Acceleration of conjugate gradient algorithms for unconstrained optimization

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

Abstract. Conjugate gradient methods are important for large-scale unconstrained optimization. This paper proposes an acceleration of these methods using a modification of steplength. The idea is to modify in a multiplicative manner the steplength α_k, computed by Wolfe line search conditions, by means of a positive parameter η_k, in such a way to improve the behavior of the classical conjugate gradient algorithms. It is shown that for uniformly convex functions the convergence of the accelerated algorithm is still linear, but the reduction in function values is significantly improved. Numerical comparisons with some conjugate gradient algorithms using a set of 750 unconstrained optimization problems, some of them from the CUTE library, show that the accelerated computational scheme outperform the corresponding conjugate gradient algorithms.

Keywords: Acceleration methods, conjugate gradient, Wolfe line search, line search gradient methods, unconstrained optimization
AMS subject classifications: 49M20, 65K05, 90C30

1. Introduction

For solving the unconstrained optimization problems

$$\min_{x\in R^n} f(x), \tag{1.1}$$

where $f: R^n \to R$ is a continuously differentiable function, bounded from below, one of the most elegant and probably the simplest methods are the conjugate gradient methods. For solving this problem, starting from an initial guess $x_0 \in R^n$, a nonlinear conjugate gradient method, generates a sequence $\{x_k\}$ as:

$$x_{k+1} = x_k + \alpha_k d_k, \tag{1.2}$$

where $\alpha_k > 0$ is obtained by line search, and the directions d_k are generated as:

$$d_{k+1} = -g_{k+1} + \beta_k d_k, \;\; d_0 = -g_0. \tag{1.3}$$

In (1.3) β_k is known as the conjugate gradient parameter, $s_k = x_{k+1} - x_k$ and $g_k = \nabla f(x_k)$. The search direction d_k, assumed to be a descent one, plays the main role in these methods. On the other hand, the stepsize α_k guarantees the global convergence in some cases and is crucial in efficiency. Plenty of conjugate gradient methods are known, and an excellent survey of these methods, with a special attention on their global convergence, is given by Hager and Zhang [17]. Different conjugate gradient algorithms correspond to different choices for the scalar parameter β_k. Line search in the conjugate gradient algorithms often is based on the standard Wolfe conditions. A numerical comparison of conjugate gradient algorithms (1.2) and (1.3) with Wolfe line search, for different formulae of parameter β_k computation, including the Dolan and Moré performance profile, is given in [2].

It is common to see that in conjugate gradient algorithms the search directions tend to be poorly scaled and as a consequence the line search must perform more function evaluations in order to obtain a suitable steplength α_k. In conjugate gradient methods the steplengths 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 methods, which accept the unit steplength most of the time along the iterations, and therefore usually they require only few function evaluations per search direction. Numerical comparisons between conjugate gradient methods and limited memory quasi Newton method by Liu and Nocedal [19] show that the later are more successful [2]. One explanation of efficiency of this limited memory quasi-Newton method is given by its ability to accept unity steplengths along the iterations.

The purpose of this paper is to present an acceleration of the conjugate gradient algorithms. The idea is to modify the steplength α_k (computed by means of Wolfe line search conditions) by means of a positive parameter η_k, in a multiplicative manner, in such a way to improve the behavior of these algorithms. We show that the resulting algorithm is linear convergent, but the reduction in function value is significantly improved. An acceleration of gradient descent algorithm with backtracking for unconstrained optimization is given in [1].

The structure of the paper is as follows. Section 2 is presenting the line search methods of Goldstein [15] and Wolfe [26], and proves that these computational schemes generate steplengths bounded away from zero. Section 3 presents the general accelerated conjugate gradient algorithm. This algorithm can be particularized using different formula of β_k computation. For uniformly convex functions we prove that if the direction satisfied the sufficient descent condition and it is bounded, then the convergence of the accelerated algorithm is at least linear. Some numerical results and comparisons including the Dolan and Moré [13] performance profiles are given in section 4.

2. Line search

For implementing the algorithm (1.2) one of the crucial elements is the stepsize computation. Many procedures have been suggested. In the *exact line search* the step α_k is selected as:

$$\alpha_k = \arg\min_{\alpha>0} f(x_k + \alpha d_k), \tag{2.1}$$

where d_k is a descent direction. In some very special cases (quadratic problems, for example) it is possible to compute the step α_k analytically, but for the vast majority of cases it is computed to approximately minimize f along the ray $\{x_k + \alpha d_k : \alpha \geq 0\}$, or at least to reduce f sufficiently. In practice the most used are the *inexact procedures*. A lot of inexact line search procedures have been proposed: Goldstein [15], Armijo [7], Wolfe [26], Powell [24], Dennis and Schnabel [12], Fletcher [14], Potra and Shi [23], Lemaréchal [18], Moré and Thuente [20], Hager and Zhang [16], and many others. The line search in conjugate gradient algorithms is often based either on the Goldstein's conditions [15]:

