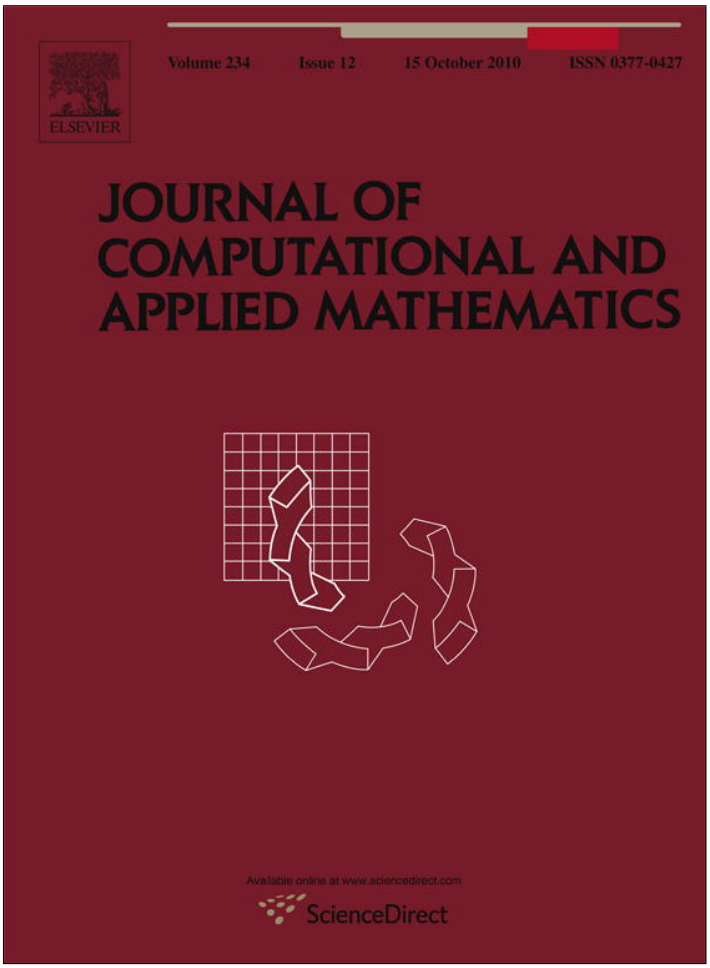

Journal of Computational and Applied Mathematics 234 (2010) 3397–3410

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

New accelerated conjugate gradient algorithms as a modification of Dai–Yuan's computational scheme for unconstrained optimization

Neculai Andrei [1]

Research Institute for Informatics, Center for Advanced Modeling and Optimization, 8-10, Averescu Avenue, Bucharest 1, Romania

ARTICLE INFO

Article history:
Received 4 February 2009
Received in revised form 3 May 2010

MSC:
49M07
49M10
90C06
65K05

Keywords:
Unconstrained optimization
Conjugate gradient method
Sufficient descent condition
Conjugacy condition
Newton direction
Numerical comparisons

ABSTRACT

New accelerated nonlinear conjugate gradient algorithms which are mainly modifications of 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 orders 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, Hestenes–Stiefel, Polak–Ribière–Polyak, CONMIN conjugate gradient algorithms, limited quasi-Newton algorithm LBFGS and compare favorably with CG_DESCENT. In the frame of this numerical study the accelerated scaled memoryless BFGS preconditioned conjugate gradient ASCALCG algorithm proved to be more robust.

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 in [1]. A discussion of development of different versions of nonlinear conjugate gradient methods, with special attention to global convergence properties is presented in [2].

In this paper we suggest new nonlinear conjugate gradient algorithms for the solution of $\min f(x)$, where $f : R^n \to R$ is continuously differentiable and bounded below. Our algorithms are mainly modifications of the Dai and Yuan [3] 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 $g_k = \nabla f(x_k)$ and $s_k = x_{k+1} - x_k$. 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 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 [4], Hestenes and Stiefel [5],

E-mail address: nandrei@ici.ro.

[1] N. Andrei is member of Academy of Romanian Scientists, 54, Splaiul Independenţei, Bucharest sector 5, Romania.

0377-0427/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.cam.2010.05.002

Polak–Ribière [6] and Polyak [7], CG_DESCENT in [8], CONMIN in [9], as well as LBFGS in [10], using 750 unconstrained optimization test problems from the CUTE [11] library along with some other large-scale unconstrained optimization problems presented in [12]. Using the Dolan and Moré performance profiles [13] we prove these new accelerated conjugate gradient algorithms outperform the Dai–Yuan algorithm as well as its hybrid variants, Hestenes–Stiefel, Polak–Ribière–Polyak, CONMIN, LBFGS and compare favourably with CG_DESCENT by Hager and Zhang. The accelerated scaled memoryless BFGS preconditioned conjugate gradient ASCALCG algorithm [14] proved to be more robust.

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:

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

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 to be determined which follows. Here $y_k = g_{k+1} - g_k$ and $s_k = x_{k+1} - x_k$.

The line search in the conjugate gradient algorithms for α_k computation is often based on the standard Wolfe conditions [15,16]:

$$f(x_k + \alpha_k d_k) - f(x_k) \le \rho \alpha_k g_k^T d_k, \tag{2.5}$$

$$g_{k+1}^T d_k \ge \sigma g_k^T d_k, \tag{2.6}$$

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

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 [3]. 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

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

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 the 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 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*

$$g_{k+1}^T d_{k+1} \le -\left(\theta_{k+1} - \frac{1}{4}\right) \|g_{k+1}\|^2. \tag{2.7}$$

Proof. Since $d_0 = -g_0$, we have $g_0^T d_0 = -\|g_0\|^2$, which satisfy (2.7). Multiplying (2.3) by g_{k+1}^T, we have

$$g_{k+1}^T d_{k+1} = -\theta_{k+1} \|g_{k+1}\|^2 + \frac{(g_{k+1}^T g_{k+1})(g_{k+1}^T s_k)}{y_k^T s_k} - \frac{\|g_{k+1}\|^2 (s_k^T g_{k+1})^2}{(y_k^T s_k)^2}. \tag{2.8}$$

