Accelerated conjugate gradient algorithm with finite difference Hessian / vector product approximation 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. In this paper we propose a fundamentally different conjugate gradient method, in which the well known parameter β_k is computed by an approximation of the Hessian / vector product through finite differences. For search direction computation, the method uses a forward difference approximation to the Hessian / vector product in combination with a careful choice of the finite difference interval. For the steplength computation we suggest an acceleration scheme able to improve the efficiency of the algorithm. Under common assumptions, the method is proved to be globally convergent. 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 conjugate gradient algorithms including CONMIN by Shanno and Phua [22], SCALCG by Andrei [3-5], and new conjugacy condition and related new conjugate gradient by Li, Tang and Wei [16] or truncated Newton TN by Nash [18] using a set of 750 unconstrained optimization test problems show that the suggested algorithm outperforms the these conjugate gradient algorithms as well as TN.

MSC: 49M07, 49M10, 90C06, 65K
Keywords: Unconstrained optimization, conjugate gradient method, Newton direction, forward difference approximation of Hessian/vector product, numerical comparisons

1. Introduction

Conjugate gradient algorithms are very powerful methods for solving large-scale unconstrained optimization problems characterized by low memory requirements and strong local and global convergence properties. Let us consider the nonlinear unconstrained optimization problem

$$\min\left\{ f(x) : x \in R^n \right\}, \tag{1.1}$$

where $f : R^n \to R$ is a continuously differentiable function, bounded from below. As we know, 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

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

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 line search in the conjugate gradient algorithms is often based on the standard Wolfe conditions [23,24]:

$$f(x_k + \alpha_k d_k) - f(x_k) \le \rho \alpha_k g_k^T d_k, \tag{1.4}$$

$$g(x_k + \alpha_k d_k)^T d_k \ge \sigma g_k^T d_k, \tag{1.5}$$

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

The search direction d_k, assumed to be a descent one, plays the main role in these methods. Different conjugate gradient algorithms correspond to different choices for the scalar parameter β_k. On the other hand the stepsize α_k guarantees the global convergence in some cases and is crucial in efficiency. The line search in the conjugate gradient algorithms is often based on the standard Wolfe conditions. 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 [14]. A numerical comparison of conjugate gradient algorithms (1.2) and (1.3) with Wolfe line search (1.4) and (1.5), for different formulae of parameter β_k computation, including the Dolan and Moré [12] performance profile, is given in [6].

In [19] Jorge Nocedal articulated a number of open problems in conjugate gradient algorithms. Two of them seem to be really very important. One refers to the direction computation in order to take into account the problem structure. The second one focuses on the step length.

In this paper we present a conjugate gradient algorithm which address to these open problems. The structure of the paper is as follows. In section 2 we present a conjugate gradient algorithm in which the well known parameter β_k contains the Hessian $\nabla^2 f(x_{k+1})$ of the minimizing function. The idea of this algorithm is to use the Newton direction for β_k computation in (1.3). In section 3 we present the convergence of the algorithm. We prove that under common assumptions and if the direction is a descent one then the method is globally convergent. In section 4 we present an acceleration scheme of the algorithm. The idea of this computational scheme is to take advantage that the step lengths α_k in conjugate gradient algorithms are very different from 1. Therefore, we suggest we modify α_k in such a manner as to improve the reduction of the function values along the iterations. Section 5 is devoted to present the ACGHES algorithm. We prove 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 of our algorithm with some other conjugate gradient algorithms including CONMIN by Shanno and Phua [22], SCALCG by Andrei [3-5] or new conjugacy condition and related new conjugate gradient by Li, Tang and Wei [16] as well as truncated Newton TN by Nash [18] are presented in section 6. For this we use a set of 750 unconstrained optimization problems presented in [1]. We present numerical computational evidence that our suggested algorithm outperforms the known conjugate gradient algorithms as well as TN.

