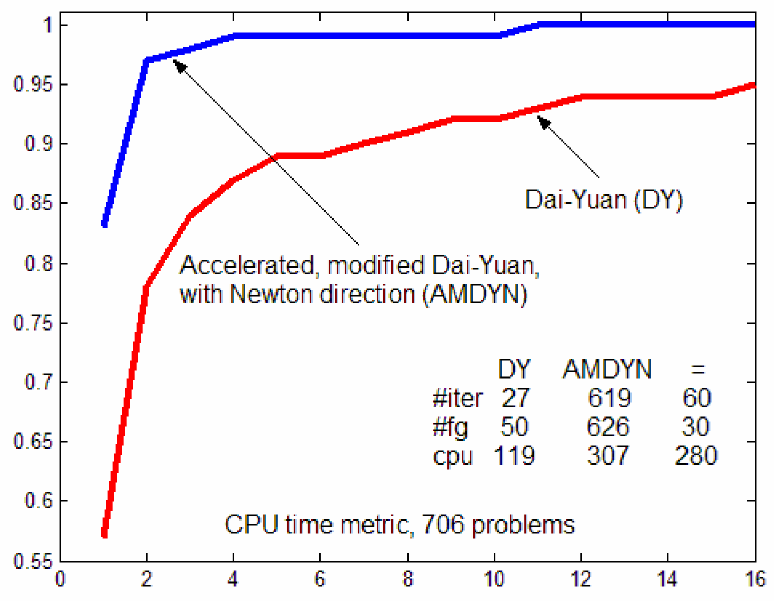

Fig. 1. Performance profile of AMDYN versus DY.

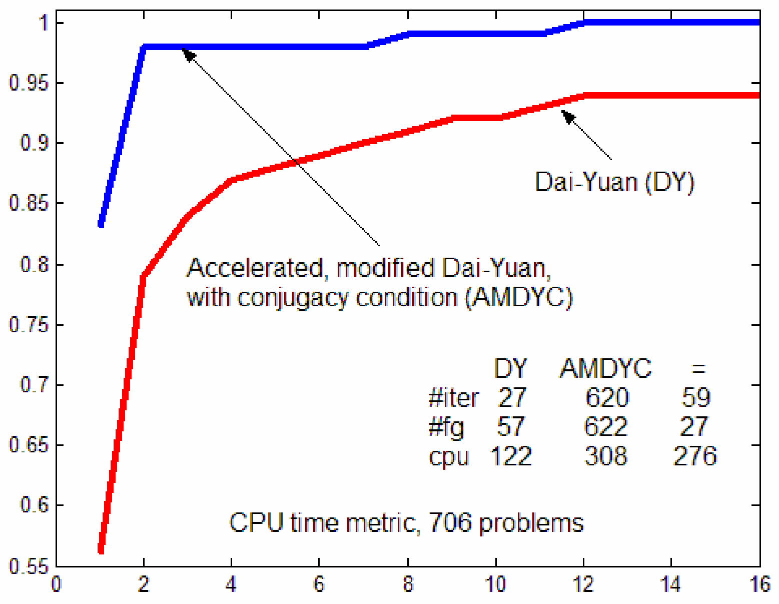

Fig. 2. Performance profile of AMDYC versus DY.

Dai and Yuan [16] studied the hybrid conjugate gradient algorithms and proposed the following two hybrid methods:

$$\beta_k^{hDY} = \max\left\{-\frac{1-\sigma}{1+\sigma}\beta_k^{DY}, \min\left\{\beta_k^{HS}, \beta_k^{DY}\right\}\right\}, \tag{5.2}$$

$$\beta_k^{hDYz} = \max\left\{0, \min\left\{\beta_k^{HS}, \beta_k^{DY}\right\}\right\}, \tag{5.3}$$

where $\beta_k^{HS} = y_k^T g_{k+1} / y_k^T s_k$, showing their global convergence when the Lipschitz assumption holds and the standard Wolfe line search is used. The numerical experiments of Dai and Ni [14] proved that the second hybrid method (hDYz) is the better, outperforming the Polak-Ribière [22] and Polyak [23] method. In the third set of numerical experiments we compare the Dolan-Moré performance profile of AMDYN and AMDYC versus Dai-Yuan hybrid conjugate gradient β_k^{hDY} subject to the cpu time metric, as in Figures 3 and 4.

