Another Nonlinear Conjugate Gradient Algorithm with Sufficient Descent Condition for Unconstrained Optimization

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

Abstract. A nonlinear conjugate gradient algorithm with conjugacy condition and sufficient descent condition for unconstrained optimization is proposed. Using the exact line search, the algorithm reduces to a version of the Dai and Yuan conjugate gradient computational scheme. For inexact line search the algorithm satisfies both the sufficient descent and the conjugacy conditions. A global convergence result is proved when the Wolfe line search conditions are used. 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 [22]. 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 new nonlinear conjugate gradient algorithm that produces a descent direction at every iteration and converges globally to the solution provided that the line search satisfies the Wolfe conditions. The algorithm is a modification of the Dai and Yuan [11] conjugate gradient algorithm satisfying both the sufficient descent condition and the conjugacy condition at every iteration. Under exact line search the algorithm reduces to the Dai and Yuan computational scheme. At the same time the 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]. The algorithm has a built-in restart feature that addresses to the jamming phenomenon.

The structure of the paper is as follows. In section 2 we present the new conjugate gradient algorithm and prove that it generates descent directions satisfying both the sufficient descent condition and the conjugacy condition. Section 3 is devoted to the convergence analysis for both the uniformly convex functions and general nonlinear functions. It is shown that under very common assumptions the proposed algorithm is globally convergent. 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. 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. A conjugate gradient algorithm with sufficient descent condition

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 [31, 32], 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 in order to satisfy both the sufficient descent condition and the conjugacy condition in the frame of conjugate gradient algorithms. In order to satisfy the sufficient descent condition, in our algorithm the direction is computed 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 parameter which follows to be determined.

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 $g_0^T d_0 = -\left\|g_0\right\|^2$, which satisfy (6). Multiplying (5) by g_{k+1}^T, we have

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

But

$$\frac{(g_{k+1}^T g_{k+1})(g_{k+1}^T s_k)}{y_k^T s_k} = \frac{\left[(y_k^T s_k) g_{k+1} / \sqrt{2\delta_k}\right]^T \left[\sqrt{2\delta_k}\,(g_{k+1}^T s_k) g_{k+1}\right]}{(y_k^T s_k)^2}$$

$$\leq \frac{\frac{1}{2}\left[\frac{1}{2\delta_k}(y_k^T s_k)^2\|g_{k+1}\|^2 + 2\delta_k (g_{k+1}^T s_k)^2\|g_{k+1}\|^2\right]}{(y_k^T s_k)^2}$$

$$= \frac{1}{4\delta_k}\|g_{k+1}\|^2 + \delta_k \frac{(g_{k+1}^T s_k)^2\|g_{k+1}\|^2}{(y_k^T s_k)^2}. \qquad (9)$$

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, $1/(4\delta_k) \leq 1$, and the line search satisfies the Wolfe conditions, then for all k the search direction (5) and (6) satisfy the sufficient descent condition. Note that in (7) we bound $g_{k+1}^T d_{k+1}$ by $-(1-1/4\delta_k)\|g_{k+1}\|^2$, while for scheme of Dai and Yuan only the non-negativity of $g_{k+1}^T d_{k+1}$ is established.

In [7] 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 $\gamma_1 \leq \|g_k\| \leq \gamma_2$ for all k, then for any $p \in (0,1)$, there exists a constant $c > 0$ such that the sufficient descent condition $g_i^T d_i \leq -c\|g_i\|^2$ holds for at least $\lfloor pk \rfloor$ indices $i \in [0,k]$, where $\lfloor j \rfloor$ denotes the largest integer $\leq j$. In our algorithm the famous parameter β_k is selected in such a manner that the sufficient descent condition is satisfied at every iteration.

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.