2. Conjugate gradient algorithm with Hessian in β_k

Our motivation to get a good algorithm for solving (1.1) is to choose the parameter β_k in (1.3) in such a way so that for every $k \geq 1$ the direction d_{k+1} given by (1.3) be the Newton direction. This is motivated by the fact that when the initial point x_0 is near the solution of (1.1) and the Hessian is a nonsingular matrix then the Newton direction is the best line search direction. Therefore, from the equation

$$-\nabla^2 f(x_{k+1})^{-1} g_{k+1} = -g_{k+1} + \beta_k s_k .$$

after some algebra we get:

$$\beta_k = \frac{s_k^T \nabla^2 f(x_{k+1}) g_{k+1} - s_k^T g_{k+1}}{s_k^T \nabla^2 f(x_{k+1}) s_k} . \tag{2.1}$$

The salient point with this formula for β_k computation is the presence of the Hessian. Observe that if the line search is exact we get the Daniel method [11]. Using (2.1) in (1.3) we get:

$$d_{k+1} = -g_{k+1} + \frac{s_k^T \nabla^2 f(x_{k+1}) g_{k+1} - s_k^T g_{k+1}}{s_k^T \nabla^2 f(x_{k+1}) s_k} s_k . \tag{2.2}$$

Theorem 2.1. *Suppose that* $\nabla^2 f(x_{k+1})$ *is positive definite. If* $0 \le [\nabla^2 f(x_{k+1})]_{ij} \le 2$, $1 \le i, j \le n$, *then* d_{k+1} *given by (2.2) is a descent direction.*

Proof. From (2.2) we can write:

$$\begin{aligned} g_{k+1}^T d_{k+1} &= -\|g_{k+1}\|^2 + \frac{(s_k^T \nabla^2 f(x_{k+1}) g_{k+1})(s_k^T g_{k+1})}{s_k^T \nabla^2 f(x_{k+1}) s_k} - \frac{(s_k^T g_{k+1})^2}{s_k^T \nabla^2 f(x_{k+1}) s_k} \\ &= \frac{1}{(s_k^T \nabla^2 f(x_{k+1}) s_k)^2} \Big[-\|g_{k+1}\|^2 (s_k^T \nabla^2 f(x_{k+1}) s_k)^2 \\ &\quad + (s_k^T \nabla^2 f(x_{k+1}) g_{k+1})(s_k^T g_{k+1})(s_k^T \nabla^2 f(x_{k+1}) s_k) \\ &\quad - (s_k^T g_{k+1})^2 (s_k^T \nabla^2 f(x_{k+1}) s_k) \Big] . \end{aligned}$$

But,

$$\begin{aligned} &(s_k^T \nabla^2 f(x_{k+1}) g_{k+1})(s_k^T g_{k+1})(s_k^T \nabla^2 f(x_{k+1}) s_k) \\ &= \Big[(s_k^T \nabla^2 f(x_{k+1}) s_k) g_{k+1} \Big]^T \Big[(s_k^T g_{k+1}) \nabla^2 f(x_{k+1}) s_k \Big] \\ &\le \frac{1}{2} \Big[(s_k^T \nabla^2 f(x_{k+1}) s_k)^2 \|g_{k+1}\|^2 + (s_k^T g_{k+1})^2 \|\nabla^2 f(x_{k+1}) s_k\|^2 \Big] \end{aligned}$$

Therefore,

$$\begin{aligned} g_{k+1}^T d_{k+1} \le \frac{1}{(s_k^T \nabla^2 f(x_{k+1}) s_k)^2} \Big[&-\frac{1}{2} \|g_{k+1}\|^2 (s_k^T \nabla^2 f(x_{k+1}) s_k)^2 \\ &+ \frac{1}{2} (s_k^T g_{k+1})^2 s_k^T \Big((\nabla^2 f(x_{k+1}))^2 - 2\nabla^2 f(x_{k+1}) \Big) s_k \Big] . \end{aligned}$$

Since $(\nabla^2 f(x_{k+1}))^2 - 2\nabla^2 f(x_{k+1})$ is a convex function and negative on $[0, 2]$ it follows that $g_{k+1}^T d_{k+1} \le 0$, i.e. d_{k+1} is a descent direction. ■

3. Convergence analysis

In this section we analyse the convergence of the algorithm (1.2) and (2.2), where $d_0 = -g_0$. In the following we consider that $g_k \ne 0$ for all $k \ge 1$, otherwise a stationary point is obtained. Assume that:

(i) *The level set* $S = \{ x \in R^n : f(x) \le f(x_0) \}$ *is bounded, i.e. there is a constant D such that* $\|x\| \le D$ *for all* $x \in S$.