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

$$\begin{aligned}\frac{(g_{k+1}^T g_{k+1})(g_{k+1}^T s_k)}{y_k^T s_k} &= \frac{\left[(y_k^T s_k) g_{k+1}/\sqrt{2}\right]^T \left[\sqrt{2}(g_{k+1}^T s_k) g_{k+1}\right]}{(y_k^T s_k)^2} \\ &\leq \frac{\frac{1}{2}\left[\frac{1}{2}(y_k^T s_k)^2 \|g_{k+1}\|^2 + 2(g_{k+1}^T s_k)^2 \|g_{k+1}\|^2\right]}{(y_k^T s_k)^2} \\ &= \frac{1}{4}\|g_{k+1}\|^2 + \frac{(g_{k+1}^T s_k)^2 \|g_{k+1}\|^2}{(y_k^T s_k)^2}. \end{aligned} \tag{2.9}$$

Using (2.9) in (2.8) we get (2.7). □

To conclude, the sufficient descent condition from (2.7), 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 [3,4] 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 yields a descent. In our algorithm observe that, if for all k, $\theta_{k+1} > 1/4$, and the line search satisfies the Wolfe conditions (2.5) and (2.6), then for all k the search direction (2.3) and (2.4) satisfies the sufficient descent condition. It is well known that if the Wolfe line search conditions are satisfied, then $y_k^T s_k > 0$ and the steplength α_k is bounded away from zero [8]. Observe that $y_k^T s_k > 0$ is crucial in (2.4) for β_k^N computation. Note that in (2.7) we bound $g_{k+1}^T d_{k+1}$ by $-(\theta_{k+1} - 1/4)\|g_{k+1}\|^2$, while for the computational 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) When the initial point x_0 is near the solution of (2.1) and the Hessian of function f is a nonsingular matrix we know that the Newton direction is the best line search direction. Therefore, to get a good algorithm for solving (2.1) this is a very good motivation to choose the parameter θ_k in such a way that for every $k \geq 1$ the direction d_{k+1} given by (2.3) is the Newton direction. Therefore, from the equation

$$-\nabla^2 f(x_{k+1})^{-1} g_{k+1} = -\theta_{k+1} g_{k+1} + \beta_k^N s_k \tag{2.10}$$

after some algebra we get

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

Observe that the choice (2.11) does not imply that d_{k+1} given by (2.3) is the Newton direction. This is only a technical operation to get θ_{k+1} as in (2.11). 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 (2.10) 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:

$$\theta_{k+1} = \frac{1}{y_k^T g_{k+1}} \left[\|g_{k+1}\|^2 - \frac{\|g_{k+1}\|^2 (s_k^T g_{k+1})}{y_k^T s_k} + s_k^T g_{k+1}\right]. \tag{2.12}$$

Observe that if θ_{k+1} given by (2.12) is greater than or equal to 1/4, then according to Theorem 2.1 the direction (2.3) satisfies the sufficient descent condition (2.7). On the other hand, if in (2.12) $\theta_{k+1} < 1/4$, 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 computations 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:

$$\theta_{k+1} = \frac{1}{y_k^T g_{k+1}} \left[\|g_{k+1}\|^2 - \frac{\|g_{k+1}\|^2 (s_k^T g_{k+1})}{y_k^T s_k}\right]. \tag{2.13}$$

As above, if θ_{k+1} given by (2.13) is greater than or equal to 1/4, then according to Theorem 2.1 the direction (2.3) satisfies the sufficient descent condition (2.7). On the other hand, if in (2.13) $\theta_{k+1} < 1/4$, then we take $\theta_{k+1} = 1$ in (2.3).

Observe that since $s_k^T g_{k+1} \to 0$ along the iterations, θ_k given by (2.12) obtained from the Newton direction paradigm is very similar to (2.13) based on the conjugacy condition. Besides, θ_{k+1} from (2.13) can be written as

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

Since at every iteration d_k is a descent direction and α_k is computed by the Wolfe line search (2.5) and (2.6), it follows that $g_k^T g_{k+1} \to 0$. (This is reminiscence from the steepest descent method.) Therefore, along the iterations, $\theta_k \to 1$.

In [4] Dai and Yuan proved the global convergence of a conjugate gradient algorithm for which $\beta_k = \beta_k^{\mathrm{DY}} t_k$, where $t_k \in [-c, 1]$ with $c = (1-\sigma)/(1+\sigma)$. Our algorithm is a proper modification of the Dai and Yuan's with the following property.

Observe that

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

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 $\frac{s_k^T g_{k+1}}{y_k^T s_k} \geq \frac{-\sigma}{1-\sigma}$. Hence $r_k \leq \frac{1}{1-\sigma}$. Therefore,

$$\beta_k^N \leq \beta_k^{\mathrm{DY}} \frac{1}{1-\sigma}. \tag{2.16}$$

3. Convergence analysis

In this section we analyze the convergence of the algorithm (2.2), (2.3), (2.4), and (2.12) or (2.13) 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 a strong Wolfe line search, we have the following results given by Lemmas 3.1 and 3.2, which were first proved in [19,15,16]. 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*

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

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*

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

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

$$d_k = -\theta_k g_k + \alpha_{k-1}\beta_{k-1}^N d_{k-1} \tag{3.5}$$

and $\theta_k > 1/4$, *with a 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.* [20].

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

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

On the other hand, from (3.5) we get

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

Since d_k is a descent direction, it follows that

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

Therefore,

$$\alpha_{k-1} \left|\beta_{k-1}^N\right| \left|g_k^T d_{k-1}\right| + \left|g_k^T d_k\right| \geq \theta_k \|g_k\|^2 .$$

From the strong Wolfe condition we have that

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

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

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

where $c = \theta_k^2/(1 + \sigma^2)$ is a positive constant. Therefore, from (3.10) we get

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

From (3.8) observe that

$$\frac{\|d_k\|^2}{\|d_{k-1}\|^2} \geq (\alpha_{k-1}\beta_{k-1}^N)^2 - \theta_k^2 \frac{\|g_k\|^2}{\|d_{k-1}\|^2}.$$

Therefore,

$$\frac{(g_k^T d_k)^2}{\|d_k\|^2} + \frac{(g_{k-1}^T d_{k-1})^2}{\|d_{k-1}\|^2} \geq \frac{\|g_k\|^4}{\|d_k\|^2}\left[c - \theta_k^2 \frac{(g_{k-1}^T d_{k-1})^2}{\|d_{k-1}\|^2}\frac{1}{\|g_k\|^2}\right], \tag{3.11}$$

where $\theta_k > 1/4$. 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

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

holds for all sufficiently large k. Hence, the inequality (3.7) follows from the 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*

