New Accelerated Conjugate Gradient Algorithms for Unconstrained Optimization

Neculai Andrei
Research Institute for Informatics,
Center for Advanced Modeling and Optimization,
8-10, Averescu Avenue, Bucharest 1, Romania
and
Academy of Romanian Scientists,
54, Splaiul Independentei, Bucharest 5, 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, we suggest an acceleration scheme able to improve the efficiency of the algorithms. A global convergence result is proved when the strong Wolfe line search conditions are used. 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, 65K

Dedicated to the memory of Professor Gene H. Golub (1932-2007)

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 history of these algorithms has been given by Golub and O'Leary [28], as well as by O'Leary [38]. A survey of development of different versions of nonlinear conjugate gradient methods, with special attention to global convergence properties is presented by Hager and Zhang [31]. A survey on their definition including 40 conjugate gradient algorithms for unconstrained optimization is given by Andrei [7]. The conjugate gradient algorithms have been introduced early in 1952 by Cornelius Lanczos [34, 35] and by Magnus Hestenes with the cooperation of J.B. Rosser, G. E. Forsythe, L. Paige, M. Stein, R. Hayes and U. Hochstrasser at the Institute for Numerical Analysis, a part of National Bureau of Standards in Los Angeles, and by Eduard Stiefel at Eidgenössischen Technischen Hochschule in Zürich (see Hestenes and Stiefel [32]). Even if the conjugate gradient methods are now over 50 years old, they continue to be of a considerable interest particularly due to their convergence properties and to a very easy implementation in computer programs of the corresponding algorithms.

Many researchers in this area brought significant contributions, clarifying many theoretical and computational aspects of this class of algorithms. Particularly, for the development of effective algorithms for solving basic linear algebra problems Golub and Kahan [27] discussed the use of the Lanczos algorithm in computing the singular value decomposition of non-quadratic functions. Concus, Golub and O'Leary [15] considered preconditioning of conjugate gradient algorithms. An important extension of the conjugate

gradient algorithm was given by Concus and Golub [16] where the Hermitian part of the matrix is taken as a preconditioner. A development of a block form of the Lanczos algorithm has been proposed by Golub, Underwood and Wilkinson [29]. Fletcher and Reeves extended the conjugate gradient algorithm to minimization of non-quadratic functions [25], thus opening an important area of research and applications. Developments of this algorithm have been considered *inter allia* by Polak and Ribière [41], Polyak [42], Perry [40], Shanno [44, 45], Gilbert and Nocedal [26], Liu and Storey [37], Hu and Storey [33],Touati-Ahmed and Storey [46]. Recently, the papers of Nocedal [39], Dai and Yuan [20, 21], Hager and Zhang [30], Andrei [1-6, 8-11] present important theoretical and computational contributions to the development of this class of algorithms.

In this paper we suggest new nonlinear conjugate gradient algorithms which are mainly modifications of the Dai and Yuan [20] 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} 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, while in section 3 we prove the global convergence of the algorithms under strong Wolfe line search conditions. In Section 4 we present an acceleration of the algorithm while in Section 5 we compare the computational performance of the new conjugate gradient schemes against the Dai and Yuan method and its hybrid variants [21] using 750 unconstrained optimization test problems from the CUTE [14] library along with some other large-scale unconstrained optimization problems presented in [12]. Using the Dolan and Moré performance profiles [23] we prove that 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

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

where $f : R^n \to R$ is continuously differentiable 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 [20]. 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 conserve the sufficient descent condition and to have 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 [17] 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 the 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 $u^T v \le \frac{1}{2}(\|u\|^2 + \|v\|^2)$ 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 that $\theta_{k+1} - 1/4 > 0$, then the direction given by (2.3) and (2.4) is a descent direction. Dai and Yuan [20, 21] 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, $\theta_{k+1} \geq 1/4,$ 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 $k \geq 1$ 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 [18] 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:

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

Again, 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 $0 < \rho \le \sigma < 1$.

In [21] 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 $s_k^T g_{k+1} \ge \sigma s_k^T g_k = -\sigma y_k^T s_k + \sigma s_k^T g_{k+1}$, i.e.

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

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

Therefore, $\beta_k^N \le \beta_k^{DY} \frac{1}{1-\sigma}$.

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 $g_k \ne 0$ for all $k \ge 1$, otherwise a stationary point is obtained. Assume that:

(i) *The level set* $S = \{x \in R^n : f(x) \le f(x_0)\}$ *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* $\|\nabla f(x) - \nabla f(y)\| \le L\|x - y\|$, *for all* $x, y \in N$.

Under these assumptions on f there exists a constant $\Gamma \ge 0$ such that $\|\nabla f(x)\| \le \Gamma$ 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,

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

$$\left|g_{k+1}^T d_k\right| \le \sigma g_k^T d_k. \tag{3.2}$$

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

Dai *et al.* [22] proved that for any conjugate gradient method with strong Wolfe line search the following general result holds:

Lemma 3.1. *Suppose that the assumptions (i) and (ii) hold and consider any conjugate gradient method (2.2) and (2.3), where* d_k *is a descent direction and* α_k *is obtained by the strong Wolfe line search (3.1) and (3.2). If*

$$\sum_{k \ge 1} \frac{1}{\|d_k\|^2} = \infty, \tag{3.3}$$

then

$$\liminf_{k\to\infty} \|g_k\| = 0. \ \blacksquare \tag{3.4}$$

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* $\overline{\gamma} \geq 0$ *such* $\overline{\gamma} \leq \|\nabla f(x)\|$, $1/4 \leq \theta_k \leq \tau$, *where* τ *is a positive constant and the angle* φ_k *between* g_k *and* d_k *is bounded, i.e.* $\cos\varphi_k \leq \xi$ *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

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

where

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

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 Lemma 3.1 it follows that $\liminf_{k\to\infty} \|g_k\| = 0$. $\blacksquare$

4. Acceleration of the algorithm

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. In order to improve the performances of the conjugate gradient algorithms the efforts were directed to design procedures for direction computation based on the second order information. For example, CONMIN [43], and SCALCG [2-5] take this idea of BFGS preconditioning. In this section we focus on the step length modification. In the context of gradient descent algorithm with backtracking the step length modification has been considered for the first time in [6]. Nocedal [39] 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. Numerical comparisons between conjugate gradient methods and the limited memory quasi Newton method, by Liu and Nocedal [36], show that the latter is more successful [8]. One explanation of the efficiency of the 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 present an acceleration scheme. Basically this modifies the step length in a multiplicative manner to improve the reduction of the function values along the iterations.