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

Under these assumptions on f there exists a constant $\Gamma \ge 0$ such that $\|\nabla f(x)\| \le \Gamma$ for all $x \in S$. In order to prove the global convergence, we assume that the step size α_k in (1.2) is obtained by the strong Wolfe line search, that is,

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

$$\left| g(x_k + \alpha_k d_k)^T d_k \right| \le \sigma g_k^T d_k . \tag{3.2}$$

where ρ and σ are positive constants such that $0 < \rho \le \sigma < 1$.
Dai *et al.* [10] proved that for any conjugate gradient method with strong Wolfe line search the following general result holds:

Lemma 3.1. *Suppose that the assumptions (i) and (ii) hold and consider any conjugate gradient method (1.2) and (1.3), where d_k is a descent direction and α_k is obtained by the strong Wolfe line search (3.1) and (3.2). If*

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

then

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

Therefore, the following theorem can be proved.

Theorem 3.1. *Suppose that the assumptions (i) and (ii) hold and consider the conjugate gradient algorithm (1.2), where the direction d_{k+1} is given by (2.2) and the step length α_k is obtained by the strong Wolfe line search (3.1) and (3.2). Assume that $mI \le \nabla^2 f(x_{k+1}) \le MI$, where m and M are positive constants, then $\liminf_{k\to\infty} \|g_k\| = 0$.*

Proof. Since $mI \le \nabla^2 f(x_{k+1}) \le MI$, it follows that

$$\begin{aligned} |\beta_k| &\le \frac{\left|s_k^T \nabla^2 f(x_{k+1}) g_{k+1}\right|}{\left|s_k^T \nabla^2 f(x_{k+1}) s_k\right|} + \frac{\left|s_k^T g_{k+1}\right|}{\left|s_k^T \nabla^2 f(x_{k+1}) s_k\right|} \\ &\le \frac{M\left|s_k^T g_{k+1}\right|}{m\|s_k\|^2} + \frac{\left|s_k^T g_{k+1}\right|}{m\|s_k\|^2} = \left(\frac{M}{m}+1\right)\frac{\left|s_k^T g_{k+1}\right|}{\|s_k\|^2} \le \left(\frac{M}{m}+1\right)\frac{\|g_{k+1}\|}{\|s_k\|}. \end{aligned}$$

Therefore,

$$\|d_{k+1}\| \le \|g_{k+1}\| + |\beta_k|\|s_k\| \le \|g_{k+1}\| + \left(\frac{M}{m}+1\right)\frac{\Gamma}{\|s_k\|}\|s_k\| \le \Gamma\left(\frac{M}{m}+2\right)$$

Hence,

$$\sum_{k\ge 1} \frac{1}{\|d_k\|^2} \ge \left(\frac{m}{\Gamma(M+2m)}\right)^2 \sum_{k\ge 1} 1 = \infty.$$

By Lemma 1 we have $\liminf_{k\to\infty} \|g_k\| = 0$. $\blacksquare$

4. Acceleration of the algorithm

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. Therefore, the research efforts was directed to design procedures for direction computation which takes the second order information. The algorithms implemented in CONMIN by Shanno and Phua [22] or SCALCG by Andrei [3-5] use the BFGS preconditioning with remarkable results. In this section we focus on the step length modification. In the context of gradient descent algorithm with backtracking this idea of step length modification has been considered for the first time in [2].
Jorge Nocedal [19] pointed out that in conjugate gradient methods the step lengths may differ from 1 in a very unpredictable manner. They can be larger or smaller than 1 depending on how the problem is scaled. This is in very sharp contrast to the Newton and quasi-Newton

methods, including the limited memory quasi-Newton methods, which accept the unit steplength most of the time along the iterations, and therefore usually they require only few function evaluations per search direction. Numerical comparisons between conjugate gradient methods and the limited memory quasi Newton method, by Liu and Nocedal [17], show that the latter is more successful [6]. One explanation of efficiency of this limited memory quasi-Newton method is given by its ability to accept unity step lengths along the iterations. In this section we take advantage of this behavior of conjugate gradient algorithms and present an acceleration scheme. Basically it modifies the step length in a multiplicative manner to improve the reduction of the function values along the iterations. First we prove that the step length α_k given by the Goldstein [13] or the Wolfe line search conditions [23,24] is bounded away from zero. Secondly, we present the acceleration scheme.

Line search. For implementing the algorithm (1.2) one of the crucial elements is the stepsize computation. In the following we consider the line searches that satisfy either the Goldstein's conditions

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

where $0 < \rho_2 < \frac{1}{2} < \rho_1 < 1$ and $\alpha_k > 0$, or the Wolfe conditions (1.4) and (1.5).