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

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* (2.12) *or* (2.13)*, 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 as* $\gamma \leq \|\nabla f(x)\|$, $1/4 \leq \theta_k \leq \tau$*, 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, from (2.16) we have

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

where

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

Therefore

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

This relation shows that

$$\sum_{k\geq 1} \frac{1}{\|d_k\|^2} \geq \frac{1}{(\tau\Gamma + \eta)^2} \sum_{k\geq 1} 1 = \infty. \tag{3.14}$$

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 comparison between conjugate gradient methods and the limited memory quasi Newton method in [10] shows that the latter is more successful [22]. 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 [23]). In an 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 a Newton direction (AMDYN) or with a 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, otherwise set $k = k + 1$.

Step 3. *Line search.* Compute α_k satisfying the Wolfe line search conditions (2.5) and (2.6).

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 (2.12). For the algorithm AMDYC, θ_{k+1} is computed as in (2.13). 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 (2.5) and (2.6). 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 $-g_{k+1}$ [4]. 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 (2.5), (2.6) and (4.1) are sufficient to prove the global convergence of the algorithm.

The initial selection 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 $\alpha_{k-1} \|d_{k-1}\|_2 / \|d_k\|_2$. This selection, was considered for the first time by Shanno and Phua in CONMIN [9]. It is also considered in the packages: SCG in [25] and ASCALCG in [14].

For uniformly convex functions, like in [23], 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 $S = \{x : f(x) \leq f(x_0)\}$, 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).* □

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 [12]. For each function we have considered ten numerical experiments with the increasing number of variables $n = 1000, 2000, \ldots, 10\,000$. 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

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

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, 1.8 GHz 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.

Both algorithms have similar performances. However, subject to the cpu time metric, AMDYN proves to be slightly better. In the following we shall compare AMDYN versus some known conjugate gradient algorithms.

In the second set of numerical experiments we compare the AMDYN algorithm with the Dai and Yuan (DY) algorithm. Fig. 1 presents the Dolan–Moré performance profile for these algorithms subject to the cpu time metric. We see that AMDYN is the top performer, being more successful and more robust than the Dai and Yuan algorithm. When comparing AMDYN

Table 1
Performance of AMDYN versus AMDYC. 750 problems.

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

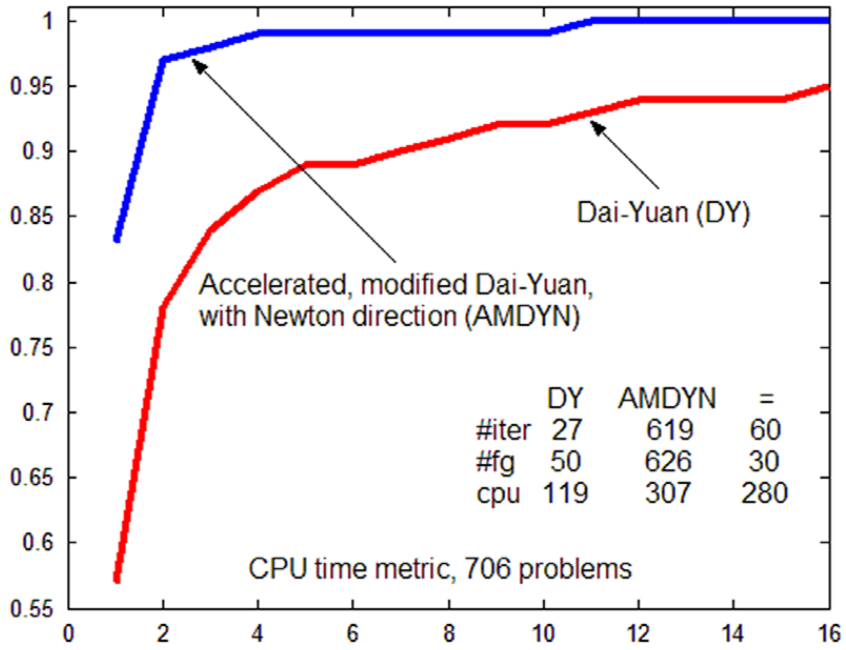

Fig. 1. Performance profile of AMDYN versus DY.

with the Dai and Yuan algorithm (Fig. 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.

Dai and Yuan [4] studied the hybrid conjugate gradient algorithms and proposed the following two hybrid methods:

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

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

where $\beta_k^{\mathrm{HS}} = y_k^T g_{k+1}/y_k^T s_k$, showing their global convergence when the Lipschitz assumption holds and the standard Wolfe line search is used. The numerical experiments of Dai and Ni [26] proved that the second hybrid method (hDYz) is the better, outperforming the Polak–Ribière [6] and Polyak [7] method. In the third set of numerical experiments we compare the Dolan–Moré performance profile of AMDYN versus Dai–Yuan hybrid conjugate gradient β_k^{hDY} subject to the cpu time metric, as in Fig. 2. Observe that the differences are substantial. Again AMDYN is the top performer.

In the fourth set of numerical experiments, in Fig. 3, we compare the Dolan–Moré performance profile of AMDYN versus the Dai–Yuan hybrid conjugate gradient β_k^{hDYz} subject to the cpu time metric. Again observe that AMDYN is the top performer.

In the fifth set of numerical experiments we compare AMDYN versus the Hestenes–Stiefel conjugate gradient algorithm ($\beta_k^{\mathrm{HS}} = y_k^T g_{k+1}/y_k^T s_k$). Fig. 4 presents the performance profiles of these algorithms. The HS method has the property that the conjugacy condition $y_k^T d_{k+1} = 0$ always holds, independent of the line search. On the other hand, the AMDYN algorithm satisfies the Dai–Liao conjugacy condition $y_k^T d_{k+1} = -s_k^T g_{k+1}$ which is a little more relaxed than the pure conjugate condition $y_k^T d_{k+1} = 0$.