Line search. For using the algorithm (2.2) one of the crucial elements is the stepsize computation. For sake of generality in the following we consider the line searches that satisfy either the Goldstein's conditions [24]:

$$\rho_1\alpha_k g_k^T d_k \le f(x_k+\alpha_k d_k)-f(x_k)\le \rho_2\alpha_k g_k^T d_k, \tag{4.1}$$

where $0<\rho_2<\frac{1}{2}<\rho_1<1$ and $\alpha_k>0,$ or the Wolfe conditions (2.12) and (2.13).

Proposition 4.1. *Assume that d_k is a descent direction and ∇f satisfies the Lipschitz condition $\|\nabla f(x)-\nabla f(x_k)\|\le L\|x-x_k\|$ for all x on the line segment connecting x_k and $x_{k+1},$ where L is a positive constant. If the line search satisfies the Goldstein conditions (4.1), then*

$$\alpha_k \ge \frac{(1-\rho_1)}{L}\frac{\left|g_k^T d_k\right|}{\|d_k\|^2}. \tag{4.2}$$

If the line search satisfies the Wolfe conditions (2.12) and (2.13), then

$$\alpha_k \ge \frac{(1-\sigma)}{L}\frac{\left|g_k^T d_k\right|}{\|d_k\|^2}. \tag{4.3}$$

Proof. If the Goldstein conditions are satisfied, then using the mean value theorem from (4.1) we get:

$$\begin{aligned}\rho_1\alpha_k g_k^T d_k &\le f(x_k+\alpha_k d_k)-f(x_k)\\ &=\alpha_k\nabla f(x_k+\xi d_k)^T d_k \le \alpha_k g_k^T d_k + L\alpha_k^2\|d_k\|^2,\end{aligned}$$

where $\xi\in[0,\alpha_k].$ From this inequality we immediately get (4.2).

Now, to prove (4.3) subtract $g_k^T d_k$ from both sides of (2.13) and using the Lipschitz condition we get:

$$(\sigma-1)g_k^T d_k \le (g_{k+1}-g_k)^T d_k \le \alpha_k L\|d_k\|^2. \tag{4.4}$$

But, d_k is a descent direction and since $\sigma<1$, we immediately get (4.3). ■

Therefore, satisfying the Goldstein or the Wolfe line search conditions α is bounded away from zero, i.e. there exists a positive constant ω, such that $\alpha\ge\omega.$

Acceleration scheme [6]. Given the initial point x_0 we can compute $f_0=f(x_0),$ $g_0=\nabla f(x_0)$ and by Wolfe line search conditions (2.12) and (2.13) the steplength α_0 is determined. With these, the next iteration is computed as: $x_1=x_0+\alpha_0 d_0$, ($d_0=-g_0$) where f_1 and g_1 are immediately determined and the direction d_1 can be computed as: $d_1=-\theta_1 g_1+\beta_0^N s_0$, where the conjugate gradient parameter β_0^N is computed as in (2.4) and the scaling factor θ_1 is computed as in (2.10) or (2.11). Therefore, at the iteration $k=1,2,...$ we know x_k, f_k, g_k and $d_k=-\theta_k g_k+\beta_{k-1}^N s_{k-1}.$ Suppose that d_k is a descent direction (i.e. $\theta_k\ge 1/4$). By the Wolfe line search (2.12) and (2.13) we can compute α_k with which the following point $z=x_k+\alpha_k d_k$ is determined. The first Wolfe condition (2.12) shows that the steplength $\alpha_k>0$ satisfies:

$$f(z)=f(x_k+\alpha_k d_k)\le f(x_k)+\rho\alpha_k g_k^T d_k.$$

With these, let us introduce the accelerated conjugate gradient algorithm by means of the following iterative scheme:

$$x_{k+1} = x_k + \gamma_k \alpha_k d_k, \tag{4.5}$$

where $\gamma_k > 0$ is a parameter which follows to be determined in such a manner as to improve the behavior of the algorithm. Now, we have:

$$f(x_k + \alpha_k d_k) = f(x_k) + \alpha_k g_k^T d_k + \frac{1}{2}\alpha_k^2 d_k^T \nabla^2 f(x_k) d_k + o\left(\left\|\alpha_k d_k\right\|^2\right). \tag{4.6}$$

On the other hand, for $\gamma > 0$ we have:

$$f(x_k + \gamma\alpha_k d_k) = f(x_k) + \gamma\alpha_k g_k^T d_k + \frac{1}{2}\gamma^2\alpha_k^2 d_k^T \nabla^2 f(x_k) d_k + o\left(\left\|\gamma\alpha_k d_k\right\|^2\right). \tag{4.7}$$

With these we can write:

$$f(x_k + \gamma\alpha_k d_k) = f(x_k + \alpha_k d_k) + \Psi_k(\gamma), \tag{4.8}$$

where

$$\begin{aligned}\Psi_k(\gamma) = {} & \frac{1}{2}(\gamma^2 - 1)\alpha_k^2 d_k^T \nabla^2 f(x_k) d_k + (\gamma - 1)\alpha_k g_k^T d_k \\ & + \gamma^2 \alpha_k o\left(\alpha_k \left\|d_k\right\|^2\right) - \alpha_k o\left(\alpha_k \left\|d_k\right\|^2\right).\end{aligned} \tag{4.9}$$

