Another Nonlinear Conjugate Gradient Algorithm for Unconstrained Optimization

Neculai Andrei
Research Institute for Informatics,
Center for Advanced Modeling and Optimization,
8-10, Averescu Avenue, Bucharest 1, Romania
E-mail: nandrei@ici.ro
Academy of Romanian Scientists,
54, Splaiul Independentei, Bucharest 5, Romania

Abstract. A nonlinear conjugate gradient algorithm which is a modification of the Dai and Yuan [Y.H. Dai and Y, Yuan, *A nonlinear conjugate gradient method with a strong global convergence property,* SIAM J. Optim., 10 (1999), pp.177-182.] conjugate gradient algorithm satisfying a parametrized sufficient descent condition with a parameter δ_k is proposed. The parameter δ_k is computed by means of the conjugacy condition, thus an algorithm which is a positive multiplicative modification of the Hestenes and Stiefel [M.R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Research Nat. Bur. Standards Sec. B. 48, pp. 409-436, 1952.] algorithm is obtained. The algorithm can be viewed as an adaptive version of the Dai and Liao [Y.H. Dai and L.Z. Liao, *New conjugacy conditions and related nonlinear conjugate gradient methods.* Appl. Math. Optim., 43 (2001), pp. 87-101.] conjugate gradient algorithm. Close to our computational scheme is the conjugate gradient algorithm recently proposed by Hager and Zhang [W.W. Hager and H. Zhang, *A new conjugate gradient method with guaranteed descent and an efficient line search*, SIAM Journal on Optimization, 16 (2005), 170-192.]. Computational results, for a set consisting of 750 unconstrained optimization test problems, show that this new conjugate gradient algorithm substantially outperforms the known conjugate gradient algorithms.

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

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 [16], as well as by O'Leary [23]. An excellent survey of development of different versions of nonlinear conjugate gradient methods, with special attention to global convergence properties, is presented by Hager and Zhang [18]. This family of algorithms includes a lot of variants, well known in the literature, with important convergence properties and numerical efficiency.

In this paper we propose a nonlinear conjugate gradient algorithm which is a modification of the Dai and Yuan [11] conjugate gradient algorithm satisfying a parametrized sufficient descent condition with a parameter δ_k. Under exact line search the algorithm reduces to the Dai and Yuan computational scheme [11]. For different choices of the parameter δ_k, the performance of this algorithm can be quite different. In order to get an efficient algorithm, the parameter δ_k is computed by means of the conjugacy condition, thus obtaining another algorithm which is a positive multiplicative modification of the Hestenes and Stiefel algorithm [19]. The new algorithm can be viewed as an adaptive version of the Dai and Liao [9] conjugate gradient algorithm. Close to our computational scheme is the

conjugate gradient algorithm recently proposed by Hager and Zhang [17]. Even that the direction generated by this algorithm is not a descent one at every iteration, however the algorithm has a built-in restart feature that directly addresses to the jamming phenomenon.

The structure of the paper is as follows. In section 2 we present the new conjugate gradient algorithm. Section 3 is devoted to the convergence analysis for uniformly convex functions. Section 4 presents intensive numerical results and comparisons of our algorithm versus 20 nonlinear conjugate gradient algorithms, subject to the number of iterations, the number of function and its gradient evaluations, as well as subject to the CPU time on a set consisting of 750 unconstrained optimization problems from the CUTE collection [7] along with some other large-scale unconstrained optimization problems presented in [5]. We present computational evidence that the performances of our algorithm are substantially higher than those of the known conjugate gradient algorithms, at least for this set of 750 problems.

2. The algorithm

For solving the unconstrained optimization problem

$$\min\left\{ f(x) : x \in R^n \right\}, \tag{1}$$

where $f : R^n \to R$ is continuously differentiable, Dai and Yuan [11] suggested the following nonlinear conjugate gradient algorithm:

$$x_{k+1} = x_k + \alpha_k d_k, \tag{2}$$

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

$$d_{k+1} = -g_{k+1} + \beta_k^{DY} s_k, \quad d_0 = -g_0, \tag{3}$$

$$\beta_k^{DY} = \frac{g_{k+1}^T g_{k+1}}{y_k^T s_k}, \tag{4}$$

where $g_k = \nabla f(x_k)$ and $y_k = g_{k+1} - g_k$, $s_k = x_{k+1} - x_k$. Using a standard Wolfe line search [32, 33], the Dai and Yuan method always generates descent directions and under Lipschitz assumption it is globally convergent.

In this paper we present a modification of the Dai and Yuan computational scheme as:

$$d_{k+1} = -g_{k+1} + \beta_k^A s_k, \quad d_0 = -g_0, \tag{5}$$

where

$$\beta_k^A = \underbrace{\frac{g_{k+1}^T g_{k+1}}{y_k^T s_k}}_{\beta_k^{DY}} - \delta_k \frac{(g_{k+1}^T s_k)(g_{k+1}^T g_{k+1})}{(y_k^T s_k)^2}. \tag{6}$$

and δ_k is a positive parameter which follows to be determined. We assume that $y_k^T s_k \neq 0$, so that β_k^A is well defined. (Using a standard Wolfe line search, $y_k^T s_k > 0$ when $g_k \neq 0$.) The general form (6) which is used to deduce the β_k^A in (5) has also been proposed in [34]. The motivation of the proposed β_k^A in (6) we introduced in this paper is that under suitable conditions it assures the descent character of the direction d_{k+1} in (5). We prove this in the following theorem.