$$\rho_1 \alpha_k g_k^T d_k \leq f(x_k + \alpha_k d_k) - f(x_k) \leq \rho_2 \alpha_k g_k^T d_k, \tag{2.2}$$

where $0 < \rho_2 < \frac{1}{2} < \rho_1 < 1$ and $\alpha_k > 0,$ or the Wolfe conditions [26]:

$$f(x_k + \alpha_k d_k) - f(x_k) \leq \rho \alpha_k g_k^T d_k, \tag{2.3}$$

$$g_{k+1}^T d_k \geq \sigma g_k^T d_k, \tag{2.4}$$

where $0 < \rho < 1/2 \leq \sigma < 1.$ The following proposition shows that α satisfying the Goldstein or the Wolfe line search conditions is bounded away from zero (see also [16]).

Proposition 2.1. *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 positive constant. If the line search satisfies the Goldstein conditions (2.2), then*

$$\alpha_k \geq \frac{(1-\rho_1)}{L}\frac{\left|g_k^T d_k\right|}{\left\|d_k\right\|^2}. \tag{2.5}$$

If the line search satisfies the Wolfe conditions (2.3) and (2.4), then

$$\alpha_k \geq \frac{(1-\sigma)}{L}\frac{\left|g_k^T d_k\right|}{\left\|d_k\right\|^2}. \tag{2.6}$$

Proof. If the Goldstein conditions are satisfied, then using the mean value theorem from (2.2) we get:

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

where $\xi \in (0,\alpha_k)$. From this inequality we get immediately (2.5).

Now, to prove (2.6) subtract $g_k^T d_k$ from both sides of (2.4) and using the Lipschitz condition we get:

$$(\sigma-1)g_k^T d_k \leq (g_{k+1}-g_k)^T d_k \leq \alpha_k L\left\|d_k\right\|^2.$$

But, d_k is a descent direction and since $\sigma<1$, we get immediately (2.6). ■

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

3. Accelerated conjugate gradient algorithms

In this section let us present the accelerated conjugate gradient algorithms for solving the unconstrained optimization problem (1.1). Suppose that the function f is twice continuously differentiable. At the iteration $k=1,2,...$ we know x_k, f_k, g_k and $d_k=-g_k+\beta_{k-1}d_{k-1}$, where β_{k-1} is computed according to the conjugate gradient algorithm we consider to accelerate. Suppose that d_k is a descent direction. For example, the Dai-Yuan conjugate gradient algorithm, $\beta_k = g_{k+1}^T g_{k+1}/y_k^T d_k$, with a standard Wolfe line search [11] and the scaled memoryless BFGS preconditioned conjugate gradient SCALCG algorithm by Andrei [3-5] with Wolfe line search always generate descent directions. Also, the directions generated by the CG_DESCENT conjugate gradient algorithm by Hager and Zhang [16] satisfies the sufficient descent conditions, independent by the accuracy of line search.

Now, by the Wolfe line search (2.3) and (2.4) we can compute α_k with which the following point $z=x_k+\alpha_k d_k$ is determined. The first Wolfe condition (2.3) shows that the steplength $\alpha_k>0$, satisfies:

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

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

$$x_{k+1}=x_k+\eta_k\alpha_k d_k, \tag{3.1}$$

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

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

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

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

With these we can write:

$$f(x_k + \eta\alpha_k d_k) = f(x_k + \alpha_k d_k) + \Psi_k(\eta), \tag{3.2}$$

where

$$\Psi_k(\eta) = \frac{1}{2}(\eta^2 - 1)\alpha_k^2 d_k^T \nabla^2 f(x_k) d_k + (\eta - 1)\alpha_k g_k^T d_k$$
$$+\eta^2\alpha_k o\left(\alpha_k \left\|d_k\right\|^2\right) - \alpha_k o\left(\alpha_k \left\|d_k\right\|^2\right). \tag{3.3}$$

Let us denote:

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

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

$$\Psi_k(\eta) = \frac{1}{2}(\eta^2 - 1)b_k + (\eta - 1)a_k + \eta^2\alpha_k\varepsilon_k - \alpha_k\varepsilon_k. \tag{3.4}$$

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

$$\eta_m = -\frac{a_k}{b_k + 2\alpha_k\varepsilon_k}. \tag{3.5}$$

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

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

Considering $\eta = \eta_m$ in (3.2) and since $b_k \ge 0$, we see that for every k

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

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

$$f(x_{k+1}) = f(x_k + \eta_k\alpha_k d_k) \le f(x_k) + \rho\alpha_k g_k^T d_k - \frac{(a_k + (b_k + 2\alpha_k\varepsilon_k))^2}{2(b_k + 2\alpha_k\varepsilon_k)}$$
$$= f(x_k) - \left[\frac{(a_k + (b_k + 2\alpha_k\varepsilon_k))^2}{2(b_k + 2\alpha_k\varepsilon_k)} - \rho a_k\right] \le f(x_k), \tag{3.6}$$

since $a_k \le 0$, (d_k is a descent direction).
Now, neglecting the contribution of ε_k in (3.6), we still get an improvement on the function values as

$$f(x_{k+1}) \le f(x_k) - \left[\frac{(a_k + b_k)^2}{2b_k} - \rho a_k\right] \le f(x_k). \tag{3.7}$$