Let us denote:

$$\begin{aligned} a_k &= \alpha_k g_k^T d_k \le 0, \\ b_k &= \alpha_k^2 d_k^T \nabla^2 f(x_k) d_k, \\ \varepsilon_k &= o\left(\alpha_k \left\|d_k\right\|^2\right). \end{aligned}$$

Observe that $a_k \le 0$, since d_k is a descent direction, and for convex functions $b_k \ge 0$. Therefore,

$$\Psi_k(\gamma) = \frac{1}{2}(\gamma^2 - 1)b_k + (\gamma - 1)a_k + \gamma^2\alpha_k\varepsilon_k - \alpha_k\varepsilon_k. \tag{4.10}$$

Now, we see that $\Psi_k'(\gamma) = (b_k + 2\alpha_k\varepsilon_k)\gamma + a_k$ and $\Psi_k'(\gamma_m) = 0$ where

$$\gamma_m = -\frac{a_k}{b_k + 2\alpha_k\varepsilon_k}. \tag{4.11}$$

Observe that $\Psi_k'(0) = a_k \le 0$. Therefore, assuming that $b_k + 2\alpha_k\varepsilon_k > 0$, then $\Psi_k(\gamma)$ is a convex quadratic function with minimum value in point γ_m and

$$\Psi_k(\gamma_m) = -\frac{(a_k + (b_k + 2\alpha_k\varepsilon_k))^2}{2(b_k + 2\alpha_k\varepsilon_k)} \le 0.$$

Considering $\gamma = \gamma_m$ in (4.8) and since $b_k \ge 0$, we see that for every k

$$f(x_k + \gamma_m\alpha_k d_k) = f(x_k + \alpha_k d_k) - \frac{(a_k + (b_k + 2\alpha_k\varepsilon_k))^2}{2(b_k + 2\alpha_k\varepsilon_k)} \le f(x_k + \alpha_k d_k),$$

which is a possible improvement of the values of function f (when $a_k + (b_k + 2\alpha_k\varepsilon_k) \ne 0$). Therefore, using this simple multiplicative modification of the stepsize α_k as $\gamma_k\alpha_k$ where $\gamma_k = \gamma_m = -a_k/(b_k + 2\alpha_k\varepsilon_k)$ we get:

$$\begin{aligned} f(x_{k+1}) = f(x_k + \gamma_k\alpha_k d_k) &\le f(x_k) + \rho\alpha_k g_k^T d_k - \frac{(a_k + (b_k + 2\alpha_k\varepsilon_k))^2}{2(b_k + 2\alpha_k\varepsilon_k)} \\ &= f(x_k) - \left[\frac{(a_k + (b_k + 2\alpha_k\varepsilon_k))^2}{2(b_k + 2\alpha_k\varepsilon_k)} - \rho a_k\right] \le f(x_k), \end{aligned} \tag{4.12}$$

since $a_k \le 0$, (d_k is a descent direction).

Observe that if d_k is a descent direction, then

$$\frac{(a_k + (b_k + 2\alpha_k \varepsilon_k))^2}{2(b_k + 2\alpha_k \varepsilon_k)} > \frac{(a_k + b_k)^2}{2b_k}$$

and from (4.12) we get:

$$f(x_{k+1}) \le f(x_k) - \left[\frac{(a_k + (b_k + 2\alpha_k \varepsilon_k))^2}{2(b_k + 2\alpha_k \varepsilon_k)} - \rho a_k \right]$$

$$< f(x_k) - \left[\frac{(a_k + b_k)^2}{2b_k} - \rho a_k \right] \le f(x_k).$$

Therefore, neglecting the contribution of ε_k, we still get an improvement on the function values.

Now, in order to get the algorithm we have to determine a way for b_k computation. For this, at point $z = x_k + \alpha_k d_k$ we have:

$$f(z) = f(x_k + \alpha_k d_k) = f(x_k) + \alpha_k g_k^T d_k + \frac{1}{2}\alpha_k^2 d_k^T \nabla^2 f(\tilde{x}_k) d_k,$$

where $\tilde{x}_k$ is a point on the line segment connecting x_k and z. On the other hand, at point $x_k = z - \alpha_k d_k$ we have:

$$f(x_k) = f(z - \alpha_k d_k) = f(z) - \alpha_k g_z^T d_k + \frac{1}{2}\alpha_k^2 d_k^T \nabla^2 f(\bar{x}_k) d_k,$$

where $g_z = \nabla f(z)$ and $\bar{x}_k$ is a point on the line segment connecting x_k and z. Having in view the local character of searching and that the distance between x_k and z is small enough, we can consider $\tilde{x}_k = \bar{x}_k = x_k$. So, adding the above equalities we get:

$$b_k = -\alpha_k y_k^T d_k, \tag{4.13}$$

where $y_k = g_k - g_z$.

Observe that if $|a_k| > b_k$, then $\gamma_k > 1$. In this case $\gamma_k \alpha_k > \alpha_k$ and it is also possible that $\gamma_k \alpha_k \le 1$ or $\gamma_k \alpha_k > 1$. Hence, the steplength $\gamma_k \alpha_k$ can be greater than 1. On the other hand, if $|a_k| \le b_k$, then $\gamma_k \le 1$. In this case $\gamma_k \alpha_k \le \alpha_k$, so the steplength $\gamma_k \alpha_k$ is reduced. Therefore, if $|a_k| \ne b_k$, then $\gamma_k \ne 1$ and the steplength α_k computed by Wolfe conditions will be modified by its increasing or its reducing through factor γ_k.