Theorem 1. *If* $y_k^T s_k \neq 0$ *and* $d_{k+1} = -g_{k+1} + \beta_k^A s_k$, *(*$d_0 = -g_0$*), where* β_k^A *is given by (6), then*

$$g_{k+1}^T d_{k+1} \leq -\left(1 - \frac{1}{4\delta_k}\right) \left\| g_{k+1} \right\|^2. \tag{7}$$

Proof. Since $d_0 = -g_0$, we have $g_0^T d_0 = -\|g_0\|^2$, which satisfy (7). Multiplying (5) by g_{k+1}^T, we have

$$g_{k+1}^T d_{k+1} = -\|g_{k+1}\|^2 + \frac{(g_{k+1}^T g_{k+1})(g_{k+1}^T s_k)}{y_k^T s_k} - \delta_k \frac{\|g_{k+1}\|^2 (s_k^T g_{k+1})^2}{(y_k^T s_k)^2}. \tag{8}$$

But

$$\begin{aligned}
\frac{(g_{k+1}^T g_{k+1})(g_{k+1}^T s_k)}{y_k^T s_k} &= \frac{\left[(y_k^T s_k) g_{k+1} / \sqrt{2\delta_k}\right]^T \left[\sqrt{2\delta_k}\,(g_{k+1}^T s_k) g_{k+1}\right]}{(y_k^T s_k)^2} \\
&\le \frac{\frac{1}{2}\left[\frac{1}{2\delta_k}(y_k^T s_k)^2 \|g_{k+1}\|^2 + 2\delta_k (g_{k+1}^T s_k)^2 \|g_{k+1}\|^2\right]}{(y_k^T s_k)^2} \\
&= \frac{1}{4\delta_k}\|g_{k+1}\|^2 + \delta_k \frac{(g_{k+1}^T s_k)^2 \|g_{k+1}\|^2}{(y_k^T s_k)^2}.
\end{aligned} \tag{9}$$

Using (9) in (8) we get (7). ■

To conclude the sufficient descent condition from (7), the quantity $1 - 1/(4\delta_k)$ is required to be nonnegative. Supposing that $1 - 1/(4\delta_k) > 0$, then the direction given by (5) and (6) is a descent direction. Dai and Yuan [11, 12] 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, $1/(4\delta_k) \le 1$, and the line search satisfies the Wolfe conditions, then for all k the search direction (5) and (6) satisfy the sufficient descent condition.

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 (6) for β_k^A reduces to the Dai and Yuan computational scheme [11, 12]. However, in this paper we consider general nonlinear functions and inexact line search.

Our numerical experience with algorithm (2), (5) and (6) shows that for different choices of δ_k, its performance is quite different. Therefore, in order to get an efficient algorithm in the following we shall present a procedure for δ_k computation. Mainly, this is based on the conjugacy condition. Dai and Liao [9] introduced a new conjugacy condition as $y_k^T d_{k+1} = -t s_k^T g_{k+1}$, where $t \ge 0$ is a scalar. This is reasonable since in general the inexact line search is used in real computation. However, this condition is very dependent by the nonnegative parameter t, for which we do not know any formula to choose it. Therefore, even that in our developments we use the inexact line search we adopt a more conservative approach and consider the conjugacy condition $y_k^T d_{k+1} = 0$.

Observe that using (6) in (5) we get the following direction:

$$d_{k+1} = -g_{k+1} + \frac{g_{k+1}^T g_{k+1}}{y_k^T s_k} s_k - \delta_k \frac{\|g_{k+1}\|^2}{(y_k^T s_k)^2} (g_{k+1}^T s_k) s_k \tag{10}$$

which can be written as

$$d_{k+1} = -Q_{k+1} g_{k+1}, \tag{11}$$

where the matrix Q_{k+1} is:

$$Q_{k+1} = I - \frac{s_k g_{k+1}^T}{y_k^T s_k} + \delta_k \frac{\|g_{k+1}\|^2}{(y_k^T s_k)^2}(s_k s_k^T). \tag{12}$$

Now, by symmetrization of Q_{k+1} as:

$$\bar{Q}_{k+1} = I - \frac{s_k g_{k+1}^T + g_{k+1} s_k^T}{y_k^T s_k} + \delta_k \frac{\|g_{k+1}\|^2}{(y_k^T s_k)^2}(s_k s_k^T), \tag{13}$$

we can consider the direction

$$d_{k+1} = -\bar{Q}_{k+1} g_{k+1}. \tag{14}$$

The reason of symmetrization the Q_{k+1} as $\bar{Q}_{k+1}$ in (13) is that the direction d_{k+1} computed as in (14) resembles the quasi-Newton methods. However, in this paper we use only the symmetry and we don't modify further $\bar{Q}_{k+1}$ in order to satisfy the quasi-Newton equation.