In order to determine δ_k, observe that using (6) in (5) we get the following direction:

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

which can be written as

$$d_{k+1} = -Q_{k+1} g_{k+1}, \qquad (10)$$

where the matrix Q_{k+1} is:

$$Q_{k+1} = I - \frac{s_k g_{k+1}^T}{y_k^T s_k} + \delta_k \frac{\|g_{k+1}\|^2}{(y_k^T s_k)^2}(s_k s_k^T). \qquad (11)$$

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

$$\bar{Q}_{k+1} = I - \frac{s_k g_{k+1}^T + g_{k+1} s_k^T}{y_k^T s_k} + \delta_k \frac{\|g_{k+1}\|^2}{(y_k^T s_k)^2}(s_k s_k^T), \qquad (12)$$

we can consider the direction

$$d_{k+1} = -\bar{Q}_{k+1} g_{k+1}. \qquad (13)$$

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

$$y_k^T \bar{Q}_{k+1} g_{k+1} = 0\,, \tag{14}$$

after some algebra it follows that

$$\delta_k = \frac{y_k^T s_k}{g_{k+1}^T s_k} + \frac{g_{k+1}^T y_k}{\|g_{k+1}\|^2} - \frac{(g_{k+1}^T y_k)(y_k^T s_k)}{\|g_{k+1}\|^2 (g_{k+1}^T s_k)}\,. \tag{15}$$

Therefore, using (15) in (6) we get

$$\beta_k^A = \frac{1}{y_k^T s_k}\left(y_k - \frac{g_{k+1}^T y_k}{y_k^T s_k} s_k \right)^T g_{k+1}\,. \tag{16}$$

Using the same arguments as in Theorem 1, but this time on the Hestenes and Stiefel parameter β_k^{HS} [19] where

$$\beta_k^{HS} = \frac{g_{k+1}^T y_k}{d_k^T y_k}\,, \tag{17}$$

we get exactly the conjugate gradient algorithm proposed by Hager and Zhang [17], where

$$\beta_k^{HZ} = \frac{1}{d_k^T y_k}\left(y_k - 2\frac{\|y_k\|^2}{d_k^T y_k} d_k \right)^T g_{k+1}\,. \tag{18}$$

It is worth saying that Hager and Zhang obtained their computational scheme by deleting a term from the search direction for the memoryless quasi-Newton scheme of Perry [23] and Shanno [28, 29]. We see that formula (16) for β_k^A is very close to the Hager and Zhang computational scheme β_k^{HZ} (18), 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 (16) is very close to the Dai and Liao [9] computational scheme,

$$\beta_k^{DL} = \frac{1}{y_k^T s_k}\left(y_k - t s_k \right)^T g_{k+1}\,, \tag{19}$$

where the parameter t is replaced by $g_{k+1}^T y_k / y_k^T s_k$. The method (5), (16) can be viewed as an adaptive version of the Dai and Liao computational scheme, corresponding to $t = g_{k+1}^T y_k / y_k^T s_k$.

Considering the definitions of g_k, s_k and y_k we present the following Conjugate Gradient Algorithm with Conjugacy and Sufficient Descent conditions:

ACGSD 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 $\|g_k\|_\infty \le 10^{-6}$, then stop, else set $k = k + 1$.

Step 3. Line search. Compute α_k satisfying the Wolfe line search conditions

$$f(x_k + \alpha_k d_k) - f(x_k) \le \sigma_1 \alpha_k g_k^T d_k\,, \tag{20}$$

$$\nabla f(x_k + \alpha_k d_k)^T d_k \ge \sigma_2 g_k^T d_k\,, \tag{21}$$

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 (16). If

$$g_{k+1}^T d \le -10^{-3}\|d\|_2 \left\|g_{k+1}\right\|_2, \tag{22}$$

then define $d_{k+1} = d$, otherwise set $d_{k+1} = -g_{k+1}$. Compute the initial guess $\alpha_k = \alpha_{k-1}\left\|d_{k-1}\right\| / \left\|d_k\right\|$, 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 (20) and (21). 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 both the sufficient descent and the conjugacy conditions. Under reasonable assumptions, conditions (20), (21) and (22) are sufficient to prove the global convergence of the algorithm. We consider this aspect in the next section.