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

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

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

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

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

$$b_k = -\alpha_k y_k^T d_k, \tag{3.8}$$

where $y_k = g_k - g_z$. Observe that the computation of b_k needs an additional evaluation of the gradient in point z. Therefore, neglecting the contribution of ε_k and considering in our algorithm $\eta_k = \eta_m = -a_k / b_k$, the following algorithm can be presented.

Accelerated conjugate gradient algorithm (ACG)

Step 1. Select a starting point $x_0 \in dom\, f$ and compute: $f_0 = f(x_0)$ and $g_0 = \nabla f(x_0)$. Set $d_0 = -g_0$ and $k = 0$.

Step 2. Test a criterion for stopping the iterations. If the test is satisfied, then stop; otherwise continue with step 3.

Step 3. Using the Wolfe line search conditions determine the steplength α_k.

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

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

Step 6. If $b_k \neq 0$, then compute $\eta_k = -a_k / b_k$ and update the variables as $x_{k+1} = x_k + \eta_k \alpha_k d_k$, 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. Determine β_k according to the conjugate gradient in use.

Step 8. Compute the search direction as: $d_{k+1} = -g_{k+1} + \beta_k d_k$.

Step 9. Restart criterion. If $\left|g_{k+1}^T g_k\right| > 0.2\left\|g_{k+1}\right\|^2$ then set $d_{k+1} = -g_{k+1}$.

Step 10. Consider $k = k + 1$ and go to step 2. ■

The conjugate gradient algorithm (CG) can be immediately particularized from ACG by skipping steps 4 and 5 and by modifying step 6 where the variables are updated. In step 7, where the conjugate gradient parameter β_k is computed we can consider the formula corresponding to the conjugate gradient algorithm we have selected for acceleration. For example, we can consider the Dai and Yuan [11] $\beta_k = g_{k+1}^T g_{k+1} / y_k^T d_k$, the BFGS preconditioned conjugate gradient SCALCG [3-5], Polak-Ribière-Polyak [21, 22] $\beta_k = y_k^T g_{k+1} / g_k^T g_k$, Dai and Liao [10] $\beta_k = g_{k+1}^T (y_k - t s_k) / y_k^T d_k$ $(t > 0)$, or any other conjugate gradient we want to accelerate.

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 (2.3) and (2.4). In our algorithm when the Powell

restart condition is satisfied, then we restart the algorithm with the negative gradient $-g_{k+1}$. Under reasonable assumptions, the Wolfe conditions and the Powell restart criterion are sufficient to prove the global convergence of the algorithm.
The first trial 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 $\alpha_{k-1}\|d_{k-1}\|/\|d_k\|$. This selection was used for the first time by Shanno and Phua in CONMIN [25]. It is also considered in SCALCG [3-5].

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

Neglecting ε_k in (3.4), we see that $\Psi_k(1) = 0$ and if $|a_k| \leq b_k/2$, then $\Psi_k(0) = -a_k - b_k/2 \leq 0$ and $\eta_k < 1$. Therefore, for any $\eta \in [0,1]$, $\Psi_k(\eta) \leq 0$. As a consequence for any $\eta \in (0,1)$, it follows that $f(x_k + \eta\alpha_k d_k) < f(x_k)$. In this case, for any $\eta \in [0,1]$, $\eta_k \alpha_k \leq \alpha_k$. However, in our algorithm we selected $\eta_k = \eta_m$ as the point achieving the minimum value of $\Psi_k(\eta)$.

In the following, for uniformly convex functions, we prove the linear convergence of the acceleration scheme. Recall that a function f is uniformly convex on the level set $S = \{x : f(x) \leq f(x_0)\}$ if there is a positive constant m such that

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) + \frac{1}{2} m \|y - x\|^2$$

for all $x, y \in S$. For uniformly convex functions it is easy to prove that

$$\|\nabla f(x)\|^2 \geq 2m\big(f(x) - f(x^*)\big),$$

for all $x \in S$, where x^* is a local solution of (1.1) [9].

Proposition 3.1. *Suppose that f is a uniformly convex function on the level set S, 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 ACG converges linearly to x^*, solution to problem (1.1).*

Proof. From (3.6) we have that $f(x_{k+1}) \leq f(x_k)$ for all k. Since f is bounded below, it follows that

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

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

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

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

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

Therefore,

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

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

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

since $\rho < 1/2$.