Neglecting ε_k in (4.10), we see that $\Psi_k(1) = 0$ and if $|a_k| \le b_k / 2$, then $\Psi_k(0) = -a_k - b_k / 2 \le 0$ and $\gamma_k < 1$. Therefore, for any $\gamma \in [0,1]$, $\Psi_k(\gamma) \le 0$. As a consequence for any $\gamma \in (0,1)$, it follows that $f(x_k + \gamma \alpha_k d_k) < f(x_k)$. In this case, for any $\gamma \in [0,1]$, $\gamma_k \alpha_k \le \alpha_k$. However, in our algorithm we selected $\gamma_k = \gamma_m$ as the point achieving the minimum value of $\Psi_k(\gamma)$.

5. AMDYN and AMDYC Algorithms

Considering the definitions of g_k, s_k and y_k 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. 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

$$g_{k+1}^T d \leq -10^{-3} \|d\|_2 \|g_{k+1}\|_2, \tag{5.1}$$

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 $-g_{k+1}$. More sophisticated reasons for restarting the algorithms have been proposed in the literature, 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 (5.1) 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 [43]. It is also considered in the packages: SCG by Birgin and Martínez [13] and in SCALCG by Andrei [2-5, 11].

Proposition 5.1. *Suppose that f is a uniformly convex function on the level set $S = \{x : f(x) \leq f(x_0)\}$, and d_k satisfies the sufficient descent condition $g_k^T d_k < -c_1 \|g_k\|^2$, where $c_1 > 0$, and $\|d_k\|^2 \leq c_2 \|g_k\|^2$, where $c_2 > 0$. Then the sequence generated by AMDYN or AMDYC converges linearly to x^*, solution to the problem (2.1).*

Proof. From (4.12) we have that $f(x_{k+1}) \leq f(x_k)$ for all k. Since f is bounded below, it follows that

$$\lim_{k \to \infty} (f(x_k) - f(x_{k+1})) = 0.$$

Now, since f is uniformly convex there exist positive constants m and M, such that $mI \le \nabla^2 f(x) \le MI$ on S. Suppose that $x_k + \alpha d_k \in S$ and $x_k + \gamma_m \alpha d_k \in S$ for all $\alpha > 0$. We have:

$$f(x_k + \gamma_m \alpha d_k) \le f(x_k + \alpha d_k) - \frac{(a_k + b_k)^2}{2b_k}.$$

But, from uniform convexity we have the following quadratic upper bound on $f(x_k + \alpha d_k)$:

$$f(x_k + \alpha d_k) \le f(x_k) + \alpha g_k^T d_k + \frac{1}{2} M \alpha^2 \|d_k\|^2.$$

Therefore,

$$\begin{aligned} f(x_k + \alpha d_k) &\le f(x_k) - \alpha c_1 \|g_k\|^2 + \frac{1}{2} M c_2 \alpha^2 \|g_k\|^2 \\ &= f(x_k) + \left[-c_1 \alpha + \frac{1}{2} M c_2 \alpha^2 \right] \|g_k\|^2. \end{aligned}$$

Observe that for $0 \le \alpha \le c_1/(Mc_2)$, $-c_1\alpha + \frac{1}{2} M c_2 \alpha^2 \le -\frac{c_1}{2}\alpha$ which follows from the convexity of $-c_1\alpha + (Mc_2/2)\alpha^2$. Using this result we get:

$$f(x_k + \alpha d_k) \le f(x_k) - \frac{1}{2} c_1 \alpha \|g_k\|^2 \le f(x_k) - \rho c_1 \alpha \|g_k\|^2,$$

since $\rho < 1/2$.

From proposition 4.1 the Wolfe line search terminates with a value $\alpha \ge \omega > 0$. Therefore, for $0 \le \alpha \le c_1/(Mc_2)$, this provides a lower bound on the decrease in the function f, i.e.

$$f(x_k + \alpha d_k) \le f(x_k) - \rho c_1 \omega \|g_k\|^2. \tag{5.2}$$

On the other hand,

$$\frac{(a_k + b_k)^2}{2b_k} \ge \frac{(\alpha^2 M c_2 - \alpha c_1)^2 \|g_k\|^4}{2\alpha^2 M c_2 \|g_k\|^2} \ge \frac{(\omega M c_2 - c_1)^2}{2Mc_2} \|g_k\|^2. \tag{5.3}$$

Considering (5.2) and (5.3) we get:

$$f(x_k + \gamma_m \alpha d_k) \le f(x_k) - \rho c_1 \omega \|g_k\|^2 - \frac{(\omega M c_2 - c_1)^2}{2Mc_2} \|g_k\|^2. \tag{5.4}$$

Therefore,

$$f(x_k) - f(x_k + \gamma_m \alpha d_k) \ge \left[\rho c_1 \omega + \frac{(\omega M c_2 - c_1)^2}{2Mc_2} \right] \|g_k\|^2.$$

But, $f(x_k) - f(x_{k+1}) \to 0$ and as a consequence g_k goes to zero, i.e. x_k converges to x^*. Having in view that $f(x_k)$ is a nonincreasing sequence, it follows that $f(x_k)$ converges to $f(x^*)$. From (5.4) we see that

$$f(x_{k+1}) \le f(x_k) - \left[\rho c_1 \omega + \frac{(\omega M c_2 - c_1)^2}{2Mc_2} \right] \|g_k\|^2. \tag{5.5}$$

Combining this with $\|g_k\|^2 \ge 2m(f(x_k) - f^*)$ and subtracting f^* from both sides of (5.5) we conclude: $f(x_{k+1}) - f^* \le c(f(x_k) - f^*)$, where

$$c = 1 - 2m\left[\rho c_1 \omega + \frac{(\omega M c_2 - c_1)^2}{2Mc_2}\right] < 1.$$

Therefore, $f(x_k)$ converges to f^* at least as fast as a geometric series with a factor that depends on the parameter ρ in the first Wolfe condition and the bounds m and M of the Hessian. So, the convergence of the acceleration scheme is at least linear. ■

6. 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. The test problems are the unconstrained problems in the CUTE [14] library, along with other large-scale optimization problems presented in [12]. We selected 75 large-scale unconstrained optimization problems in extended or generalized form. 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 $\|g_k\|_\infty \leq 10^{-6}$, 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

$$\left| f_i^{ALG1} - f_i^{ALG2} \right| < 10^{-3} \tag{6.1}$$

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 (6.1) hold.