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

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

If the line search satisfies the Wolfe conditions (1.4) and (1.5), then

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

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

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

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

Now, to prove (4.3) subtract $g_k^T d_k$ from both sides of (1.5) and using the Lipschitz condition we get:

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

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

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

Acceleration scheme. Given the initial point x_0 we can compute $f_0 = f(x_0)$, $g_0 = \nabla f(x_0)$ and by Wolfe line search conditions (1.4) and (1.5) the steplength α_0 is determined. With these, the next iteration is computed as: $x_1 = x_0 + \alpha_0 d_0$, ($d_0 = -g_0$) where f_1 and g_1 are

immediately determined, and the direction d_1 can be computed as: $d_1 = -g_1 + \beta_0 d_0$, where β_0 is determined like in (2.1) as it is specified later. Therefore, at the iteration $k = 1, 2, ...$ we know x_k, f_k, g_k and $d_k = -g_k + \beta_{k-1} s_{k-1}$. Suppose that d_k is a descent direction. By the Wolfe line search (1.4) and (1.5) we can compute α_k with which the following point $z = x_k + \alpha_k d_k$ is determined. The first Wolfe condition (1.4) shows that the steplength $\alpha_k > 0$, satisfies:

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

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

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

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

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

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

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

With these we can write:

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

where

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

Let us denote:

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

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

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

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

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

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

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

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

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

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

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

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

Observe that if d_k is a descent direction, then

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

and from (4.12) we get:

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

Therefore, neglecting the contribution of ε_k, and considering $\gamma_k=-a_k/b_k$, we still get an improvement on the function values.

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

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

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

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

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

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

where $y_k=g_k-g_z$. Observe that for strictly convex functions $b_k>0$. However, if $b_k=0$, then the acceleration scheme doesn't have any effect by considering $\gamma_k=1$ in (4.5).

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

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

5. ACGHES algorithm

For large-scale problems, choices for the update parameter that do not require the evaluation of the Hessian matrix are often preferred in practice to the methods that require the Hessian. However, the presence of the Hessian in β_k recalls the open problem articulated by Nocedal [19]: *whether one can take advantage of the problem structure to design a more efficient nonlinear conjugate gradient iteration.* Indeed, our numerical experiments proved that even though the Hessian is partially separable (block diagonal) or it is a multi-diagonal matrix, the Hessian / vector product $\nabla^2 f(x_{k+1})s_k$ is time consuming, especially for large-scale problems. Therefore, in an effort to use the Hessian in β_k we suggest a nonlinear conjugate gradient algorithm in which the Hessian / vector product $\nabla^2 f(x_{k+1})s_k$ is approximated by finite differences:

$$\nabla^2 f(x_{k+1})s_k = \frac{\nabla f(x_{k+1} + \delta s_k) - \nabla f(x_{k+1})}{\delta}, \tag{5.1}$$

where

$$\delta = \frac{2\sqrt{\varepsilon_m}(1 + \|x_{k+1}\|)}{\|s_k\|}, \tag{5.2}$$

and ε_m is epsilon machine. The above forward difference approximation to $\nabla^2 f(x_{k+1})s_k$, with a careful choice of the finite difference interval, is generally satisfactory. Observe that the forward difference formula require one additional gradient evaluation. The choice of δ must balance the truncation errors. Besides, a number of precautions against division by small values of $\|s_k\|$ as well as some restrictions on upper / lower values on δ are also used, like in the truncated Newton TN package by Nash [18].
The ACGHES algorithm is as follows:

Step 1. Select the initial 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 $k = 0$. Select a value for the parameter ε.

Step 2. Test a criterion for stopping the iterations. For example, if $\|g_k\|_\infty \le \varepsilon$, then stop; otherwise continue with step 3.

Step 3. Using the Wolfe line search conditions (1.4) and (1.5) 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 $b_k \ne 0$, then compute $\gamma_k = -a_k / b_k$ and update the variables as $x_{k+1} = x_k + \gamma_k \alpha_k d_k$, otherwise update the variables as $x_{k+1} = x_k + \alpha_k d_k$. Compute f_{k+1} and g_{k+1}. Compute $s_k = x_{k+1} - x_k$.

Step 7. Determine δ as in (5.2) and compute $y_k = (\nabla f(x_{k+1} + \delta s_k) - \nabla f(x_{k+1}))/\delta$.