The initial selection of the step length crucially affects the practical behaviour of the algorithm. At every iteration $k \ge 1$ the starting guess for the step α_k in the line search is computed as $\alpha_{k-1}\left\|d_{k-1}\right\|_2 / \left\|d_k\right\|_2$. This selection, was considered for the first time by Shanno and Phua in CONMIN [27]. It is also considered in the packages: SCG by Birgin and Martínez [5] and in SCALCG by Andrei [2-4].

3. Convergence analysis

Throughout this section we assume that:

(i) *The level set* $L = \left\{x \in R^n : f(x) \le f(x_0)\right\}$ *is bounded.*

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

Under these assumptions on f, there exists a constant $\Gamma \ge 0$ such that $\left\|\nabla f(x)\right\| \le \Gamma$, for all $x \in L$.

Dai *et al* [8] proved that for any conjugate gradient method with the strong Wolfe line search,

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

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 a descent direction and* α_k *is selected by the strong Wolfe line search. If*

$$\sum_{k\ge 1} \frac{1}{\left\|d_k\right\|^2} = \infty, \tag{23}$$

then

$$\liminf_{k\to\infty} \left\|g_k\right\| = 0. \tag{24}$$

For uniformly convex functions we can prove that the norm of the direction d_{k+1} generated by (5) and (16) 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 (16), and* α_k *is obtained by the strong Wolfe line search. If there exists a constant* $\mu > 0$ *such that*

$$(\nabla f(x) - \nabla f(y))^T (x - y) \geq \mu \|x - y\|^2 \tag{25}$$

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

$$\lim_{k \to \infty} g_k = 0. \tag{26}$$

Proof. From (25) it follows that f is a uniformly convex function on L and $y_k^T s_k \geq \mu \|s_k\|^2$. 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 (21) we have:

$$y_k^T s_k = (g_{k+1} - g_k)^T s_k \geq (\sigma_2 - 1) g_k^T s_k = -(1 - \sigma_2) g_k^T s_k. \tag{27}$$

From (16) we can write

$$\beta_k^A = \frac{g_{k+1}^T y_k}{y_k^T s_k} \left(1 - \frac{g_{k+1}^T s_k}{y_k^T s_k} \right). \tag{28}$$

But, from (27) we get

$$1 - \frac{g_{k+1}^T s_k}{y_k^T s_k} = -\frac{g_k^T s_k}{y_k^T s_k} \leq \frac{1}{1 - \sigma_2}. \tag{29}$$

Therefore,

$$\left| \beta_k^A \right| \leq \frac{\|g_{k+1}\| L \|s_k\|}{\mu \|s_k\|^2} \frac{1}{1 - \sigma_2} \leq \frac{\Gamma L}{\mu (1 - \sigma_2)} \frac{1}{\|s_k\|}. \tag{30}$$

Hence

$$\|d_{k+1}\| \leq \|g_{k+1}\| + \left| \beta_k^A \right| \|s_k\| \leq \left(1 + \frac{L}{\mu (1 - \sigma_2)} \right) \Gamma, \tag{31}$$

i.e. (23) is true. Therefore, by Lemma 1 we have (24), which for uniformly convex functions is equivalent to (26). ■

For general nonlinear functions the convergence analysis of our algorithm exploits insights developed by Gilbert and Nocedal [15], Dai and Liao [9] and that of Hager and Zhang [17]. Global convergence proof of ACGSD algorithm is based on the Zoutendijk condition combined with the analysis showing that the sufficient descent condition holds and $\|d_k\|$ is bounded. Suppose that the level set L is bounded and the function f is bounded from below.