From proposition 2.1 the Wolfe line search terminates with a value $\alpha \ge \gamma > 0$. Therefore, for $0 \le \alpha \le c_1/(Mc_2)$, this provides a lower bound on the decrease in the function f, i.e.

$$f(x_k + \alpha d_k) \le f(x_k) - \rho c_1 \gamma \|g_k\|^2. \tag{3.9}$$

On the other hand,

$$\frac{(a_k + b_k)^2}{2b_k} \ge \frac{(\alpha^2 M c_2 - \alpha c_1)^2 \|g_k\|^4}{2\alpha^2 M c_2 \|g_k\|^2} \ge \frac{(\gamma M c_2 - c_1)^2}{2Mc_2} \|g_k\|^2. \tag{3.10}$$

Considering (3.9) and (3.10) we get:

$$f(x_k + \eta_m \alpha d_k) \le f(x_k) - \rho c_1 \gamma \|g_k\|^2 - \frac{(\gamma M c_2 - c_1)^2}{2Mc_2} \|g_k\|^2. \tag{3.11}$$

Therefore

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

But, $f(x_k) - f(x_{k+1}) \to 0$ and as a consequence g_k goes to zero, i.e. x_k converges to x^*. Having in view that $f(x_k)$ is a nonincreasing sequence, it follows that $f(x_k)$ converges to $f(x^*)$. From (3.11) we see that

$$f(x_{k+1}) \le f(x_k) - \left[\rho c_1 \gamma + \frac{(\gamma M c_2 - c_1)^2}{2Mc_2} \right] \|g_k\|^2. \tag{3.12}$$

Combining this with $\|g_k\|^2 \ge 2m(f(x_k) - f(x^*))$ and subtracting f^* from both sides of (3.12) we conclude:

$$f(x_{k+1}) - f(x^*) \le c(f(x_k) - f(x^*)),$$

where

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

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

4. Numerical results and comparisons

In this section we report some numerical results obtained with a Fortran implementation of conjugate gradient algorithms and their accelerated variants. All codes are written in Fortran and compiled with f77 (default compiler settings) on a Workstation Intel Pentium 4 with 1.8 GHz. We selected a number of 75 large-scale unconstrained optimization test functions in generalized or extended form [6] (some from CUTE library [8]). For each test function we have considered ten numerical experiments with the number of variables $n = 1000, 2000, ..., 10000$. In the following we present the numerical performance of CG and ACG codes corresponding to different formula for β_k computation. All algorithms implement the Wolfe line search conditions with $\rho = 0.0001$ and $\sigma = 0.9$, and the same stopping criterion $\|g_k\|_\infty \leq 10^{-6}$, where $\|.\|_\infty$ is the maximum absolute component of a vector.

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

$$\left| f_i^{ALG1} - f_i^{ALG2} \right| < 10^{-3} \tag{4.1}$$

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.

In the first set of computational experiments we accelerate the Dai and Yuan (DY) conjugate gradient algorithm [11], where β_k is computed as $\beta_k = g_{k+1}^T g_{k+1} / y_k^T d_k$. Recall that the directions generated by the Dai-Yuan algorithm are descent. Figure 4.1 illustrates the Dolan and Moré [13] performance profiles of DY and accelerated DY.