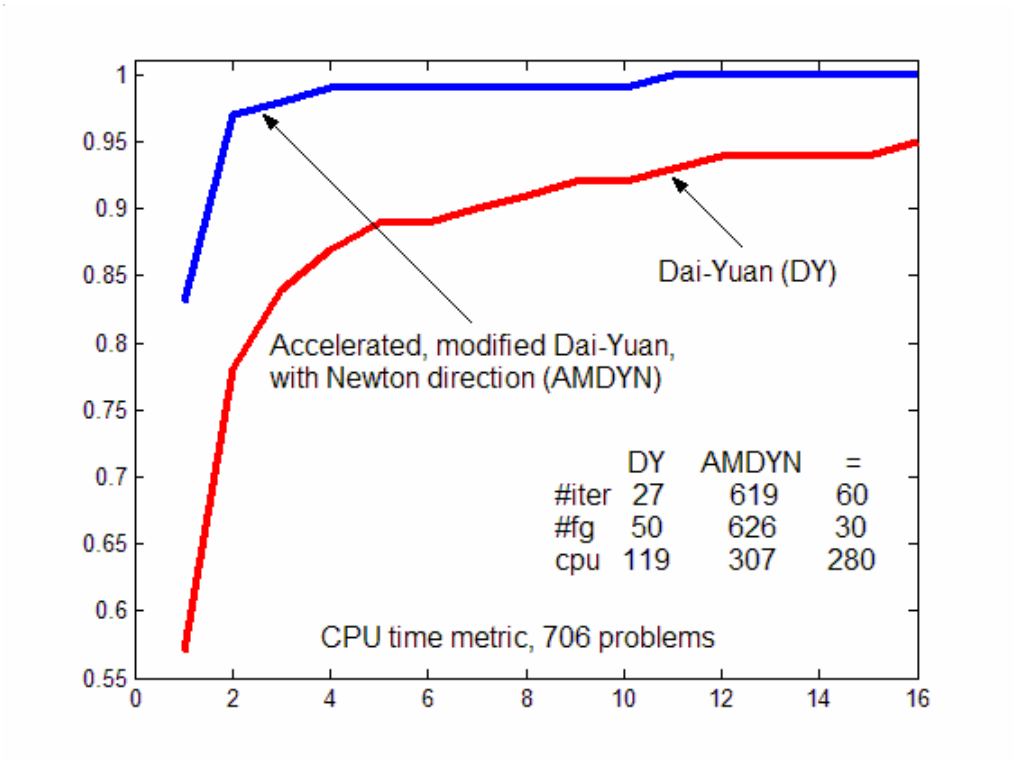

Fig. 1. Performance profile of AMDYN versus DY.

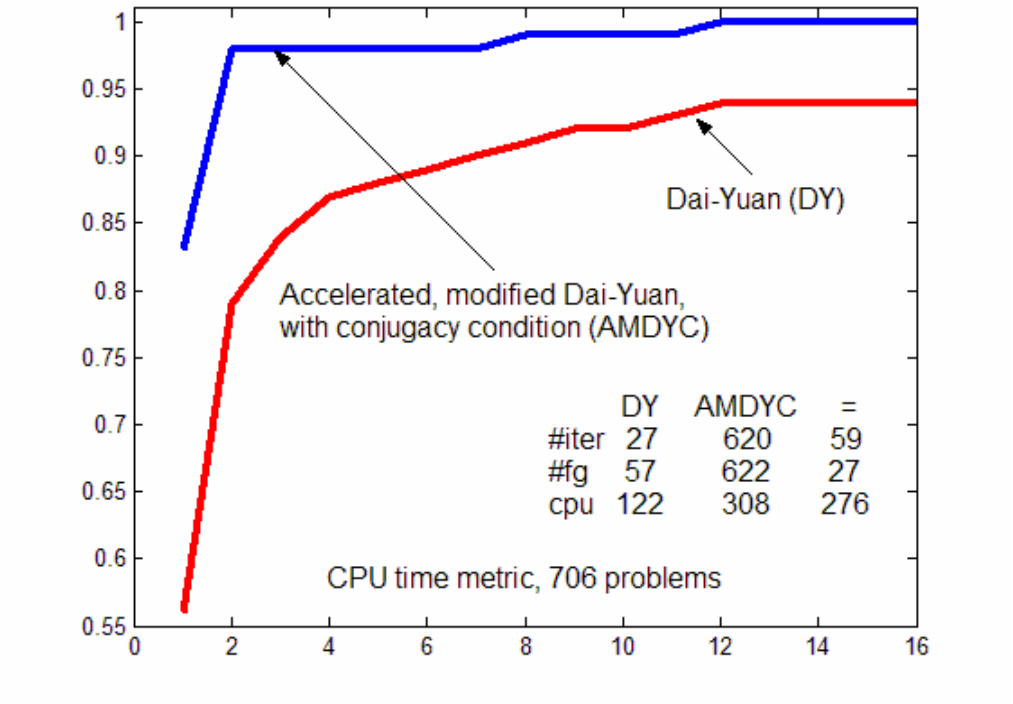

Fig. 2. Performance profile of AMDYC versus DY.