Lemma 2. *Assume that* d_k *is a descent direction and* ∇f *satisfies the Lipschitz condition* $\|\nabla f(x) - \nabla f(x_k)\| \leq L \|x - x_k\|$ *for all* x *on the line segment connecting* x_k *and* x_{k+1}*, where* L *is a constant. If the line search satisfies the second Wolfe condition (21), then*

$$\alpha_k \geq \frac{1 - \sigma_2}{L} \frac{\left| g_k^T d_k \right|}{\|d_k\|^2}. \tag{32}$$

Proof. Subtracting $g_k^T d_k$ from both sides of (21) and using the Lipschitz condition we have

$$(\sigma_2 - 1) g_k^T d_k \leq (g_{k+1} - g_k)^T d_k \leq L \alpha_k \|d_k\|^2. \tag{33}$$

Since d_k is a descent direction and $\sigma_2 < 1$, (32) follows immediately from (33). ■

Theorem 3. *Suppose that for all* $k \geq 0$ *there exists a positive constant* ω, *such that* $1-1/(4\delta_k) \geq \omega > 0$ *and there exists the constants* γ *and* Γ, *such that* $\gamma \leq \|g_k\| \leq \Gamma$. *If the level set* $L = \{x \in R^n : f(x) \leq f(x_0)\}$ *is bounded and the Lipschitz condition* $\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$ *holds for all* $x, y \in L$, *then for the computational scheme (2), (5) and (16) with a line search satisfying the Wolfe conditions (20) and (21) , either* $g_k = 0$ *for some* k *or*

$$\liminf_{k\to\infty} \|g_k\| = 0. \tag{34}$$

Proof. Suppose that $g_k \neq 0$ for all k. Since $g_k \neq 0$ it follows that $\gamma > 0$. By the Wolfe condition (21) we have:

$$y_k^T s_k = (g_{k+1} - g_k)^T s_k \geq (\sigma_2 - 1) g_k^T s_k = -(1-\sigma_2) g_k^T s_k. \tag{35}$$

By Theorem 1, and the assumption $1-1/(4\delta_k) \geq \omega$,

$$g_k^T d_k \leq -\left(1 - \frac{1}{4\delta_{k-1}}\right)\|g_k\|^2 \leq -\omega\|g_k\|^2. \tag{36}$$

Therefore,

$$-g_k^T d_k \geq \omega\|g_k\|^2. \tag{37}$$

Combining (35) with (37) we get

$$y_k^T s_k \geq (1-\sigma_2)\omega\alpha_k\gamma^2. \tag{38}$$

Observe that

$$g_{k+1}^T s_k = y_k^T s_k + g_k^T s_k < y_k^T s_k. \tag{39}$$

From (21) we have

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

Since $\sigma_2 < 1,$ we obtain

$$g_{k+1}^T s_k \geq \frac{-\sigma_2}{1-\sigma_2} y_k^T s_k. \tag{40}$$

Combining this lower bound for $g_{k+1}^T s_k$ with the upper bound (39) yields

$$\left|\frac{g_{k+1}^T s_k}{y_k^T s_k}\right| \leq \max\left\{1, \frac{\sigma_2}{1-\sigma_2}\right\}. \tag{41}$$

On the other hand $\|y_k\| = \|g_{k+1} - g_k\| \leq L\|s_k\|$. Hence

$$\left|g_{k+1}^T y_k\right| \leq \|g_{k+1}\|\|y_k\| \leq \Gamma L\|s_k\|. \tag{42}$$

With these, using (38) and (42) in (16) we get:

$$\begin{aligned}\left|\beta_k^A\right| &\leq \frac{1}{\left|y_k^T s_k\right|}\left[\left|g_{k+1}^T y_k\right| + \frac{\left|g_{k+1}^T y_k\right|\left|g_{k+1}^T s_k\right|}{\left|y_k^T s_k\right|}\right] = \frac{\left|g_{k+1}^T y_k\right|}{\left|y_k^T s_k\right|}\left[1 + \left|\frac{g_{k+1}^T s_k}{y_k^T s_k}\right|\right] \\ &\leq \frac{\Gamma L\|s_k\|}{(1-\sigma_2)\omega\alpha_k\gamma^2}\left[1 + \max\left\{1, \frac{\sigma_2}{1-\sigma_2}\right\}\right] = E\|s_k\| \leq ED,\end{aligned} \tag{43}$$