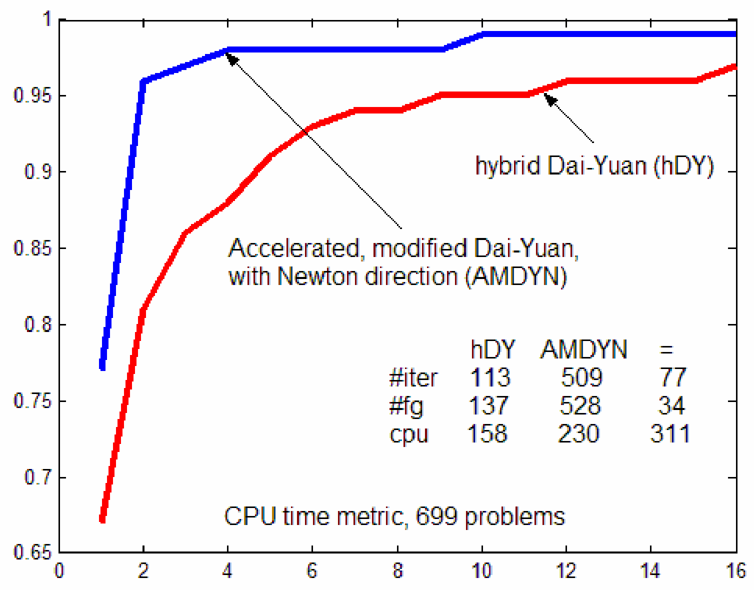

Fig. 3. Performance profile of AMDYN versus hDY.

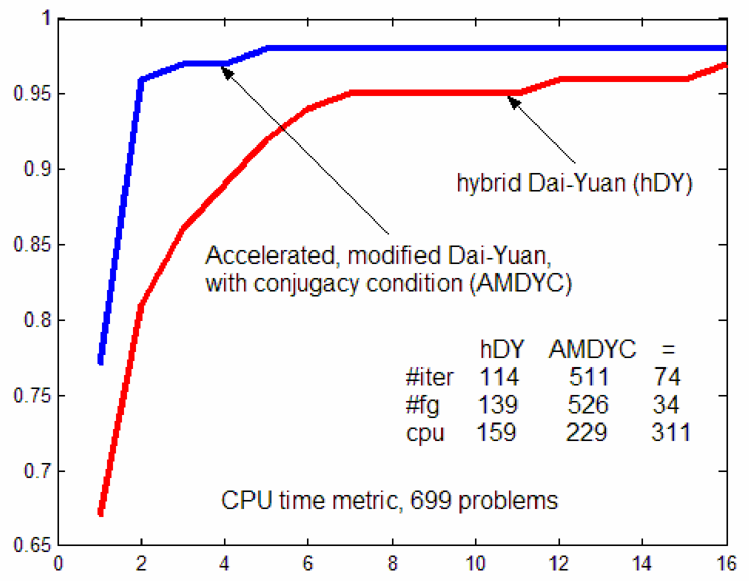

Fig. 4. Performance profile of AMDYC versus hDY.

In the fourth set of numerical experiments, in Figures 5 and 6, we compare the Dolan-Moré performance profile of AMDYN and AMDYC versus Dai-Yuan hybrid conjugate gradient β_k^{hDYz} subject to the cpu time metric.