Now, from the conjugacy condition $y_k^T d_{k+1} = 0$, i.e.

$$y_k^T \bar{Q}_{k+1} g_{k+1} = 0, \tag{15}$$

after some algebra it follows that

$$\delta_k = \frac{y_k^T s_k}{g_{k+1}^T s_k} + \frac{g_{k+1}^T y_k}{\|g_{k+1}\|^2} - \frac{(g_{k+1}^T y_k)(y_k^T s_k)}{\|g_{k+1}\|^2 (g_{k+1}^T s_k)}. \tag{16}$$

Therefore, using (16) in (6) we get

$$\beta_k^A = \frac{1}{y_k^T s_k}\left(y_k - \frac{g_{k+1}^T y_k}{y_k^T s_k} s_k\right)^T g_{k+1}. \tag{17}$$

Observe that β_k^A from (17) can be written as:

$$\beta_k^A = \underbrace{\frac{y_k^T g_{k+1}}{y_k^T s_k}}_{\beta_k^{HS}}\left[1 - \frac{s_k^T g_{k+1}}{y_k^T s_k}\right] = \underbrace{\frac{y_k^T g_{k+1}}{y_k^T s_k}}_{\beta_k^{HS}}\left(-\frac{s_k^T g_k}{y_k^T s_k}\right). \tag{18}$$

Supposing that d_k is a descent direction, i.e. $g_k^T s_k \le 0$, and the step length α_k is determined by the Wolfe line search conditions, then we see that β_k^A is a positive multiplicative modification of $\beta_k^{HS} = y_k^T g_{k+1} / y_k^T s_k$. If the line search is exact, then in this case $s_k^T g_{k+1} = 0$ and therefore $\beta_k^A = \beta_k^{HS}$, for which the conjugacy condition holds.

Let us denote d_{k+1} and d_{k+1}^{HS} to be the search directions given by β_k^A and β_k^{HS}, respectively, namely, $d_{k+1} = -g_{k+1} + \beta_k^A s_k$ and $d_{k+1}^{HS} = -g_{k+1} + \beta_k^{HS} s_k$. Then it is easy to see that

$$g_{k+1}^T d_{k+1} = g_{k+1}^T d_{k+1}^{HS} - \beta_k^{HS} \frac{(s_k^T g_{k+1})^2}{y_k^T s_k}. \tag{19}$$

Assume that $g_{k+1}^T d_{k+1}^{HS} < 0$ and $\beta_k^{HS} \ge 0$. Then from (19) we also have that $g_{k+1}^T d_{k+1} < 0$. Thus if the direction generated by the HS method is descent, $\beta_k^{HS} \ge 0$, and if the line search is given by the Wolfe conditions, then the direction generated by β_k^A must also be a descent direction.

Observe that the direction

$$d_{k+1} = -g_{k+1} - \frac{(y_k^T g_{k+1})(s_k^T g_k)}{(y_k^T s_k)^2} s_k \tag{20}$$

is not a descent direction at every iteration. However, since $-s_k^T g_k / y_k^T s_k \geq 0$, when d_k is a descent direction, then if

$$\frac{y_k^T s_k}{s_k^T g_k}\|g_{k+1}\|^2 + \frac{(y_k^T g_{k+1})(s_k^T g_{k+1})}{y_k^T s_k} \leq 0,$$

it follows that $g_{k+1}^T d_{k+1} \leq 0$. On the other hand, observe that

$$\frac{y_k^T s_k}{s_k^T g_k}\|g_k\|^2 \leq 0$$

and tends to zero. Therefore, if

$$\frac{(y_k^T g_{k+1})(s_k^T g_{k+1})}{y_k^T s_k} \leq \frac{y_k^T s_k}{\left|s_k^T g_k\right|}\|g_{k+1}\|^2, \tag{21}$$

then $g_{k+1}^T d_{k+1} \leq 0$.

Considering the definitions of g_k, s_k and y_k we can present the following algorithm:

ACGA Algorithm

Step 1. Initialization. Select $x_0 \in R^n$ and the parameters $0 < \sigma_1 < \sigma_2 < 1$. 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

$$f(x_k + \alpha_k d_k) - f(x_k) \leq \sigma_1 \alpha_k g_k^T d_k, \tag{22}$$

$$\nabla f(x_k + \alpha_k d_k)^T d_k \geq \sigma_2 g_k^T d_k, \tag{23}$$

and update the variables $x_{k+1} = x_k + \alpha_k d_k$. Compute $f(x_{k+1})$, g_{k+1} and $s_k = x_{k+1} - x_k$, $y_k = g_{k+1} - g_k$.

Step 4. Direction computation. Compute $d = -g_{k+1} + \beta_k^A s_k$, where β_k^A is computed as in (17). If

$$g_{k+1}^T d \leq -10^{-3}\|d\|_2\|g_{k+1}\|_2, \tag{24}$$

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 (22) and (23). As we said above, the search direction $d = -g_{k+1} + \beta_k^A s_k$ can fail to be a descent direction. This fact motivated the introduction of the restart test (24). 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 simple restart criterion firstly introduced in [6]. Under reasonable assumptions, conditions (22), (23) and (24) are sufficient to prove the global convergence of the algorithm (see [22]).

Observe that if d_k is a descent direction, then under the Wolfe line search,

$$-\frac{s_k^T g_k}{y_k^T s_k} = -\frac{g_{k+1}^T s_k - y_k^T s_k}{y_k^T s_k} = -\frac{g_{k+1}^T s_k}{y_k^T s_k} + 1 > 0.$$

On the other hand, in these conditions, we can prove that

$$\left|\frac{g_{k+1}^T s_k}{y_k^T s_k}\right| \le \max\left\{\frac{\sigma_2}{1-\sigma_2}, 1\right\}$$

and since $g_{k+1}^T s_k < y_k^T s_k$, then always $g_{k+1}^T s_k$ tends to zero faster than $y_k^T s_k$. Therefore, if $-s_k^T g_k / y_k^T s_k \to 1$, we see that along the iterations $\left|\beta_k^A - \beta_k^{HS}\right| \to 0$. Therefore, the behavior of our algorithm comes close to that of Hestenes and Stiefel. A very attractive feature of the Hestenes and Stifel method is that independent of the line search, the conjugacy condition always holds. Consequently, if $-s_k^T g_k / y_k^T s_k$ in (18) is near 1, then the conjugacy condition holds approximately. Also, the Hestenes and Stiefel method is not susceptible to jamming.