In the sixth set of numerical comparisons we consider AMDYN versus the Polak–Ribière–Polyak conjugate gradient algorithm ($\beta_k^{\mathrm{PRP}} = y_k^T g_{k+1}/g_k^T g_k$). Fig. 5 presents the performance profiles of these algorithms subject to the cpu time metric. The PRP method, like HS, posses a very important built-in restart feature that addresses jamming directly. The idea is that PRP (and HS) method automatically adjust the value of the parameter β_k^{PRP} to avoid jamming. In general, the performance of these methods (PRP and HS) is better than the performance of some other conjugate gradient methods (for example DY) [2]. However, from Fig. 5 observe that AMDYN is the top performer again among these algorithms. AMDYN inherits some convergence properties from the Newton method (see (2.10)).

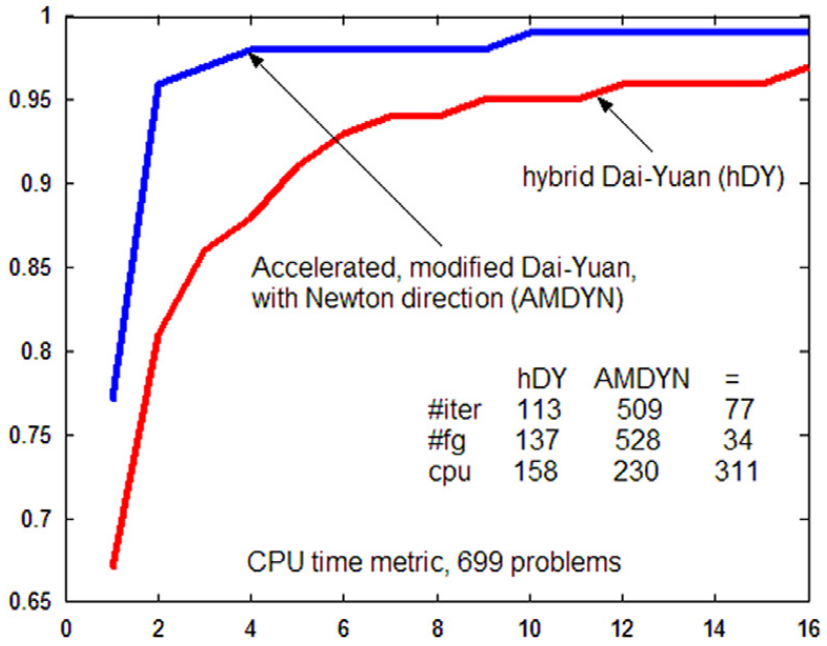

Fig. 2. Performance profile of AMDYN versus hDY.

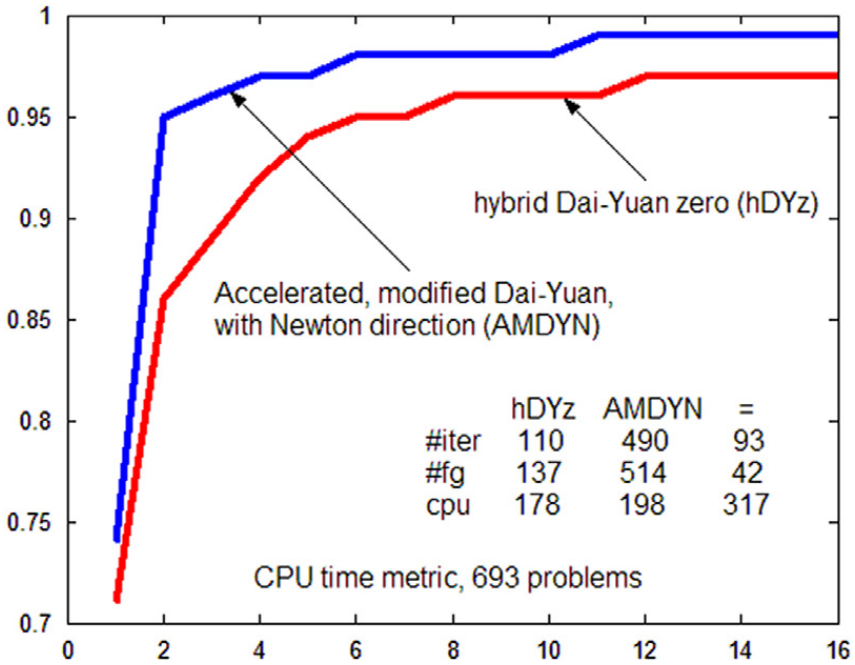

Fig. 3. Performance profile of AMDYN versus hDYz.

In the next set of numerical experiments we compare AMDYN versus CG_DESCENT in [8]. Fig. 6 presents the Dolan and Moré cpu time performance profile of AMDYN versus CG_DESCENT with the Wolfe line search. Presently CG_DESCENT is the practical conjugate gradient algorithm with a better 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 [8] proved that the direction d_k in their algorithm satisfies the sufficient descent condition $g_k^T d_k \leq -(7/8)\,\|g_k\|^2$.

At every iteration, the AMDYN algorithm satisfies the sufficient descent condition (2.7), where $\theta_k \to 1$. Therefore, at least in the last part of the iterations AMDYN satisfies the sufficient descent condition $g_k^T d_k \leq -(3/4)\,\|g_k\|^2$. 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. On the other hand, AMDYN uses an acceleration scheme which modifies the step length given by the classical Wolfe condition (2.5) and (2.6) in order to improve the reduction of the function values along the iterations.

In the following, we compare AMDYN versus COMNIN in [9]. Fig. 7 presents the performance profiles of these algorithms.