Step 8. Compute $\beta_k = (y_k^T g_{k+1} - s_k^T g_{k+1}) / s_k^T y_k$.

Step 9. Compute the search direction as $d_{k+1} = -g_{k+1} + \beta_k s_k$.

Step 10. Restart criterion. If the restart criterion of Powell $\left|g_{k+1}^T g_k\right| > 0.2\left\|g_{k+1}\right\|^2$ is satisfied, then set $d_{k+1} = -g_{k+1}$.

Step 11. 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 (1.4) and (1.5). 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 $\alpha_{k-1}\left\|d_{k-1}\right\| / \left\|d_k\right\|$. This selection was used for the first time by Shanno and Phua in CONMIN [22] and in SCALCG by Andrei [3-5].

Proposition 5.1. *Suppose that f is a uniformly convex function on the level set $S = \{x : f(x) \leq f(x_0)\}$, and d_k satisfies the sufficient descent condition $g_k^T d_k < -c_1\left\|g_k\right\|^2$, where $c_1 > 0$, and $c_3\left\|g_k\right\|^2 \leq \left\|d_k\right\|^2 \leq c_2\left\|g_k\right\|^2$, where $c_2, c_3 > 0$. Then the sequence generated by ACGHES converges linearly to x^*, solution to the problem (1.1).*

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

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

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

$$f(x_k + \gamma_m \alpha d_k) \leq f(x_k + \alpha d_k) - \frac{(a_k + b_k)^2}{2b_k}. \tag{5.3}$$

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

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

Therefore,

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

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

$$f(x_k + \alpha d_k) \leq f(x_k) - \frac{1}{2} c_1 \alpha \left\|g_k\right\|^2 \leq f(x_k) - \rho c_1 \alpha \left\|g_k\right\|^2, \tag{5.4}$$

since $\rho < 1/2$.

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

$$f(x_k + \alpha d_k) \leq f(x_k) - \rho c_1 \omega \|g_k\|^2 . \tag{5.5}$$

On the other hand,

$$\frac{(a_k + b_k)^2}{2b_k} \geq \frac{\left(-c_1 \|g_k\|^2 + \omega m c_3 \|g_k\|^2\right)^2}{2Mc_2 \|g_k\|^2} = \frac{(\omega m c_3 - c_1)^2}{2\omega M c_2} \|g_k\|^2 . \tag{5.6}$$

Considering (5.5) and (5.6) from (5.3) we get:

$$f(x_k + \gamma_m \alpha d_k) \leq f(x_k) - \rho c_1 \omega \|g_k\|^2 - \frac{(\omega m c_3 - c_1)^2}{2\omega M c_2} \|g_k\|^2 . \tag{5.7}$$

Therefore

$$f(x_k) - f(x_k + \gamma_m \alpha d_k) \geq \left[\rho c_1 \omega + \frac{(\omega m c_3 - c_1)^2}{2\omega M c_2} \right] \|g_k\|^2 .$$

But, $f(x_k) - f(x_{k+1}) \to 0$ and as a consequence g_k goes to zero, i.e. x_k converges to x^*. Having in view that $f(x_k)$ is a nonincreasing sequence, it follows that $f(x_k)$ converges to $f(x^*)$. From (5.7) we see that

$$f(x_{k+1}) \leq f(x_k) - \left[\rho c_1 \omega + \frac{(\omega m c_3 - c_1)^2}{2\omega M c_2} \right] \|g_k\|^2 . \tag{5.8}$$

Combining this with $\|g_k\|^2 \geq 2m(f(x_k) - f^*)$ and subtracting f^* from both sides of (5.8) we conclude:

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

where

$$c = 1 - 2m \left[\rho c_1 \omega + \frac{(\omega m c_3 - c_1)^2}{2\omega M c_2} \right] < 1.$$

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

6. Numerical results and comparisons

In this section we report some numerical results obtained with a Fortran implementation of the ACGHES algorithm. The code is 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 [1] (some from CUTE library [7]). For each test function we have taken ten numerical experiments with the number of variables $n = 1000, 2000, ..., 10000$. The algorithm implements the Wolfe line search conditions with $\rho = 0.0001$ and $\sigma = 0.9$, and also the same stopping criterion $\|g_k\|_\infty \leq 10^{-6}$, where $\|.\|_\infty$ is the maximum absolute component of a vector. In step 7 the computation of δ is implemented as:

$$\delta = \max\left\{\frac{\varphi}{\max\{10\varphi, \|s_k\|\}}, \frac{\varphi}{100}\right\}, \quad \varphi = 2\sqrt{\varepsilon_m}(1+\|x_{k+1}\|\sqrt{n}).$$

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

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

and the number of iterations, or the number of function-gradient evaluations, or the CPU time of ALG1 was less than the number of iterations, or the number of function-gradient evaluations, or the CPU time corresponding to ALG2, respectively.