The initial selection of the step length crucially affects the practical behavior of the algorithm. At every iteration $k \ge 1$ 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 [28]. It is also considered in the packages: SCG by Birgin and Martínez [6] and in SCALCG by Andrei [1-4].

It is worth saying that using the same arguments as in Theorem 1, but this time on the Hestenes and Stiefel parameter β_k^{HS}, i.e. considering $\beta_k = \beta_k^{HS} - \xi_k$, where

$$\xi_k = 2\|y_k\|^2 \frac{s_k^T g_{k+1}}{(y_k^T s_k)^2},$$

and using the general inequality $u^T v \le \frac{1}{2}(\|u\|^2 + \|v\|^2)$, we get exactly the conjugate gradient algorithm proposed by Hager and Zhang [17]:

$$\beta_k^{HZ} = \frac{1}{s_k^T y_k}\left(y_k - 2\frac{\|y_k\|^2}{s_k^T y_k} s_k\right)^T g_{k+1}. \tag{25}$$

Hager and Zhang obtained their computational scheme by deleting a term from the search direction for the memoryless quasi-Newton scheme of Perry [24] and Shanno [29, 30]. We see that formula (17) for β_k^A is very close to the Hager and Zhang computational scheme β_k^{HZ} (25), where the factor $2\|y_k\|^2 / y_k^T d_k$ in their scheme is replaced by $g_{k+1}^T y_k / y_k^T s_k$. The direction of Hager and Zhang satisfies the sufficient descent condition and $g_{k+1}^T d_{k+1}$ is bounded by $-(7/8)\|g_{k+1}\|^2$ [17].

At the same time, observe that (17) is very close to the Dai and Liao [9] computational scheme,

$$\beta_k^{DL} = \frac{1}{y_k^T s_k}\left(y_k - t s_k\right)^T g_{k+1}, \tag{26}$$

where the parameter t is replaced by $g_{k+1}^T y_k / y_k^T s_k$. The method (5), (17) can be viewed as an adaptive version of the Dai and Liao computational scheme, corresponding to $t = g_{k+1}^T y_k / y_k^T s_k \equiv \beta_k^{HS}$.

3. Convergence analysis

Assume that:

(i) *The level set* $S = \left\{x \in R^n : f(x) \le f(x_0)\right\}$ *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 $\Gamma \geq 0$ such that $\|\nabla f(x)\| \leq \Gamma$, for all $x \in L$. Dai *et al* [8] proved that for any conjugate gradient method with the strong Wolfe line search,

$$f(x_k + \alpha_k d_k) - f(x_k) \leq \sigma_1 \alpha_k g_k^T d_k,$$
$$\left|\nabla f(x_k + \alpha_k d_k)^T d_k\right| \leq -\sigma_2 g_k^T d_k.$$

the following general result holds.

Lemma 1. *Suppose that the assumptions (i) and (ii) holds and consider any conjugate gradient method (2) where* $d_{k+1} = -g_{k+1} + \beta_k d_k$ *is supposed to be a descent direction and* α_k *is selected by the strong Wolfe line search. If*

$$\sum_{k\geq 1} \frac{1}{\|d_k\|^2} = \infty, \tag{27}$$

then

$$\liminf_{k\to\infty} \|g_k\| = 0. \tag{28}$$

For uniformly convex functions we can prove that the norm of the direction d_{k+1} generated by (5) and (18) is bounded above. Therefore, by Lemma 1 we can prove the following result.

Theorem 2. *Suppose that the assumptions (i) and (ii) holds and consider the method (2) and (5), where* d_{k+1} *is a descent direction with* β_k^A *given by (18), and* α_k *is obtained by the strong Wolfe line search. Suppose that* f *is a uniformly convex function on* S*, i.e. there exists a constant* $\mu > 0$ *such that*

$$(\nabla f(x) - \nabla f(y))^T (x - y) \geq \mu \|x - y\|^2 \tag{29}$$

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

$$\lim_{k\to\infty} g_k = 0. \tag{30}$$

Proof. From (29) it follows that $y_k^T s_k \geq \mu \|s_k\|^2$. Since d_k is a descent direction, it follows that $g_{k+1}^T s_k = y_k^T s_k + g_k^T s_k < y_k^T s_k$. By Wolfe condition (23) we have:

$$y_k^T s_k = (g_{k+1} - g_k)^T s_k \geq (\sigma_2 - 1) g_k^T s_k = -(1 - \sigma_2) g_k^T s_k. \tag{31}$$

From (31) we get

$$1 - \frac{g_{k+1}^T s_k}{y_k^T s_k} = -\frac{g_k^T s_k}{y_k^T s_k} \leq \frac{1}{1 - \sigma_2}. \tag{32}$$

Therefore,

$$\left|\beta_k^A\right| \leq \frac{\|g_{k+1}\| L \|s_k\|}{\mu \|s_k\|^2} \frac{1}{1 - \sigma_2} \leq \frac{\Gamma L}{\mu(1 - \sigma_2)} \frac{1}{\|s_k\|}. \tag{33}$$