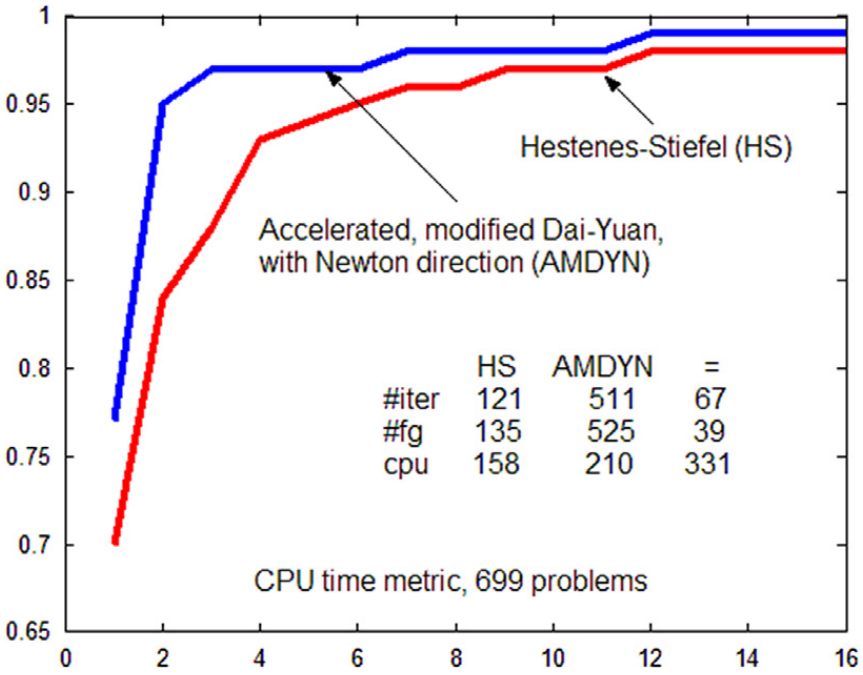

Fig. 4. Performance profile of AMDYN versus HS.

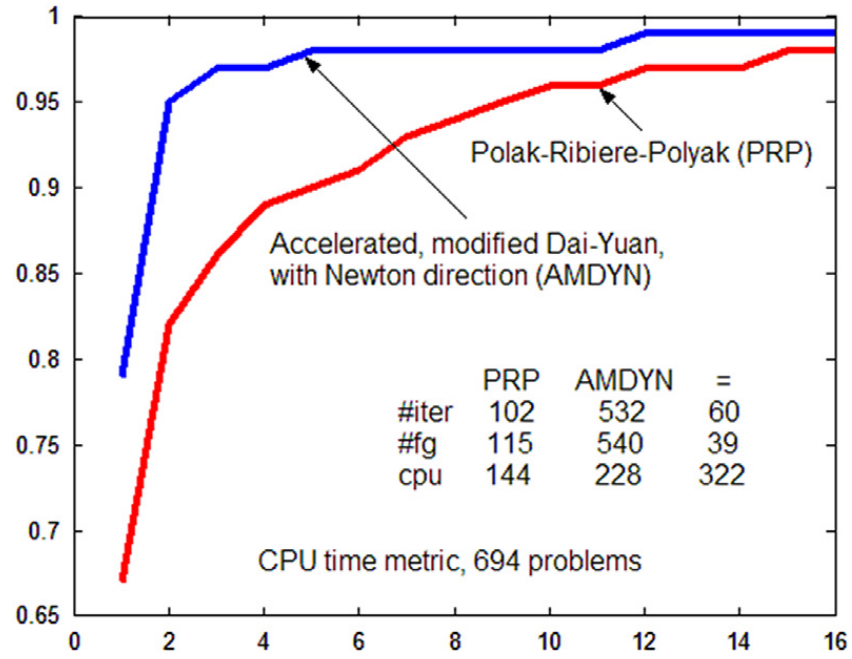

Fig. 5. Performance profile of AMDYN versus PRP.

COMNIN in [9] is a conjugate gradient algorithm which may be interpreted as a memoryless BFGS quasi-Newton algorithm optimally scaled in the sense of Oren and Spedicato [27]. In CONMIN the scaling is combined with the Powell's restart criterion. The direction d_{k+1} in CONMIN is computed as

$$d_{k+1} = -H_{k+1}g_{k+1} + A_k y_k - B_k s_k, \tag{5.4}$$

where H_{k+1} is the BFGS approximation of the inverse Hessian which at every iteration is initialized with identity matrix and A_k and B_k are specific matrices. The main drawback of this method is that if H_{k+1} contains useful information about the Hessian of the function f, then we are better off using the search direction $d_{k+1} = -H_{k+1}g_{k+1}$ since the addition of the last terms in (5.4) may prevent d_{k+1} from being a descent direction unless the line search is sufficiently accurate. The same is the case for the AMDYN algorithm. The parameter θ_{k+1} in (2.3) given by (2.12) is computed to get as much as possible information from the inverse Hessian by the secant condition. However, the approximation of the inverse Hessian used in

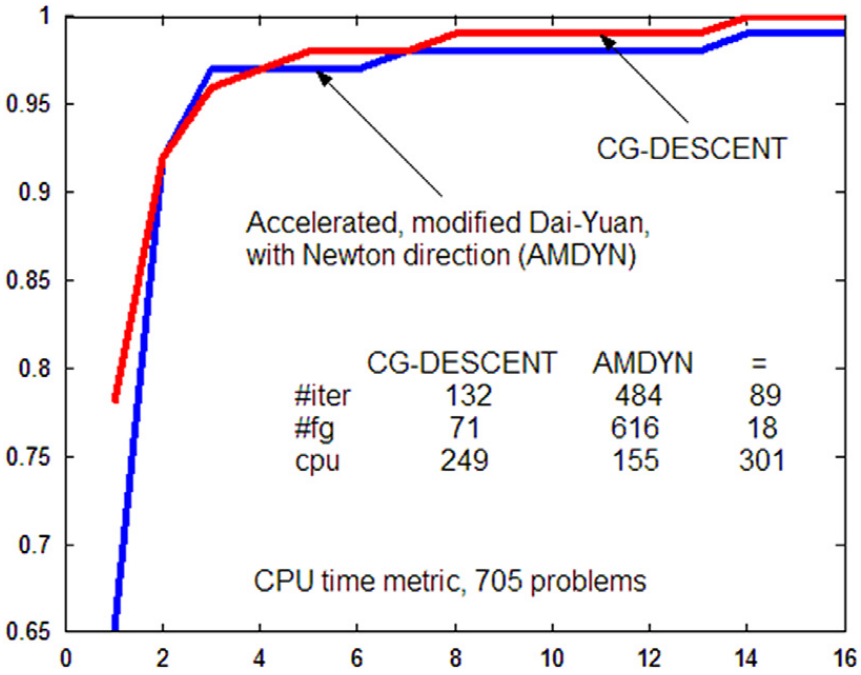

Fig. 6. Performance profile of AMDYN versus CG_DESCENT.