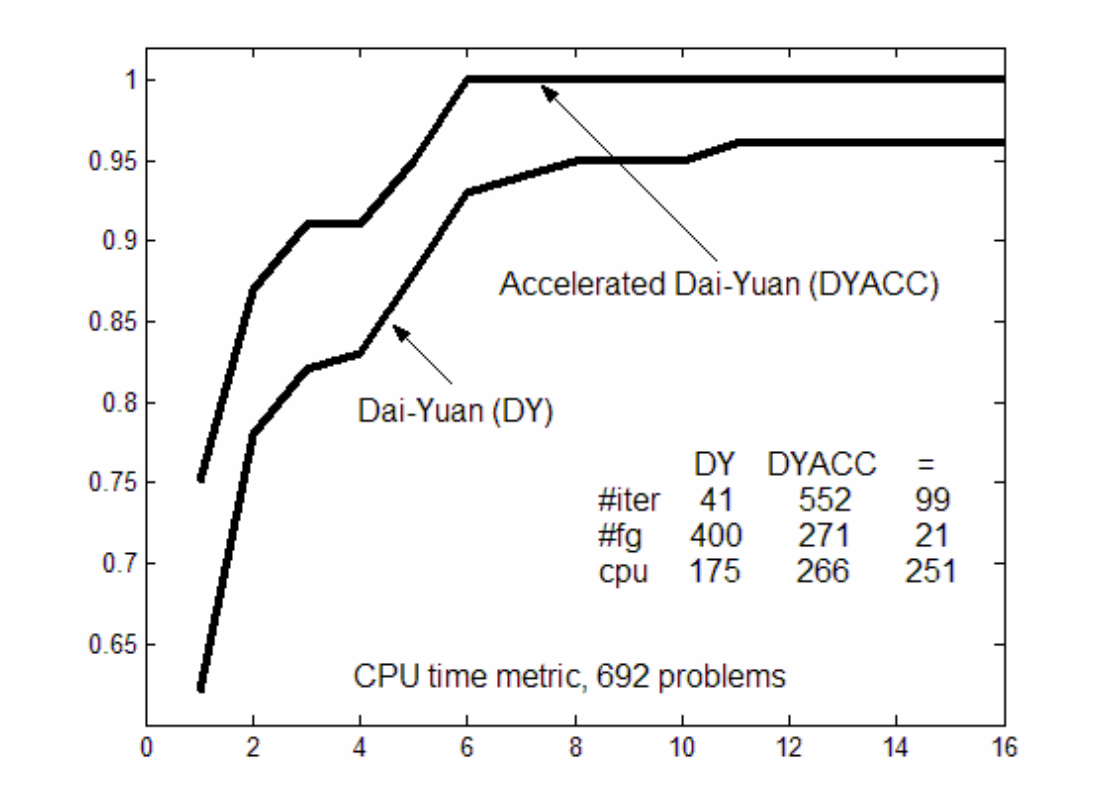

Fig. 4.1. Performance profiles DY versus accelerated DY. CPU time metric.

For each algorithm, we plot the fraction of problems for which the algorithm is within a factor τ of the best cpu time. Relative to performance profiles, the top curve corresponds to the method that solved the most problems in a time that was within a factor τ of the best time. When comparing accelerated Dai-Yuan (DYACC) with Dai-Yuan (DY) (see Figure 4.1), subject to the number of iterations, we see that DYACC was better in 552 problems (i.e. it achieved the minimum number of iterations in 552 problems). DY was better in 41 problems and they achieved the same number of iterations in 99 problems, etc. Out of 750 problems, only for 692 problems does the criterion (4.1) hold. Observe that, subject to the number of function and its gradient evaluations, DY was better in 400 problems. On the other hand, DYACC was better only in 271 problems. This is because the accelerated version of DY at every iteration needs an extra gradient evaluation (only when $b_k \neq 0$).

In the second set of numerical experiments we consider the acceleration of scaled memoryless BFGS preconditioned SCALCG conjugate gradient algorithm by Andrei [3-5]. Again this algorithm with Wolfe line search generates descent directions. Figure 4.2 presents the performance profiles proposed by Dolan and Moré for the SCALCG and accelerated SCALCG.

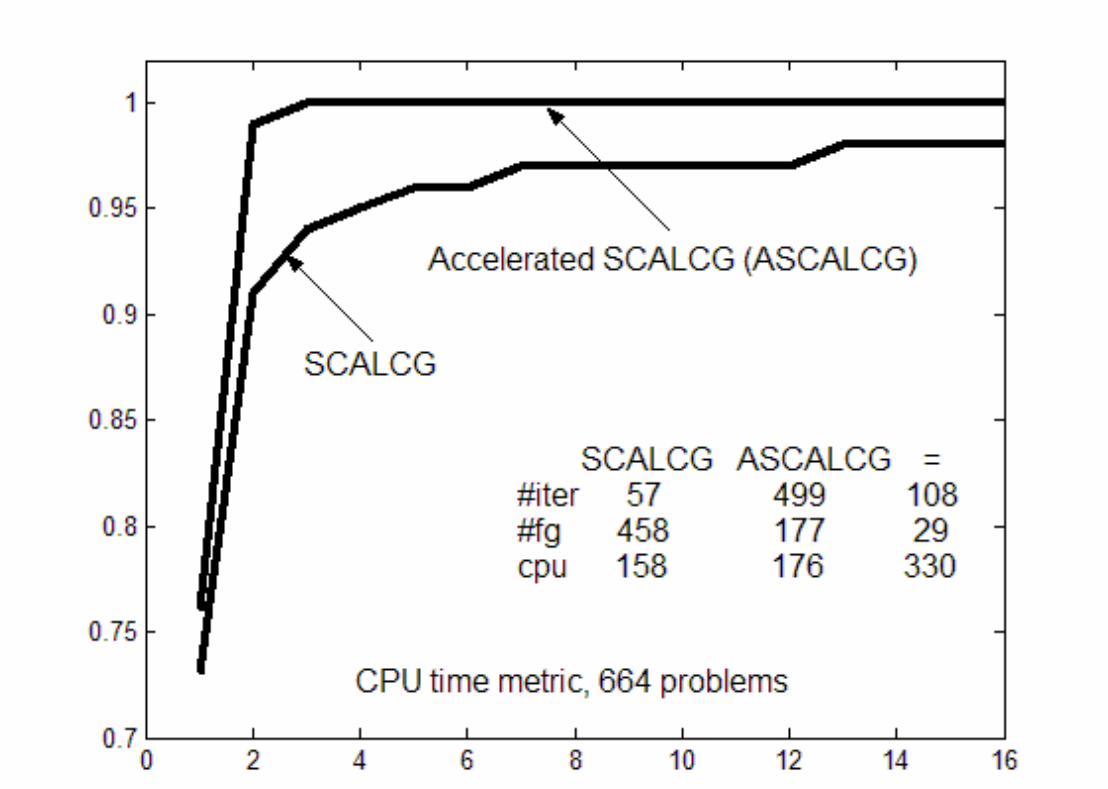

Fig. 4.2. Performance profiles SCALCG versus accelerated SCALCG. CPU time metric.