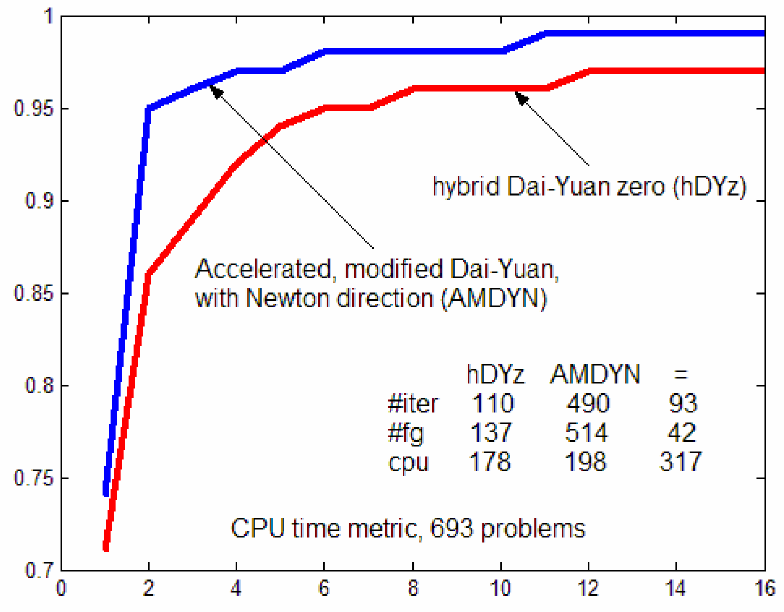

Fig. 5. Performance profile of AMDYN versus hDYz.

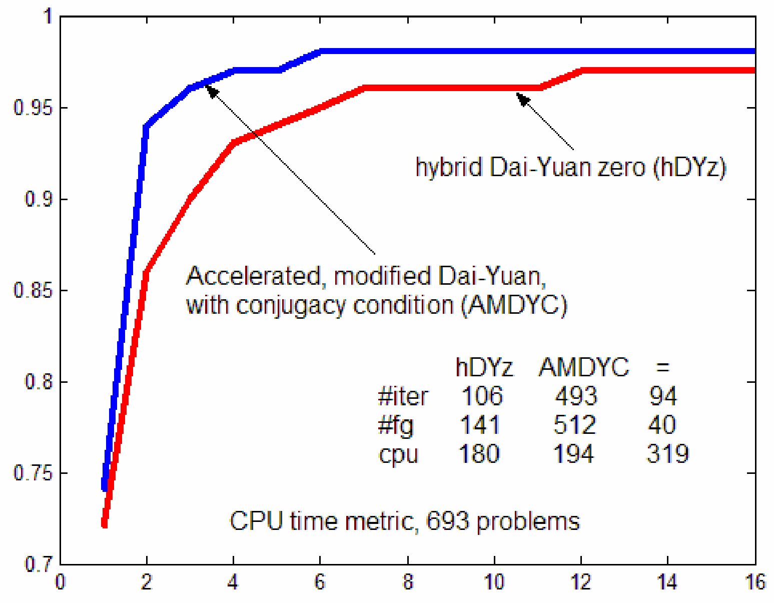

Fig. 6. Performance profile of AMDYC versus hDYz.

6. Conclusion

We have presented a new conjugate gradient algorithm for solving large-scale unconstrained optimization problems. The parameter β_k is a modification of the Dai and Yuan computational scheme in such a manner that the direction d_k generated by the algorithm satisfies the sufficient descent condition, independent of the line search. Under strong Wolfe line search conditions we proved the global convergence of the algorithm. We present computational evidence that the performance of our algorithms AMDYN and AMDYC was higher than that of the Dai and Yuan conjugate gradient algorithm and its hybrid variants, for a set consisting of 750 problems.

References

[1] Andrei, N., *Scaled conjugate gradient algorithms for unconstrained optimization*. Computational Optimization and Applications, 38 (2007), pp. 401-416.

[2] Andrei, N., *Scaled memoryless BFGS preconditioned conjugate gradient algorithm for unconstrained optimization.* Optimization Methods and Software 22 (2007), pp. 561-571.