Hence

$$\|d_{k+1}\| \leq \|g_{k+1}\| + \left|\beta_k^A\right| \|s_k\| \leq \left(1 + \frac{L}{\mu(1 - \sigma_2)}\right) \Gamma, \tag{34}$$

i.e. (27) is true. Therefore, by Lemma 1 we have (28), which for uniformly convex functions is equivalent to (30). ■

Although we proved the global convergence of the (5) and (17) computational scheme when f is strongly convex, our analysis breaks down for a general nonlinear function since β_k^A can be negative and the direction d_k generated by the algorithm does not satisfy a sufficient descent condition. For general nonlinear functions we can assume that for all sufficiently large k, $1-s_k^T g_{k+1}/y_k^T s_k \cong 1$. Therefore, the convergence properties of our algorithm should be similar to the convergence properties of Hestenes and Stiefel method. In particular, by the Powell's example [27], the ACGA method with an exact line search may not converge for a general nonlinear function. However, since $s_k^T g_{k+1}$ tends to zero faster than $y_k^T s_k$, when d_k is a descent direction, it follows that the conjugacy condition holds approximately, and consequently the ACGA method is not susceptible to jamming.

Considering $\beta_k^{HS+} = \max\left\{\beta_k^{HS}, 0\right\}$ then a similar analysis as that given by Dai and Liao [9] can be done for a modification of β_k^A as

$$\beta_k^{A+} = \max\left\{\frac{y_k^T g_{k+1}}{y_k^T s_k}, 0\right\} - \beta_k^{HS+}\frac{g_{k+1}^T s_k}{y_k^T s_k} = \beta_k^{HS+}\left(\frac{-g_k^T s_k}{y_k^T s_k}\right),$$

proving that such a modification is globally convergent for general functions. However, the resulting iterates may differ significantly from those of (5) and (17), and convergence speed may be severely reduced, especially when the function f is quadratic. This is the reason we selected (5) and (17) computational scheme for which we are not able to prove its global convergence, but instead we have computational evidence that it has better numerical performances.

4. Numerical results and comparisons

In this section we present the computational performance of a Fortran implementation of the ACGA algorithm on a set of 750 unconstrained optimization test problems. The test problems are the unconstrained problems in the CUTE [7] library, along with other large-scale optimization problems we presented in [5]. We selected 75 large-scale unconstrained optimization problems in extended or generalized form. For each function we have considered ten numerical experiments with the number of variables: $n = 1000, 2000, \ldots, 10000$.

All algorithms implement the Wolfe line search conditions with $\sigma_1 = 0.0001$ and $\sigma_2 = 0.9$, and the same stopping criterion $\|g_k\|_\infty \le 10^{-6}$, where $\|.\|_\infty$ is the maximum absolute component of a vector.

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

$$\left| f_i^{ALG1} - f_i^{ALG2} \right| < 10^{-3} \tag{35}$$

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. In this numerical study we declare that a method solved a particular problem if the final point obtained has the lowest functional value among the tested methods (up to 10^{-3} tolerance as it is specified in (35)). Clearly, this criterion is acceptable for users that are interested in minimizing functions and not finding critical points.

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 the performance of ACGA algorithm to the Dai and Yuan conjugate gradient algorithms. Dai and Yuan [12] studied the hybrid conjugate gradient algorithms and proposed the following two hybrid methods:

$$\beta_k^{hDY} = \max\left\{-\frac{1-\sigma_2}{1+\sigma_2}\beta_k^{DY}, \min\left\{\beta_k^{HS}, \beta_k^{DY}\right\}\right\}, \tag{36}$$

and

$$\beta_k^{hDYz} = \max\left\{0, \min\left\{\beta_k^{HS}, \beta_k^{DY}\right\}\right\}, \tag{37}$$

showing their global convergence when the Lipschitz assumption holds and the standard Wolfe line search is considered. The numerical experiments of Dai and Ni [10] show that the second hybrid method (hDYz) gave the best results, performing better that the Polak-Ribière [25] and Polyak [26] plus method. Tables 1-3 present the performances of these algorithms subject to the minimum number of iterations (#iter), the minimum number of function and its gradient evaluations (#fg) and the minimum cpu time (CPU), respectively.

Table 1. Performance of ACGA versus Dai-Yuan. 721 problems.

	ACGA	DY	=
# iter	382	111	228
# fg	417	201	103
CPU	477	162	82

Table 2. Performance of ACGA versus hDY (hybrid Dai-Yuan). 695 problems.

	ACGA	hDY	=
# iter	334	171	190
# fg	363	215	117
CPU	382	189	124

Table 3. Performance of ACGA versus hDYz (hybrid Dai-Yuan zero). 689 problems.

	ACGA	hDYz	=
# iter	248	236	205
# fg	310	263	116
CPU	325	255	109

When comparing ACGA and DY algorithms (Table 1), subject to the number of iterations, ACGA was better in 382 problems (i.e. it achieved the minimum number of iterations in 382 problems), DY was better in 111 problems, and they had the same number of iterations in 228 problems, etc. Out of 750 problems, only for 721 problems the criterion (35) holds. From these Tables we see that, at least for this set of 750 problems, comparing with Dai and Yuan conjugate gradient algorithms, the top performer is ACGA. Observe that the hybrid variants hDY and hDYz are better that the original conjugate gradient scheme of Dai and Yuan. The results of Table 3 seem to be consistent with the numerical experiments reported by Dai and Ni [10].