In the third set of numerical experiments we accelerate the Polak-Ribière-Polyak (PRP) conjugate gradient algorithm [21, 22]. In this algorithm the search direction is computed as in (1.3), where $\beta_k = y_k^T g_{k+1} / g_k^T g_k$. The convergence of the PRP method for general nonlinear functions is uncertain. Even for strongly convex functions, the PRP method may not generate descent directions with inexact line search. However, the PRP method has a built-in restart feature that addresses to jamming. Figure 4.3 presents the Dolan and Moré performance profiles for the PRP and accelerated PRP. The accelerated version of PRP proved to be more efficient than the classical PRP, at least for this set of 622 unconstrained optimization problems.

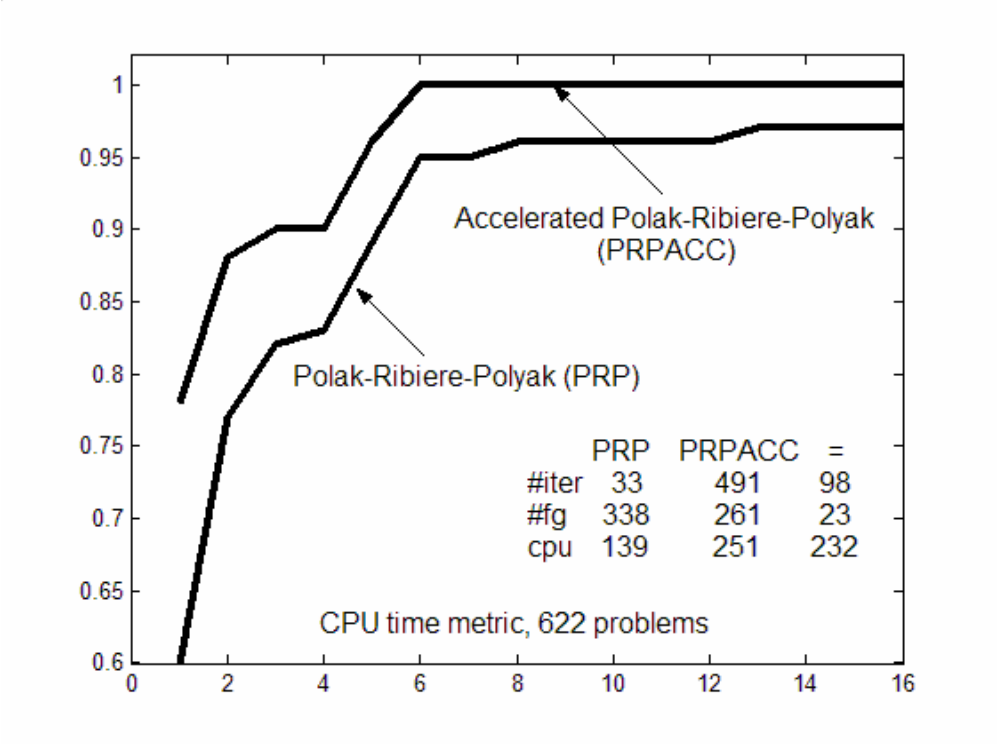

Fig. 4.3. Performance profiles PRP versus accelerated PRP. CPU time metric.

Finally, we accelerate the Dai and Liao (DL) conjugate gradient algorithm [10], where $\beta_k = g_{k+1}^T (y_k - ts_k) / y_k^T d_k$, $(t = 1)$. Figure 4.4 presents the Dolan and Moré performance profiles of DL and accelerated DL.