[3] Andrei, N., *A scaled BFGS preconditioned conjugate gradient algorithm for unconstrained optimization*. Applied Mathematics Letters 20 (2007), pp. 645-650.

[4] Andrei, N., *A scaled nonlinear conjugate gradient algorithm for unconstrained optimization.* Optimization, 57 (2008), pp. 549-570.

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

[6] Andrei, N., *40 conjugate gradient algorithms for unconstrained optimization. A survey on their definition.* ICI Technical Report No. 13/08, March 14, 2008.

[7] Andrei, N., *Performance profiles of conjugate gradient algorithms for unconstrained optimization.* Encyclopedia of Optimization, 2nd edition, C.A. Floudas and P.M. Pardalos (Eds.), Springer, New York, vol. P (2009), 2938-2953.

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

[9] Andrei, N., *Accelerated conjugate gradient algorithm with finite difference Hessian / vector product approximation for unconstrained optimization.* Journal of Computational and Applied Mathematics,

[10] Birgin, E., Martínez, J.M., *A spectral conjugate gradient method for unconstrained optimization*, Applied Math. and Optimization, 43 (2001), pp.117-128.

[11] Bongartz, I., Conn, A.R., Gould, N.I.M., Toint, Ph.L., *CUTE: constrained and unconstrained testing environments*, ACM Trans. Math. Software, 21 (1995), pp.123-160.

[12] Dai, Y.H., *New properties of a nonlinear conjugate gradient method.* Numer. Math., 89 (2001), pp.83-98.

[13] Dai, Y.H., Liao, L.Z., *New conjugacy con7 ditions and related nonlinear conjugate gradient methods.* Appl. Math. Optim., 43 (2001), pp. 87-101.

[14] Dai, Y.H. Ni, Q., *Testing different conjugate gradient methods for large-scale unconstrained optimization,* J. Comput. Math., 21 (2003), pp.311-320.

[15] Dai, Y.H., Yuan, Y., *A nonlinear conjugate gradient method with a strong global convergence property,* SIAM J. Optim., 10 (1999), pp.177-182.

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

[17] Dai, Y.H., 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.

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

[19] Hager, W.W., Zhang, H., *A survey of nonlinear conjugate gradient methods*. Pacific journal of Optimization, 2 (2006), pp.35-58.

[20] Liu, D.C. and Nocedal, J., *On the limited memory BFGS method for large scale optimization.* Mathematical Programming, 45 (1989), pp.503-528.

[21] Nocedal, J., *Conjugate gradient methods and nonlinear optimization,* in L. Adams and J.L. Nazareth (Eds.) *Linear and Nonlinear Conjugate Gradient – Related Methods,* SIAM, Philadelphia, 1996, pp.9-23.

[22] Polak, E., Ribière, G., *Note sur la convergence de méthodes de directions conjuguée,* Revue Francaise Informat. Recherche Opérationnelle, 3e Année 16 (1969), pp.35-43.

[23] Polyak, B.T., *The conjugate gradient method in extreme problems,*USSR Comp. Math. Math. Phys., 9 (1969), pp.94-112.

[24] Powell, M.J.D., *Restart procedures for the conjugate gradient method.* Mathematical Programming 12 (1977), pp.241-254.

[25] Shanno, D.F., Phua, V., *Algorithm 500, Minimization of unconstrained multivariate functions*, ACM Trans. on Math. Soft., 2, pp.87-94, 1976.

[26] Wolfe, P., *Convergence conditions for ascent methods,* SIAM Rev., 11 (1969) pp.226-235.

[27] Wolfe, P., *Convergence conditions for ascent methods II: some corrections.* SIAM Rev., 13 (1971) pp.185-188.

[28] Zoutendijk, G., *Nonlinear programming computational methods*. In J. Abadie (Ed.) Integer and Nonlinear Programming, North-Holland, Amsterdam, 1970, pp.37-86.

February 3, 2009