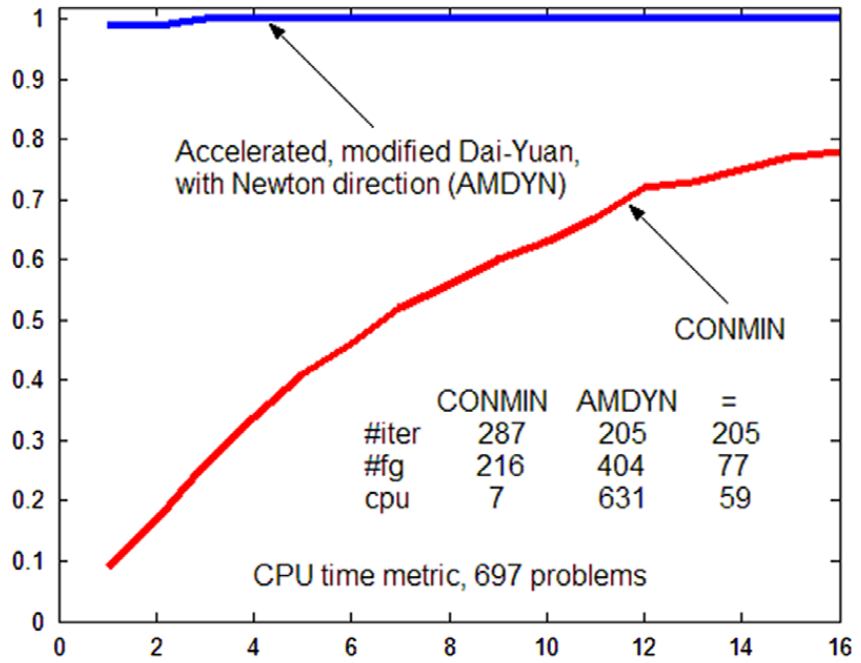

Fig. 7. Performance profile of AMDYN versus CONMIN.

AMDYN is scantier than that used in CONMIN. In Fig. 7 we have the computational evidence that subject to the cpu time metric, AMDYN is the top performer and outperforms COMNIN.

In another set of numerical experiments we compare AMDYN versus ASCALCG [14], as in Fig. 8. In Fig. 9 the performance profiles of AMDYN, CG_DESCENT and ASCALCG algorithms are presented.

ASCALCG is an accelerated scaled memoryless BFGS preconditioned conjugate gradient algorithm with Beale–Powell restart criterion, in which the parameter scaling the gradient is selected as the spectral gradient. The top curve in Fig. 8 corresponds to ASCALCG. Observe that subject to the cpu time metric, ASCALCG is more robust. Also, it is worth seeing in Fig. 8 that for $\tau = 1$, relative to the cpu time metric AMDYN is better. However, for $\tau > 1$, ASCALCG turns out to be faster and more robust than AMDYN, at least for this set of numerical experiments.

On the other hand, from Fig. 9 we see that among these conjugate gradient algorithms, ASCALCG is the top performer. Also, for $\tau = 1$, relative to the cpu time metric, CG_DESCENT is the best algorithm. In this computational scheme the

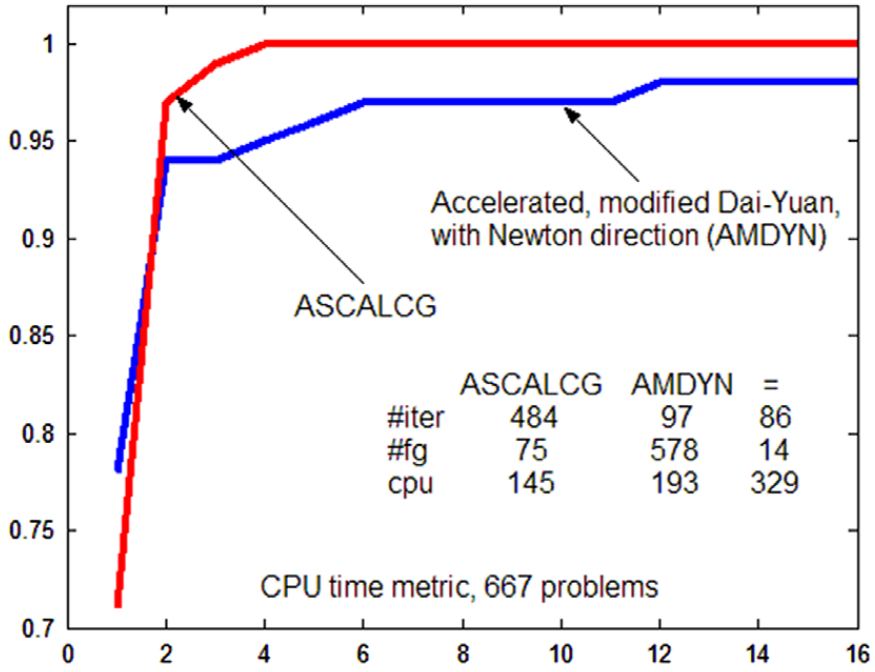

Fig. 8. Performance profile of AMDYN versus ASCALCG.