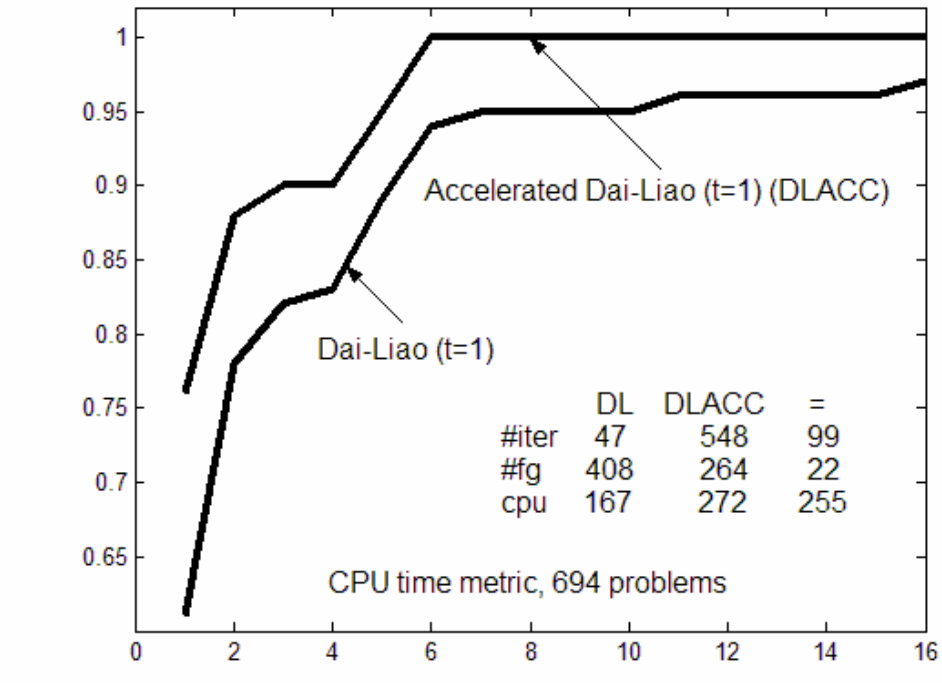

Fig. 4.4. Performance profiles DL versus accelerated DL. CPU time metric.

The left side of these Figures (small values of τ) gives the percentage of the test problems, out of 750, for which an algorithm is more successful; the right side (large values of τ) gives

the percentage of the test problems that were successfully solved by each of the algorithms. Mainly, the right side represents a measure of an algorithm's robustness. Observe that the accelerated variants outperform the corresponding conjugate gradient algorithms in the vast majority of problems, and the differences are substantial. Besides, the accelerated variants are more robust than the corresponding original conjugate gradient algorithms we considered here.

Since both the conjugate gradient algorithms DY, SCALCG, PRP and DL(t=1) and their accelerated variants use the same search direction (as dictated by the procedure for β_k selection), these algorithms only differ in their choice of the steplength. From the Tables in the above Figures it appears that the accelerated variants generate a better steplength, on average. Since the accelerated conjugate gradient algorithms performs well in the cpu time metric for all values of τ, we conclude that the overall poor performance of the original conjugate gradient algorithms is connected with the poor performance of the line search. In particular, to ensure descent the line search in conjugate gradient algorithms must achieve sufficient accuracy. In accelerated variants this is compensated by this simple modification of the steplength through η_k.

It is worth seeing that from the first Wolfe condition (2.3) we have

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \rho \alpha_k g_k^T d_k. \tag{4.2}$$

Observe that along the iterations $g_k^T d_k$ in (4.2) is of a small order of magnitude, its contribution to reduce the function values along the direction d_k being almost insignificant. Since the conjugate gradient method uses only the linear approximation of f to find the search direction, ignoring completely the second order term, we expect that the direction generated will not be very effective, if the second order term contributes significantly to the description of f, even for relatively small values of α_k. On the other hand, for accelerated conjugate gradient this is compensated by modifying the steplength in order to destroy the premature orthogonality of search directions to gradient. Besides,

$$f(x_k + \eta_m \alpha_k d_k) \leq f(x_k) + \rho a_k - \frac{(a_k + b_k)^2}{2b_k}. \tag{4.3}$$

Although the contribution of ρa_k to reducing the function values is small, the term $(a_k + b_k)^2 / (2b_k) \geq 0$ gives the possibility of a substantial progress towards minimum. Observe that the last term in (4.3) is independent of parameter ρ from the Wolfe conditions. However, the price we must pay for this acceleration scheme of the conjugate gradient algorithms is an additional evaluation at each iteration of the gradient of the function f.

Observe that subject to the number of function and its gradient evaluations the classical conjugate gradient algorithms are better. However, the accelerated conjugate gradient variants, by modifying the steplength in such a manner to emphasize the reduction of function values, determine a better trajectory of optimization.

5. Conclusions

Intensive numerical experiments with different variants of conjugate gradient algorithms proved that the step length may differ from 1 up to two orders of magnitude, being larger or smaller than 1, depending on how the problem is scaled. Moreover, the sizes of the step length tend to vary in a totally unpredictable way. This is in sharp contrast with the Newton and quasi-Newton methods, as well as with the limited memory quasi-Newton methods, which usually admit the unit step length for most of the iterations, thus requiring only very few function evaluations for step length determination. One explanation of the efficiency of the limited memory quasi-Newton method is given by its ability to accept unity steplengths along the iterations.

In this paper we take the advantage of this behavior of conjugate gradient algorithms and suggest an acceleration procedure of conjugate gradient algorithms by modifying the steplength α_k (computed by means of the Wolfe line search conditions) through a positive parameter η_k, in a multiplicative manner, like $x_{k+1} = x_k + \eta_k \alpha_k d_k$, in such a way as to improve the reduction of the function's values along the iterations. It is shown that for uniformly convex functions the acceleration scheme is linear convergent, but the reduction in function value is significantly improved. Numerical experiments proved that the accelerated DY, SCALCG, PRP and DL outperform the corresponding conjugate gradient algorithms on a set of 750 large-scale unconstrained optimization problems.