Dai and Yuan [21] studied the hybrid conjugate gradient algorithms and proposed the following two hybrid methods:

$$\beta_k^{hDY} = \max\left\{-\frac{1-\sigma}{1+\sigma}\beta_k^{DY}, \min\left\{\beta_k^{HS}, \beta_k^{DY}\right\}\right\}, \tag{6.2}$$

$$\beta_k^{hDYz} = \max\left\{0, \min\left\{\beta_k^{HS}, \beta_k^{DY}\right\}\right\}, \tag{6.3}$$

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 [19] proved that the second hybrid method (hDYz) is the better, outperforming the Polak-Ribière [41] and Polyak [42] 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.

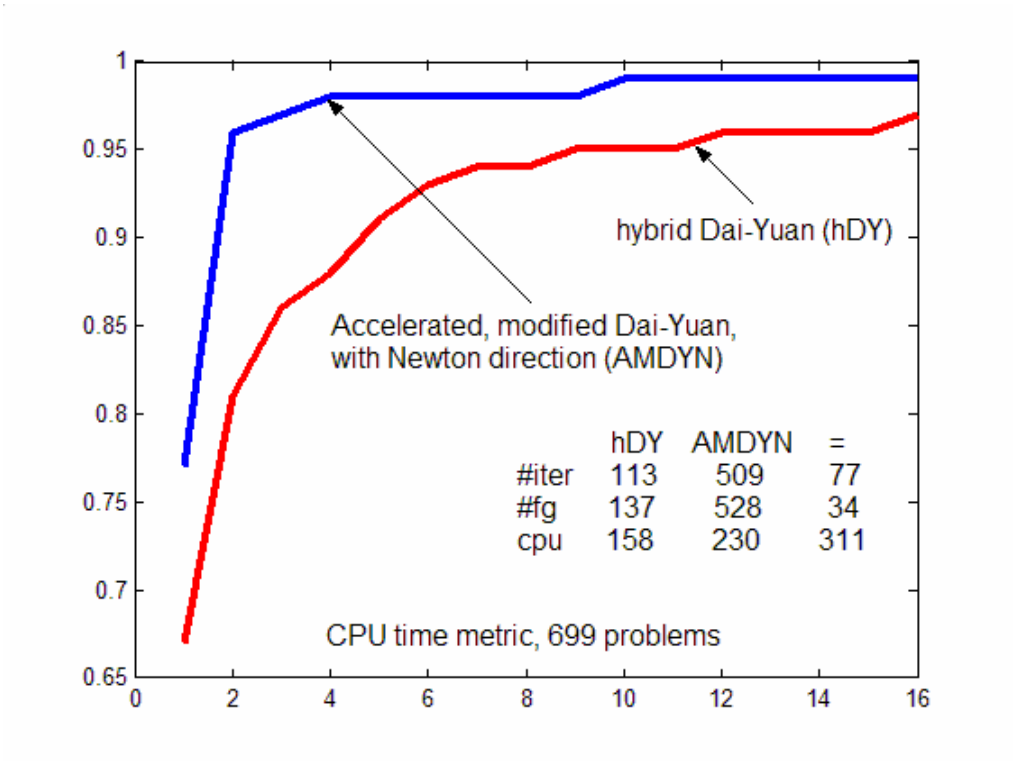

Fig. 3. Performance profile of AMDYN versus hDY.

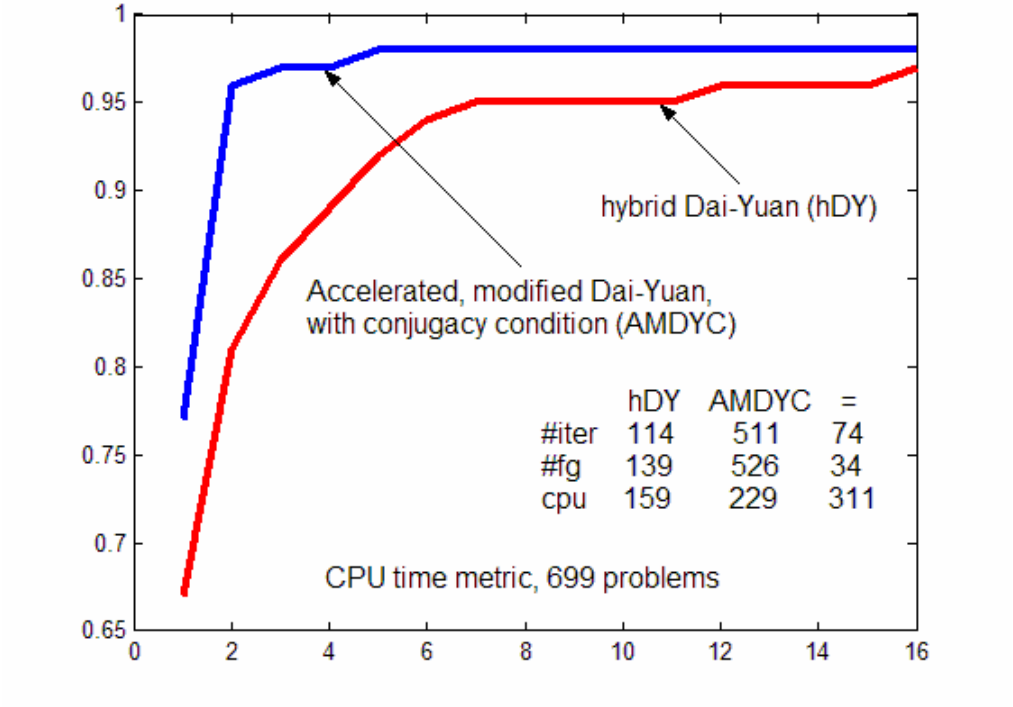

Fig. 4. Performance profile of AMDYC versus hDY.

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.