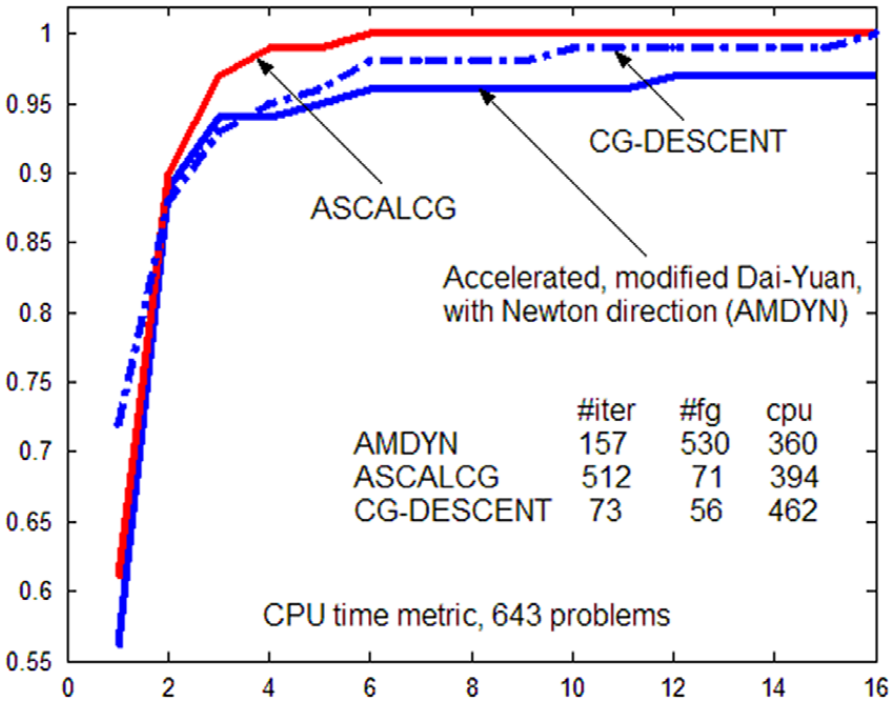

Fig. 9. Performance profile of AMDYN versus CG_DESCENT and ASCALCG.

direction d_{k+1} is generated by the rule:

$$d_{k+1} = -g_{k+1} + \beta_k^{\mathrm{HZ}} d_k, \qquad \beta_k^{\mathrm{HZ}} = \frac{1}{y_k^T d_k}\left(y_k - 2\frac{\|y_k\|^2}{y_k^T d_k} d_k\right)^T g_{k+1}. \tag{5.5}$$

This scheme is obtained by deleting a term from the search direction for the memoryless quasi-Newton scheme of Perry [28] and Shanno [29,30]. Observe that CG_DESCENT is a modification of HS and was devised in order to ensure sufficient descent, independent of the accuracy of the line search. These algorithms (and codes) differ in many respects. Although the update formula for direction computation in ASCALCG (see (2.9) and (2.17) in [14]) is more complicated than (2.3), (2.4) and (2.12) or (5.5), in numerical experiments this computational scheme proved to be efficient and more robust than AMDYN and

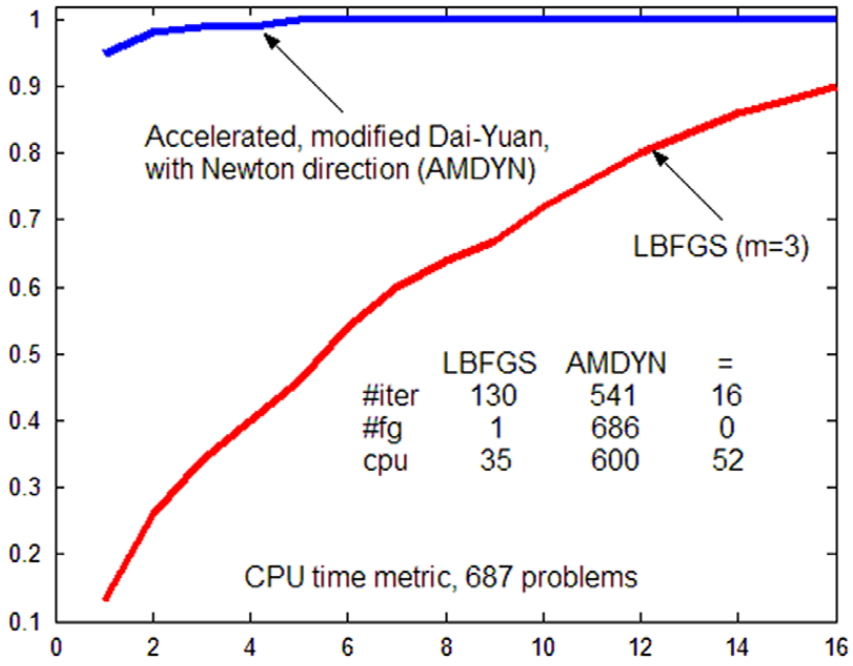

Fig. 10. Performance profile of AMDYN versus LBFGS ($m = 3$).

CG_DESCENT respectively. However, since each of these codes is different in the number of parameters which can be modified by the user to establish a context of optimization (AMDYN has 8 parameters, CG_DESCENT has 26 parameters while ASCALCG has 10 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.

Finally we compare AMDYN versus limited memory quasi-Newton L-BFGS ($m = 3$) in [10] as in Fig. 10, where m is the number of pairs (s_k, y_k) used. Observe that AMDYN is the top performer.

One explanation is that the linear algebra in the LBFGS code to update the search direction is more time consuming than the linear algebra in AMDYN. On the other hand the steplength in LBFGS is determined at each iteration by means of the line search routine MCVSRCH, which is a slight modification of the routine CSRCH written in [31]. More deeply, the conjugate gradient algorithms possess the so called regularization property [32], which is closely linked to the ill-conditioned (ill-posed) problems. This property is connected with a search in the subspace of the dominant Hessian eigenvectors, i.e. the eigenvectors corresponding to the maximal eigenvalues. Thus the search along the direction given by a conjugate gradient algorithm means the search is given in the subspace of the Hessian dominant eigenvectors. The subspace of eigenvectors corresponding to the small eigenvalues is implicitly neglected, thus providing the regularization effect. In practice, it is observed that the convergence is fast during the first iterations. Then it slows down after a relatively small number of iterations. Possibly, this is linked to the number of Hessian dominant eigenvalues [33]. From this viewpoint, the regularization is an important property of conjugate gradient algorithms [32].

6. Conclusion

In this paper we have presented new conjugate gradient algorithms 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 at every iteration 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. Using a large set of 750 test unconstrained optimization problems, with the number of variables in the range [1000, 10 000], a numerical study concerning the behavior of these new algorithms versus some known conjugate gradient algorithms and the limited memory quasi-Newton L-BFGS algorithm has been presented. We have the computational evidence that the performance of our algorithm AMDYN is higher than that of known conjugate gradient algorithms including: the Dai and Yuan conjugate gradient algorithm and its hybrid variants, Hestenes–Stiefel, Polak–Ribière–Polyak, CONMIN and the limited-memory quasi-Newton LBFGS ($m = 3$). The AMDYN algorithm compares favorably with CG_DESCENT, which is one of the fastest conjugate gradient algorithms for large-scale unconstrained optimization. In the metrics and problems presented both AMDYN and CG_DESCENT seem to perform similarly. However, in the frame of this numerical study the accelerated scaled memoryless BFGS preconditioned conjugate gradient ASCALCG algorithm [14] proved to be more robust than AMDYN. This brings us to the question of preconditioning which is not straightforward. At present a large number of conjugate gradient algorithms is known [1]. The research effort to get the fastest and the more robust is still very active. For example the accelerated conjugate gradient algorithm with guaranteed sufficient descent and conjugacy conditions is very promising [34].