where

$$E = \frac{\Gamma L}{(1-\sigma_2)\omega\alpha_k\gamma^2}\left[1 + \max\left\{1, \frac{\sigma_2}{1-\sigma_2}\right\}\right]$$

and $D = \max\{\|y - z\| : y, z \in L\}$ is the diameter of the level set L. Therefore,

$$\|d_{k+1}\| \le \|g_{k+1}\| + |\beta_k^A| \|s_k\| \le \Gamma + ED^2 \ . \tag{44}$$

Since the level set L is bounded and the function f is bounded from below, using Lemma 2, from (20) and (23) it follows that

$$0 < \sum_{k=0}^{\infty} \frac{(g_k^T d_k)^2}{\|d_k\|^2} < \infty \ , \tag{45}$$

i.e. the Zoutendijk condition holds. Therefore, from Theorem 1 using (23), the descent property yields:

$$\sum_{k=0}^{\infty} \frac{\gamma^4}{\|d_k\|^2} \le \sum_{k=0}^{\infty} \frac{\|g_k\|^4}{\|d_k\|^2} \le \sum_{k=0}^{\infty} \frac{1}{\omega^2} \frac{(g_k^T d_k)^2}{\|d_k\|^2} < \infty,$$

which contradicts (44). Hence, $\gamma = \liminf_{k \to \infty} \|g_k\| = 0.$ ■

Therefore, our conjugate gradient algorithm is globally convergent, meaning that either $g_k = 0$ for some k or (34) holds. Observe that we assume both the sufficient descent condition and the Wolfe line search conditions. But these two requirements are essentially independent of each other. On the other hand, the conjugacy condition gives a value for parameter δ_k and we assume it is bounded below.

4. Numerical results and comparisons

In this section we present the computational performance of a Fortran implementation of the ACGSD algorithm on a set of 750 unconstrained optimization test problems. The test problems are the unconstrained problems in the CUTE [6] library, along with other large-scale optimization problems. 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:

$$\left| f_i^{ALG1} - f_i^{ALG2} \right| < 10^{-3} \tag{46}$$

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 the performance of ACGSD 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:

$$\beta_k^{hDY} = \max\left\{ -\frac{1-\sigma_2}{1+\sigma_2} \beta_k^{DY}, \min\left\{ \beta_k^{HS}, \beta_k^{DY} \right\} \right\}, \tag{47}$$

and

$$\beta_k^{hDYz} = \max\left\{0, \min\left\{\beta_k^{HS}, \beta_k^{DY}\right\}\right\}, \tag{48}$$

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 [24] and Polyak [25] 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 ACGSD versus Dai-Yuan. 721 problems.

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

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

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

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

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

When comparing ACGSD and DY algorithms (Table 1), subject to the number of iterations, ACGSD 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 (46) 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 ACGSD. 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.

The second set of numerical experiments refers to the comparison of ACGSD 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:

$\beta_k^{HS} = \dfrac{g_{k+1}^T y_k}{d_k^T y_k}$	The original linear conjugate gradient algorithm by Hestenes and Stiefel [19].
$\beta_k^{FR} = \dfrac{g_{k+1}^T g_{k+1}}{g_k^T g_k}$	The first nonlinear conjugate gradient algorithm, proposed by Fletcher and Reeves [14].
$\beta_k^{PRP} = \dfrac{g_{k+1}^T y_k}{g_k^T g_k}$	Proposed by Polak and Ribière [24] and Polyak [25].