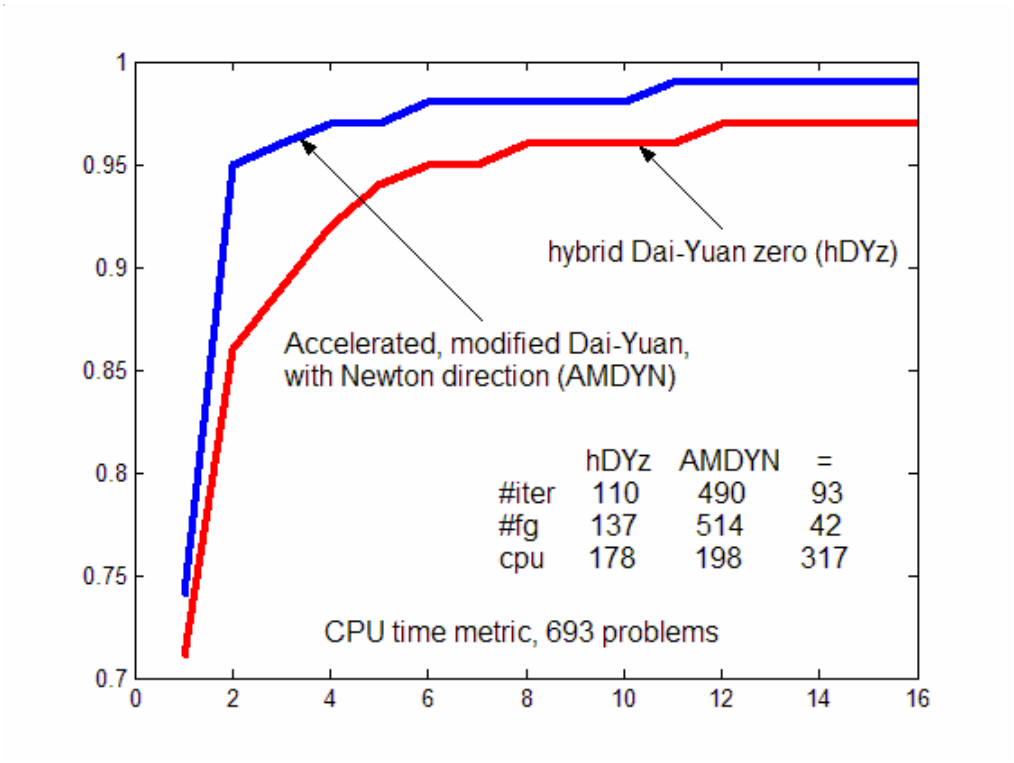

Fig. 5. Performance profile of AMDYN versus hDYz.

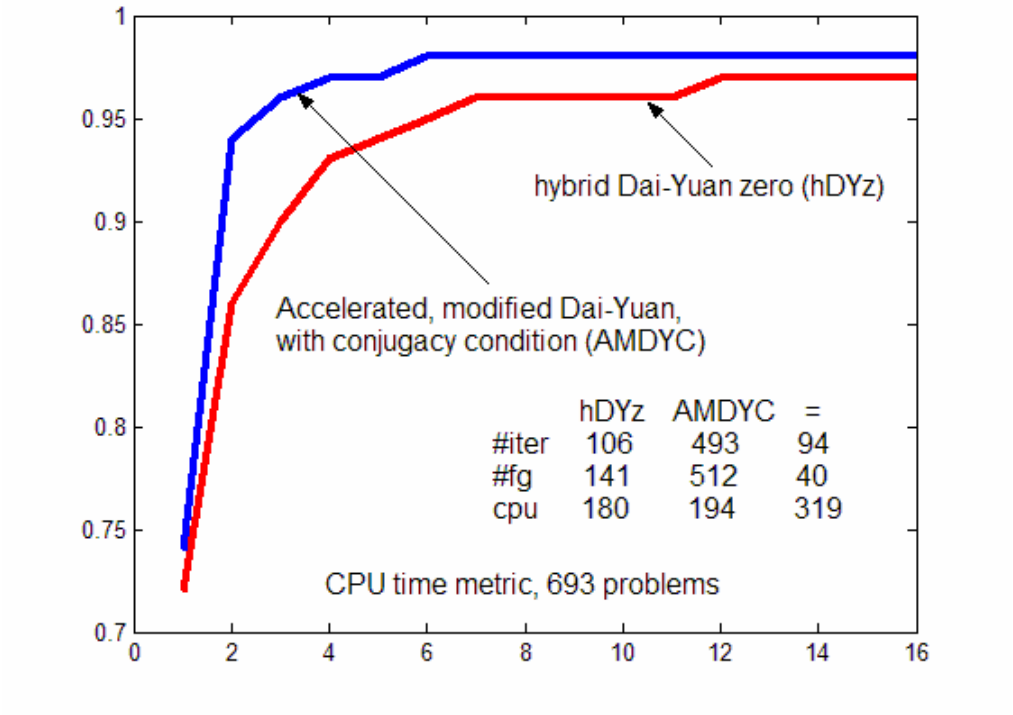

Fig. 6. Performance profile of AMDYC versus hDYz.

7. 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.: *Conjugate gradient algorithms for large scale unconstrained optimization.* ICI Technical Report, January 12, 2005.

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

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

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

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

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

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

[8] Andrei, N.: *Numerical comparison of conjugate gradient algorithms for unconstrained optimization.* Studies in Informatics and Control, 16 (2007), pp.333-352.

[9] Andrei, N.: *A Dai-Yuan conjugate gradient algorithm with sufficient descent and conjugacy conditions for unconstrained optimization.* Applied Mathematics Letters 21 (2008), pp. 165-171.

[10] Andrei, N.: *Another hybrid conjugate gradient algorithm for unconstrained optimization.* Numerical Algorithms, 47 (2008), pp.143-156.