References

[1] N. Andrei, 40 conjugate gradient algorithms for unconstrained optimization. A survey on their definition, ICI Technical Report No. 13/08, March 14, 2008.
[2] W.W. Hager, H. Zhang, A survey of nonlinear conjugate gradient methods, Pacific J. Optim. 2 (2006) 35–58.
[3] Y.H. Dai, Y. Yuan, A nonlinear conjugate gradient method with a strong global convergence property, SIAM J. Optim. 10 (1999) 177–182.
[4] Y.H. Dai, Y. Yuan, An efficient hybrid conjugate gradient method for unconstrained optimization, Ann. Oper. Res. 103 (2001) 33–47.
[5] M.R. Hestenes, E.L. Stiefel, Methods of conjugate gradients for solving linear systems, J. Res. Natl. Bur. Stand. 49 (1952) 409–436.
[6] E. Polak, G. Ribière, Note sur la convergence de méthodes de directions conjuguée, Rev. Fr. Autom. Inform. Rech. Oper. 3e Année 16 (1969) 35–43.
[7] B.T. Polyak, The conjugate gradient method in extreme problems, USSR Comput. Math. Math. Phys. 9 (1969) 94–112.
[8] W.W. Hager, H. Zhang, A new conjugate gradient method with guaranteed descent and an efficient line search, SIAM J. Optim. 16 (2005) 170–192.
[9] D.F. Shanno, V. Phua, Algorithm 500, minimization of unconstrained multivariate functions, ACM Trans. Math. Software 2 (1976) 87–94.
[10] D.C. Liu, J. Nocedal, On the limited memory BFGS method for large scale optimization, Math. Program. 45 (1989) 503–528.
[11] I. Bongartz, A.R. Conn, N.I.M. Gould, Ph.L. Toint, CUTE: constrained and unconstrained testing environments, ACM Trans. Math. Software 21 (1995) 123–160.
[12] N. Andrei, An unconstrained optimization test functions collection, Adv. Model. Optim. 10 (2008) 147–161.
[13] E.D. Dolan, J.J. Moré, Benchmarking optimization software with performance profiles, Math. Program. 91 (2002) 201–213.
[14] N. Andrei, Accelerated scaled memoryless BFGS preconditioned conjugate gradient algorithm for unconstrained optimization, European J. Oper. Res. 204 (2010) 410–420.
[15] P. Wolfe, Convergence conditions for ascent methods, SIAM Rev. 11 (1969) 226–235.
[16] P. Wolfe, Convergence conditions for ascent methods II: some corrections, SIAM Rev. 13 (1971) 185–188.
[17] Y.H. Dai, New properties of a nonlinear conjugate gradient method, Numer. Math. 89 (2001) 83–98.
[18] Y.H. Dai, L.Z. Liao, New conjugacy conditions and related nonlinear conjugate gradient methods, Appl. Math. Optim. 43 (2001) 87–101.
[19] G. Zoutendijk, Nonlinear programming computational methods, in: J. Abadie (Ed.), Integer and Nonlinear Programming, North-Holland, Amsterdam, 1970, pp. 37–86.
[20] Y.H. Dai, J.Y. Han, G.H. Liu, D.F. Sun, X. Yin, Y. Yuan, Convergence properties of nonlinear conjugate gradient methods, SIAM J. Optim. 10 (1999) 348–358.
[21] J. Nocedal, Conjugate gradient methods and nonlinear optimization, in: L. Adams, J.L. Nazareth (Eds.), Linear and Nonlinear Conjugate Gradient—Related Methods, SIAM, Philadelphia, 1996, pp. 9–23.
[22] N. Andrei, Performance profiles of conjugate gradient algorithms for unconstrained optimization, in: C.A. Floudas, P.M. Pardalos (Eds.), 2nd ed., in: Encyclopedia of Optimization, vol. P, Springer, New York, 2009, pp. 2938–2953.
[23] N. Andrei, Acceleration of conjugate gradient algorithms for unconstrained optimization, Appl. Math. Comput. 213 (2009) 361–369.
[24] M.J.D. Powell, Restart procedures for the conjugate gradient method, Math. Program. 12 (1977) 241–254.
[25] E. Birgin, J.M. Martínez, A spectral conjugate gradient method for unconstrained optimization, Appl. Math. Comput. 43 (2001) 117–128.
[26] Y.H. Dai, Q. Ni, Testing different conjugate gradient methods for large-scale unconstrained optimization, J. Comput. Math. 21 (2003) 311–320.
[27] S.S. Oren, E. Spedicato, Optimal conditioning of self-scaling variable metric algorithms, Math. Program. 10 (1976) 70–90.
[28] J.M. Perry, A class of conjugate gradient algorithms with a two step variable metric memory, Discussion Paper 269, Center for Mathematical Studies in Economics and Management Science, Northwestern University, 1977.
[29] D.F. Shanno, Conjugate gradient methods with inexact searches, Math. Oper. Res. 3 (1978) 244–256.
[30] D.F. Shanno, On the convergence of a new conjugate gradient algorithm, SIAM J. Numer. Anal. 15 (1978) 1247–1257.
[31] J.J. Moré, D.J. Thuente, Line search algorithms with guaranteed sufficient decrease, ACM Trans. Math. Softw. 20 (1994) 286–307.
[32] P.C. Hansen, Rank Deficient and Discrete Ill-Posed Problems, SIAM, Philadelphia, 1998.
[33] A.K. Alekseev, I.M. Navon, J.L. Steward, Comparison of advanced large-scale minimization algorithms for the solution of inverse ill-posed problems, Optim. Methods Softw. 24 (1) (2009) 63–87.
[34] N. Andrei, Another accelerated conjugate gradient algorithm with guaranteed descent and conjugacy conditions for large-scale unconstrained optimization, ICI Technical Report No. 7/2010, January 29, 2010.