$\beta_k^{PRP+} = max\left\{0, \frac{g_{k+1}^T y_k}{g_k^T g_k}\right\}$	Proposed by Powell [26], 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 [30]
$\beta_k^{DL} = \frac{g_{k+1}^T (y_k - t s_k)}{d_k^T y_k}, t > 0$	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 :

$$\theta_{k+1} = \frac{s_k^T s_k}{y_k^T s_k} \tag{49}$$

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 [5] 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 [5].
$\beta_k^{sPRP} = \frac{\theta_k g_{k+1}^T y_k}{\alpha_k \theta_{k-1} g_k^T g_k}$	Scaled Polak-Ribière-Polyak. Suggested by Birgin and Martínez [5] 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 [5] and Andrei [1-4].

Tables 4-18 show the number of problems, out of 750, for which ACGSD 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 (46), respectively.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Table 15. Performance of ACGSD versus scaled Perry (Birgin-Martínez). 707 problems.

	ACGSD	BM	=
# iter	249	248	210
# fg	295	285	127
CPU	325	263	119

Table 16. Performance of ACGSD versus scaled Perry plus (Birgin-Martínez plus). 697 problems.

	ACGSD	BM+	=
# iter	263	248	186
# fg	310	271	116
CPU	341	255	101

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

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

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

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

From Tables above we see that ACGSD 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, ACGSD appears to generate the best search direction, on average.

Concerning the cpu time, from Table 14, we see that the closest to ACGSD is Dai-Liao ($t=1$) algorithm. Both ACGSD and DL($t=1$) achieved the minimum cpu time for 134 problems. Dai and Liao algorithm is 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 ACGSD algorithm. Both ACGSD and HS achieved the same CPU time for 131 problems. The HS method has the property that the conjugacy condition $d_{k+1}^T y_k = 0$ always is satisfied, independent of the line search. However, ACGSD satisfies both the conjugacy condition and the sufficient descent condition. From Table 6 close to ACGSD 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 ACGSD 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 ACGSD becomes small and the new search direction d_{k+1} is essentially the steepest descent direction $-g_{k+1}$. Therefore, HS, PRP and ACGSD 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 ACGSD 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 (20). The Wolfe conditions implemented in ACGSD 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 ACGSD versus CG_DESCENT(w). 700 problems.

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

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

	ACGSD	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 modification of the Dai and Yuan computational scheme in such a manner that the direction d_k generated by the algorithm to satisfy both the sufficient descent condition and the conjugacy condition, independent of the line search. Under standard Wolfe line search conditions we proved the global convergence of the algorithm. We present computational evidence that the performance of our algorithm ACGSD was higher than those of the Dai and Yuan conjugate gradient algorithm and its hybrid variants, 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*. Accepted: Computational Optimization and Applications, 2006.

[3] N. Andrei, *Scaled memoryless BFGS preconditioned conjugate gradient algorithm for unconstrained optimization.* Accepted: Optimization Methods and Software, 2006.

[4] N. Andrei, *A scaled BFGS preconditioned conjugate gradient algorithm for unconstrained optimization*. Accepted: Applied Mathematics Letters, 2006

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

[6] 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.

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

[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] 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.

[23] 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.

[24] 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.

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

[26] 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.
[27] D.F. Shanno and K.H. Phua, *Algorithm 500, Minimization of unconstrained multivariate functions*, ACM Trans. on Math. Soft., 2, pp.87-94, 1976.
[28] D.F. Shanno, *On the convergence of a new conjugate gradient algorithm,*SIAM J. Numer. Anal., 15 (1978), pp.1247-1257.
[29] D.F. Shanno, *Conjugate gradient methods with inexact searches.* Math. Oper. Res., 3 (1978), pp.244-256.
[30] D. Touati-Ahmed and C. Storey, *Efficient hybrid conjugate gradient techniques,* Journal Of Optimization Theory and Applications, 64 (1990), pp. 379-397.
[31] P. Wolfe, *Convergence conditions for ascent methods.* SIAM Review 11 (1969) 226-235.
[32] P. Wolfe, *Convergence conditions for ascent methods, (II): some corrections.* SIAM Review 13 (1971) 185-188.

November 22, 2006