[11] Andrei, N.: *Performance profiles of conjugate gradient algorithms for unconstrained optimization.* Encyclopedia of Optimization, 2nd edition, 2008, C.A. Floudas and P.M. Pardalos (Eds.), entry 762, in press.

[12] Andrei, N.: *An unconstrained optimization test functions collection*. Advanced Modeling and Optimization – An electronic International Journal, 10 (2008), pp.147-161

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

[14] 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.

[15] Concus, P., Golub, G.H., O'Leary, D.P.: *A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations,* in J.R. Bunch and D.J. Rose (Eds.) *Sparse Matrix Computations*, Academic Press, New York, 1976, pp.309-332.

[16] Concus, P., Golub, G.H.: *A generalized conjugate gradient method for nonsymmetric systems of linear equations,* in R. Glowinski and J.L. Lions (Eds.) *Computing Methods in Applied Sciences and Engineering* (2nd International Symposium, Versailles 1975), Part I, Springer Lecture Notes in Economics and Mathematical Systems 134, New York, 1976, pp.56-65.

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

[18] Dai, Y.H., Liao, L.Z.: *New conjugacy con7 ditions and related nonlinear conjugate gradient methods.* Appl. Math. Optim., 43 (2001), pp. 87-101.
[19] Dai, Y.H. Ni, Q.: *Testing different conjugate gradient methods for large-scale unconstrained optimization,* J. Comput. Math., 21 (2003), pp.311-320.
[20] Dai, Y.H., Yuan, Y.: *A nonlinear conjugate gradient method with a strong global convergence property*, SIAM J. Optim., 10 (1999), pp.177-182.
[21] Dai, Y.H. Yuan, Y.: *An efficient hybrid conjugate gradient method for unconstrained optimization.* Annals of Operations Research, 103 (2001), pp.33-47.
[22] 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.
[23] Dolan, E.D., Moré, J.J.: *Benchmarking optimization software with performance profiles*, Math. Programming, 91 (2002), pp. 201-213.
[24] Goldstein, A.A.: *On steepest descent,* SIAM J. Control, Vol. 3, pp.147-151, 1965.
[25] Fletcher, R., Reeves, C.M.: *Function minimization by conjugate gradients.* Computer Journal, 7 (1964), pp.149-154.
[26] Gilbert, J.C. Nocedal, J.: *Global convergence properties of conjugate gradient methods for optimization.* SIAM J. Optim., 2 (1992), pp.21-42.
[27] Golub, G.H., Kahan, W.: *Calculating the singular values and pseudo-inverse of a matrix*. SIAM J. Numer. Anal., 2 (Series B) (1965), pp. 205-224.
[28] Golub, G.H. O'Leary, D.P.: *Some history of the conjugate gradient and Lanczos algorithms: 1948-1976*. SIAM Review, 31 (1976), pp.50-100.
[29] Golub, G.H., Underwood, R., Wilkinson, J.H.: *The Lanczos algorithm for the symmetric* $Ax = \lambda Bx$ *problem.* Technical Report, Stanford University Computer Science Department, Stanford, California, 1972.
[30] Hager, W.W., Zhang, H.: *A new conjugate gradient method with guaranteed descent and an efficient line search*, SIAM Journal on Optimization, 16 (2005), 170-192.
[31] Hager, W.W., Zhang, H.: *A survey of nonlinear conjugate gradient methods*. Pacific journal of Optimization, 2 (2006), pp.35-58.
[32] Hestenes, M.R., Stiefel, E.: *Methods of conjugate gradients for solving linear systems*, J. Research Nat. Bur. Standards Sec. B. 48, pp. 409-436, 1952.
[33] Hu, Y.F., Storey, C.: *Global convergence result for conjugate gradient methods*. JOTA, 71, (1991), pp.399-405.
[34] Lanczos, C.: *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators,* J. Res. Nat. Bur. Standards, 45 (1950), pp.255-282.
[35] Lanczos, C.: *Solution of systems of linear equations by minimized iterations*, J. Res. Nat. Bur. Standards, 49 (1952), pp.33-53.
[36] Liu, D.C., Nocedal, J.: *On the limited memory BFGS method for large scale optimization methods.* Mathematical Programming, 45 (1989), pp. 503-528.
[37] Liu, Y., Storey, C.: *Efficient generalized conjugate gradient algorithms, Part 1: Theory.* JOTA, 69 (1991), pp.129-137.
[38] O'Leary, D.P.: *Conjugate gradients and related KMP algorithms: The beginnings*. In L. Adams and J.L. Nazareth (Eds.) *Linear and Nonlinear Conjugate Gradient – Related Methods.* SIAM, Philadelphia, 1996, pp.1-8.
[39] 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.
[40] Perry, J.M.: *A class of conjugate gradient algorithms with a two-step variable-metric memory,* Discussion Paper 269, Center for Mathematical Studies in Economic and Management Sciences, Northwestern University, Evanston, Illinois, 1977.
[41] 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.
[42] Polyak, B.T.: *The conjugate gradient method in extreme problems,*USSR Comp. Math. Math. Phys., 9 (1969), pp.94-112.

[43] Shanno, D.F., Phua, V: *Algorithm 500, Minimization of unconstrained multivariate functions*, ACM Trans. on Math. Soft., 2, pp.87-94, 1976.
[44] Shanno, D.F.: *On the convergence of a new conjugate gradient algorithm,*SIAM J. Numer. Anal., 15 (1978), pp.1247-1257.
[45] Shanno, D.F.: *Conjugate gradient methods with inexact searches.* Math. Oper. Res., 3 (1978), pp.244-256.
[46] Touati-Ahmed, D., Storey, C.: *Efficient hybrid conjugate gradient techniques,* Journal Of Optimization Theory and Applications, 64 (1990), pp. 379-397.

June 18, 2008