The second set of numerical experiments refers to the comparison of ACGA with 15 conjugate gradient algorithms, where in these algorithms the search direction d_{k+1} is computed as $d_{k+1} = -g_{k+1} + \beta_k d_k$ where the parameter β_k is selected as:

$\beta_k^{HS} = \frac{g_{k+1}^T y_k}{d_k^T y_k}$	The original linear conjugate gradient algorithm by Hestenes and Stiefel [19].
$\beta_k^{FR} = \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k}$	The first nonlinear conjugate gradient algorithm, proposed by Fletcher and Reeves [14].

$\beta_k^{PRP} = \frac{g_{k+1}^T y_k}{g_k^T g_k}$	Proposed by Polak and Ribière [25] and Polyak [26].
$\beta_k^{PRP+} = max\left\{0, \frac{g_{k+1}^T y_k}{g_k^T g_k}\right\}$	Proposed by Powell [27], and analyzed by Gilbert and Nocedal [15].
$\beta_k^{GN} = \max\left\{-\beta_k^{FR}, \min\left\{\beta_k^{PRP}, \beta_k^{FR}\right\}\right\}$	Proposed by Gilbert and Nocedal [15]
$\beta_k^{CD} = \frac{g_{k+1}^T g_{k+1}}{-d_k^T g_k}$	Proposed by Fletcher [13] as a Conjugate Descent method
$\beta_k^{LS} = \frac{g_{k+1}^T y_k}{-d_k^T g_k}$	Proposed by Liu and Storey [21].
$\beta_k^{LS-CD} = \max\left\{0, \min\left\{\beta_k^{LS}, \beta_k^{CD}\right\}\right\}$	Hybrid Liu and Storey – Conjugate Descent
$\beta_k^{Hu-Storey} = \max\left\{0, \min\left\{\beta_k^{PRP}, \beta_k^{FR}\right\}\right\}$	Proposed by Hu and Storey [20]
$\beta_k^{TA-S} = \begin{cases} \beta_k^{PRP} & \text{if } 0 \le \beta_k^{\text{PRP}} \le \beta_k^{FR}, \\ \beta_k^{FR} & \text{otherwise} \end{cases}$	Proposed by Touati-Ahmed and Storey [31]
$\beta_k^{DL} = \frac{g_{k+1}^T (y_k - t s_k)}{d_k^T y_k}, t > 0$	Proposed by Dai and Liao [9],

or as $d_{k+1} = -\theta_{k+1} g_{k+1} + \beta_k s_k$, where the parameter θ_{k+1} is a scalar approximation of the inverse Hessian (the inverse of the Rayleigh quotient) of the function f :

$$\theta_{k+1} = \frac{s_k^T s_k}{y_k^T s_k} \tag{38}$$

and β_k is selected as:

$\beta_k^{BM} = \frac{g_{k+1}^T (\theta_k y_k - s_k)}{y_k^T s_k}$	Scaled Perry. Suggested by Birgin and Martínez [6] and Andrei [1-4].
$\beta_k^{BM+} = max\left\{0, \frac{\theta_k g_{k+1}^T y_k}{y_k^T s_k}\right\} - \frac{g_{k+1}^T s_k}{y_k^T s_k}$	Scaled Perry+. Suggested by Birgin and Martínez [6].
$\beta_k^{sPRP} = \frac{\theta_k g_{k+1}^T y_k}{\alpha_k \theta_{k-1} g_k^T g_k}$	Scaled Polak-Ribière-Polyak. Suggested by Birgin and Martínez [6] and Andrei [1-4].
$\beta_k^{sFR} = \frac{\theta_k g_{k+1}^T g_{k+1}}{\alpha_k \theta_{k-1} g_k^T g_k}$	Scaled Fletcher-Reeves. Suggested by Birgin and Martínez [6] and Andrei [1-4].

Tables 4-18 show the number of problems, out of 750, for which ACGA versus these conjugate gradient algorithms achieved the minimum number of iterations (#iter), the minimum number of function evaluations (#fg) and the minimum cpu time in seconds (CPU), subject of (35), respectively.

Table 4. Performance of ACGA versus Hestenes-Stiefel. 702 problems.

	ACGA	HS	=
# iter	285	194	223
# fg	310	233	159
CPU	346	225	131

Table 5. Performance of ACGA versus Fletcher-Reeves. 707 problems.

	ACGA	FR	=
# iter	429	85	193
# fg	457	149	101
CPU	488	145	74

Table 6. Performance of ACGA versus Polak-Ribière-Poliak. 713 problems.

	ACGA	PRP	=
# iter	314	161	238
# fg	378	189	146
CPU	388	193	132

Table 7. Performance of ACGA versus Polak-Ribière-Poliak(+). 704 problems.

	ACGA	PRP+	=
# iter	277	213	214
# fg	335	232	137
CPU	359	228	117

Table 8. Performance of ACGA versus Gilbert-Nocedal. 707 problems.

	ACGA	GN	=
# iter	351	172	184
# fg	386	212	109
CPU	413	199	95

Table 9. Performance of ACGA versus Conjugate Descent (Fletcher). 712 problems.

	ACGA	CD	=
# iter	400	97	215
# fg	437	170	105
CPU	450	178	84

Table 10. Performance of ACGA versus Liu-Storey. 692 problems.

	ACGA	LS	=
# iter	305	164	223
# fg	350	202	140
CPU	379	206	107

Table 11. Performance of ACGA versus hybrid Liu-Storey & Conjugate-Descent. 705 problems.