References

1. Andrei, N., *An acceleration of gradient descent algorithm with backtracking for unconstrained optimization*, Numerical Algorithms, 42 (2006) pp. 63-73.
2. Andrei, N., *Numerical comparison of conjugate gradient algorithms for unconstrained optimization.* Studies in Informatics and Control, 16 (2007) pp. 333-352.
3. Andrei, N., *Scaled conjugate gradient algorithms for unconstrained optimization*. Computational Optimization and Applications, 38 (2007) 401-416.
4. Andrei, N., *Scaled memoryless BFGS preconditioned conjugate gradient algorithm for unconstrained optimization*. Optimization Methods and Software, 22 (2007) 561-571.
5. Andrei, N., *A scaled BFGS preconditioned conjugate gradient algorithm for unconstrained optimization.* Applied Mathematics Letters, 20 (2007) 645-650.
6. Andrei, N., *An unconstrained optimization test functions collection.* Advanced Modeling and Optimization. An Electronic International Journal, 10 (2008) 147-161.
7. Armijo, L., *Minimization of functions having Lipschitz continuous first partial derivatives*, Pac. J. Math., 6 (1966) pp.1-3.
8. Bongartz, I., Conn, A.R., Gould, N.I.M., and Toint, P.L., *CUTE: constrained and unconstrained testing environments*, ACM Trans. Math. Software, 21 (1995) pp.123-160.
9. Boyd, S., and Vandenberghe, L., *Convex optimization.* Cambridge university Press, 2004.
10. Dai, Y.H., and Liao, L.Z., *New conjugacy conditions and related nonlinear conjugate gradient methods.* Appl. Math. Optim., 43 (2001) pp. 87-101.
11. Dai, Y.H., and Yuan, Y., *A nonlinear conjugate gradient method with a strong global convergence property*, SIAM J. Optim., 10 (1999) pp. 177-182.
12. Dennis, J.E. and Schnabel, R.B., *Numerical Methods for Unconstrained Optimization and Nonlinear Equations,* Prentice-Hall, Englewoods Cliffs, New Jersey, 1983.
13. Dolan, E., and Moré, J.J., *Benchmarking optimization software with performance profiles,* Math. Programming, 91 (2002) pp.201-213.
14. Fletcher, R., *Practical Methods of Optimization,* Wiley, New York, 1987.
15. Goldstein, A.A., *On steepest descent,* SIAM J. Control, 3 (1965) pp.147-151.
16. Hager, W.W. and Zhang, H., *A new conjugate gradient method with guaranteed descent and an efficient line search*, SIAM Journal on Optimization, 16 (2005) pp. 170-192.
17. Hager, W.W. and Zhang, H., *A survey of nonlinear conjugate gradient methods*. Pacific journal of Optimization, 2 (2006), pp.35-58.
18. Lemaréchal, C., *A view of line search.* In Optimization and Optimal Control, Edited by Auslander, A., Oettli, W., and Stoer, J., Springer, Berlin, pp.59-78, 1981.
19. Liu, D.C. and Nocedal, J., *On the limited memory BFGS method for large scale optimization.* Mathematical Programming, 45 (1989), pp.503-528.
20. Moré, J.J. and Thuente, D.J., *On line search algorithms with guaranteed sufficient decrease,* Mathematics and Computer Science Division Preprint MCS-P153-0590, Argonne National Laboratory, Argonne, 1990.
21. Polak, E., and Ribière, G., *Note sur la convergence de directions conjuguée,* Rev. Francaise Informat Recherche Operationelle, 3e Année 16 (1969), pp.35-43.
22. Polyak, B.T., *The conjugate gradient method in extreme problems*. USSR Comp. Math. Math. Phys., 9 (1969), pp.94-112.

23. Potra, F.A. and Shi, Y., *Efficient line search algorithm for unconstrained optimization,* Journal of Optimization Theory and Applications, 85 (1995) pp.677-704.
24. Powell, M.J.D., *Some global convergence properties of a variable-metric algorithm for minimization without exact searches,* SIAM-AMS Proc., Philadelphia, 9 (1976) pp.53-72.
25. Shanno, D.F. and Phua, K.H., *Algorithm 500, Minimization of unconstrained multivariate functions*, ACM Trans. on Math. Soft., 2 (1976) pp.87-94.
26. Wolfe, P., *Convergence conditions for ascent methods,* SIAM Rev., 11 (1968) pp.226-235.

March 16, 2009
Published in:
Applied Mathematics and Computation

Initial date submitted: **April 18, 2008**
First review: **November 2, 2008**
Accepted: **March 10, 2009**