	ACGA	hLS-CD	=
# iter	303	215	187
# fg	337	248	120
CPU	372	234	99

Table 12. Performance of ACGA versus Hu-Storey. 709 problems.

	ACGA	Hu-Storey	=
# iter	329	199	181
# fg	373	229	107
CPU	394	220	95

Table 13. Performance of ACGA versus Touati-Ahmed and Storey. 701 problems.

	ACGA	TA-S	=
# iter	378	147	176
# fg	414	187	100
CPU	428	185	88

Table 14. Performance of ACGA versus Dai-Liao(t=1). 697 problems.

	ACGA	DL(t=1)	=
# iter	270	194	233
# fg	323	231	143
CPU	336	227	134

Table 15. Performance of ACGA versus SCG by Birgin-Martínez. 707 problems.

	ACGA	SCG	=
# iter	249	248	210
# fg	295	285	127
CPU	325	263	119

Table 16. Performance of ACGA versus SCG plus by Birgin-Martínez. 697 problems.

	ACGA	SCG+	=
# iter	263	248	186
# fg	310	271	116
CPU	341	255	101

Table 17. Performance of ACGA versus scaled Polak-Ribière-Poliak. 694 problems.

	ACGA	sPRP	=
# iter	335	149	210
# fg	375	179	140
CPU	402	188	104

Table 18. Performance of ACGA versus scaled Fletcher-Reeves. 699 problems.

	ACGA	sFR	=
# iter	428	84	187
# fg	447	156	96
CPU	474	146	79

From Tables above we see that ACGA is top performer. Since these codes use the same Wolfe line search and the same stopping criterion they differ in their choice of the search direction. Hence, among these conjugate gradient algorithms, ACGA appears to generate the best search direction, on average.

Concerning the cpu time, from Table 14, we see that the closest to ACGA is Dai-Liao ($t = 1$) algorithm. Both ACGA and DL($t = 1$) achieved the minimum cpu time for 134 problems. Dai and Liao algorithm is also a modification of the Hestenes and Stiefel's. For an exact line search, g_{k+1} is orthogonal to s_k. Hence, for exact line search the DL method reduces to the HS method. From Table 4 we see that HS is also close to ACGA algorithm. Both ACGA and HS achieved the same CPU time for 131 problems. From Table 6 close to ACGA is also PRP. For an exact line search, $\beta_k^{PRP} = \beta_k^{HS}$. Therefore, these methods have similar convergence properties. Both HS, PRP and ACGA methods have a built-in restart feature that addresses the jamming phenomenon. When the step $s_k = x_{k+1} - x_k$ is small, then $y_k = g_{k+1} - g_k$ tends to zero. Hence, β_k from HS, PRP and ACGA becomes small and the

new search direction d_{k+1} is essentially the steepest descent direction $-g_{k+1}$. Therefore, HS, PRP and ACGA methods automatically adjust the parameter β_k to avoid jamming, the performance of these methods are better than the performance of some other conjugate gradient methods.

In the third set of numerical experiments we compare ACGA to CG_DESCENT by Hager and Zhang [17]. The CG_DESCENT code, authored by Hager and Zhang, contains the variant CG_DESCENT(w) implementing the Wolfe line search and the variant CG_DESCENT(aw) implementing an approximate Wolfe line search. The computational scheme implemented in CG_DESCENT is a modification of the Hestenes and Stiefel method which satisfies the sufficient descent condition. However, in this method the conjugacy condition holds approximately. There are two main points associated to CG_DESCENT. Firstly, the scalar products are implemented using the loop unrolling of depth 5. This is efficient for large-scale problems (over 10^6 variables). Secondly, the Wolfe line search is implemented using a very fine numerical interpretation of the first Wolfe condition (22). The Wolfe conditions implemented in ACGA and CG_DESCENT(w) can compute a solution with an accuracy on the order of the square root of the machine epsilon. In contrast, the approximate Wolfe line search implemented in CG_DESCENT(aw) can compute a solution with an accuracy of the order of machine epsilon.

Tables 19 and 20 show the number of problems solved by these algorithms in the minimum number of iterations, the minimum number of function evaluations and the minimum cpu time, respectively.

Table 19. Performance of ACGA versus CG_DESCENT(w). 700 problems.

	ACGA	CG_DESCENT(w)	=
# iter	308	309	83
# fg	440	223	37
CPU	354	275	71

Table 20. Performance of ACGA versus CG_DESCENT(aw). 700 problems.

	ACGA	CG_DESCENT(aw)	=
# iter	316	301	83
# fg	424	233	43
CPU	351	286	63

5. Conclusion

We have presented a new conjugate gradient algorithm for solving unconstrained optimization problems. The parameter β_k^A is a positive modification of the Hestenes and Stiefel computational scheme. For uniformly convex functions the algorithm with strong Wolfe line search is global convergent. For general nonlinear functions, the convergence properties of ACGA are similar to the convergence properties of Hestenes and Stiefel's. We present computational evidence that the numerical performance of ACGA algorithm was higher than those of the Hestenes and Stiefel conjugate gradient algorithm, as well as of some other conjugate gradient algorithms including the recent CG_DESCENT by Hager and Zhang, for a set consisting of 750 problems.

References

[1] N. Andrei, *Conjugate gradient algorithms for large scale unconstrained optimization.* ICI Technical Report, January 12, 2005.

[2] N. Andrei, *Scaled conjugate gradient algorithms for unconstrained optimization*. Computational Optimization and Applications, vol.38, No.3, (2007), pp. 401-416.

[3] N. Andrei, *Scaled memoryless BFGS preconditioned conjugate gradient algorithm for unconstrained optimization.* Optimization Methods and Software, vol.22, No.4, August 2007, 561-571.
[4] N. Andrei, *A scaled BFGS preconditioned conjugate gradient algorithm for unconstrained optimization*. Applied Mathematics Letters, 20 (2007) 645-650.
[5] N. Andrei, *An unconstrained optimization test functions collection.* Advanced Modeling and Optimization. An electronic international journal, 10 (2008), 147-161.
[6] E. Birgin and J.M. Martínez, *A spectral conjugate gradient method for unconstrained optimization*, Applied Math. and Optimization, 43, pp.117-128, 2001.
[7] I. Bongartz, A.R. Conn, N.I.M. Gould and P.L. Toint, *CUTE: constrained and unconstrained testing environments*, ACM Trans. Math. Software, 21, pp.123-160, 1995.
[8] Y.H. Dai, Han, J.Y., Liu, G.H., Sun, D.F., Yin, .X. and Yuan, Y., *Convergence properties of nonlinear conjugate gradient methods*. SIAM Journal on Optimization 10 (1999), 348-358.
[9] Y.H. Dai and L.Z. Liao, *New conjugacy conditions and related nonlinear conjugate gradient methods.* Appl. Math. Optim., 43 (2001), pp. 87-101.
[10] Y.H. Dai and Q. Ni, *Testing different conjugate gradient methods for large-scale unconstrained optimization,* J. Comput. Math., 21 (2003), pp.311-320.
[11] Y.H. Dai and Y, Yuan, *A nonlinear conjugate gradient method with a strong global convergence property,* SIAM J. Optim., 10 (1999), pp.177-182.
[12] Y.H. Dai and Y. Yuan, *An efficient hybrid conjugate gradient method for unconstrained optimization.* Annals of Operations Research, 103 (2001), pp.33-47.
[13] R. Fletcher, *Practical Methods of Optimization vol.1: Unconstrained Optimization.* Jhon Wiley & Sons, New York, 1987.
[14] R. Fletcher and C.M. Reeves, *Function minimization by conjugate gradients.* Computer Journal, 7 (1964), pp.149-154.
[15] J.C. Gilbert and J. Nocedal, *Global convergence properties of conjugate gradient methods for optimization.* SIAM J. Optim., 2 (1992), pp.21-42.
[16] G.H. Golub and D.P. O'Leary, *Some history of the conjugate gradient and Lanczos algorithms: 1948-1976*. SIAM Review, 31 (1976), pp.50-100.
[17] W.W. Hager and H. Zhang, *A new conjugate gradient method with guaranteed descent and an efficient line search*, SIAM Journal on Optimization, 16 (2005), 170-192.
[18] W.W. Hager and H. Zhang, *A survey of nonlinear conjugate gradient methods*. Pacific journal of Optimization, 2 (2006), pp.35-58.
[19] M.R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Research Nat. Bur. Standards Sec. B. 48, pp. 409-436, 1952.
[20] Y.F. Hu, and C. Storey, *Global convergence result for conjugate gradient methods*. JOTA, 71, (1991), pp.399-405.
[21] Y. Liu, and C. Storey, *Efficient generalized conjugate gradient algorithms, Part 1: Theory.* JOTA, 69 (1991), pp.129-137.
[22] J. Nocedal, *Theory of algorithms for unconstrained optimization.* Acta Numerica 1992, (1992), 199-242.
[23] D.P. O'Leary, *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.
[24] J.M. Perry, *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.
[25] E. Polak and G. Ribière, *Note sur la convergence de méthodes de directions conjuguée,* Revue Francaise Informat. Recherche Opérationnelle, 3e Année 16 (1969), pp.35-43.
[26] B.T. Polyak, *The conjugate gradient method in extreme problems,*USSR Comp. Math. Math. Phys., 9 (1969), pp.94-112.

[27] M.J.D. Powell, *Nonconvex minimization calculations and the conjugate gradient method,* Numerical Analysis (Dundee, 1983), Lecture Notes in Mathematics, vol.1066, Springer Verlag, Berlin, 1984, pp.122-141.
[28] D.F. Shanno and K.H. Phua, *Algorithm 500, Minimization of unconstrained multivariate functions*, ACM Trans. on Math. Soft., 2, pp.87-94, 1976.
[29] D.F. Shanno, *On the convergence of a new conjugate gradient algorithm,*SIAM J. Numer. Anal., 15 (1978), pp.1247-1257.
[30] D.F. Shanno, *Conjugate gradient methods with inexact searches.* Math. Oper. Res., 3 (1978), pp.244-256.
[31] D. Touati-Ahmed and C. Storey, *Efficient hybrid conjugate gradient techniques,* Journal of Optimization Theory and Applications, 64 (1990), pp. 379-397.
[32] P. Wolfe, *Convergence conditions for ascent methods.* SIAM Review 11 (1969) 226-235.
[33] P. Wolfe, *Convergence conditions for ascent methods, (II): some corrections.* SIAM Review 13 (1971) 185-188.
[34] G. Yu, L. Guan and W. Chen, *Spectral conjugate gradient methods with sufficient descent property for large-scale unconstrained optimization.* Optimization Methods and Software, vol.23, No.2, (2008), pp. 275-293.

July 23, 2008