In the first set of numerical experiments we compare ACGHES versus some conjugate gradient algorithms. Figures 1-6 present the Dolan and Moré [12] CPU performance profile of ACGHES versus Hestenes-Stiefel ($\beta_k^{HS} = \frac{y_k^T g_{k+1}}{y_k^T s_k}$) [15], Polak-Ribière-Polyak ($\beta_k^{PRP} = \frac{y_k^T g_{k+1}}{g_k^T g_k}$) [20,21], Dai-Yuan ($\beta_k^{DY} = \frac{g_{k+1}^T g_{k+1}}{y_k^T s_k}$) [9], Dai-Liao ($t = 1$) ($\beta_k^{DL} = \frac{g_{k+1}^T (y_k - t s_k)}{y_k^T s_k}$) [8], hybrid Dai-Yuan ($\beta_k^{hDY} = max\left\{c\beta^{DY}, min\left\{\beta^{HS}, \beta^{DY}\right\}\right\}$, $c = -(1-\sigma)/(1+\sigma)$) [9], and new conjugacy condition and related new conjugate gradient by Li, Tang and Wei [16] ($\beta_k = \max\left\{\frac{g_{k+1}^T \hat{y}_k^*}{s_k^T \hat{y}_k^*}, 0\right\} - t\frac{g_{k+1}^T s_k}{s_k^T \hat{y}_k^*}$, $\hat{y}_k^* = y_k + \frac{\max\{\theta_k, 0\}}{\|s_k\|^2} s_k$, $\theta_k = 2(f_k - f_{k+1}) + (g_k + g_{k+1})^T s_k$, $y_k = g_{k+1} - g_k$, $t = 0.1$), respectively.

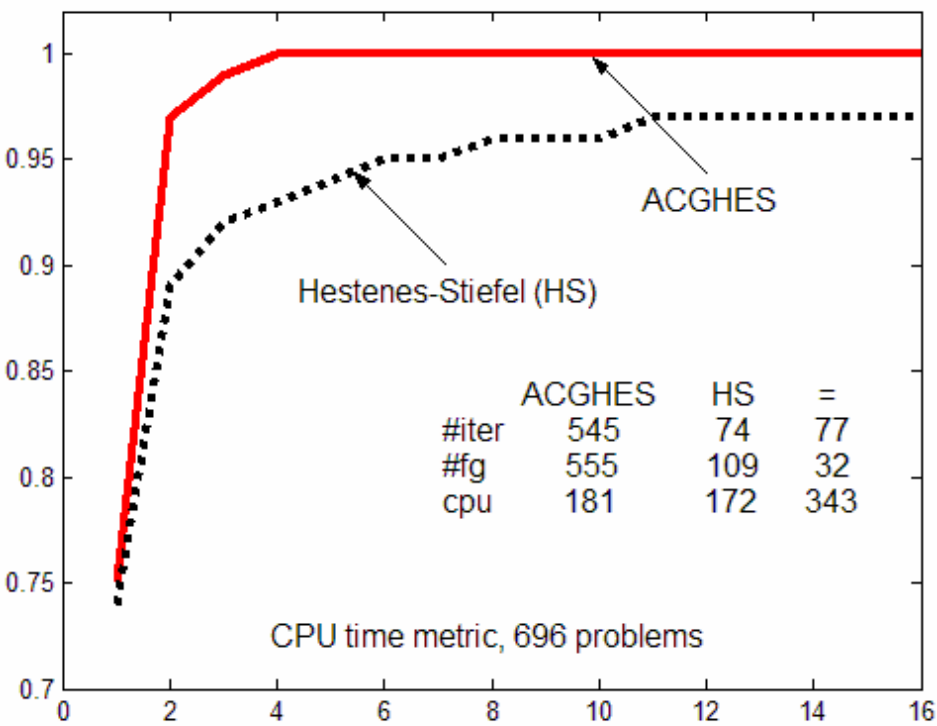

Fig. 1. ACGHES versus Hestenes-Stiefel.

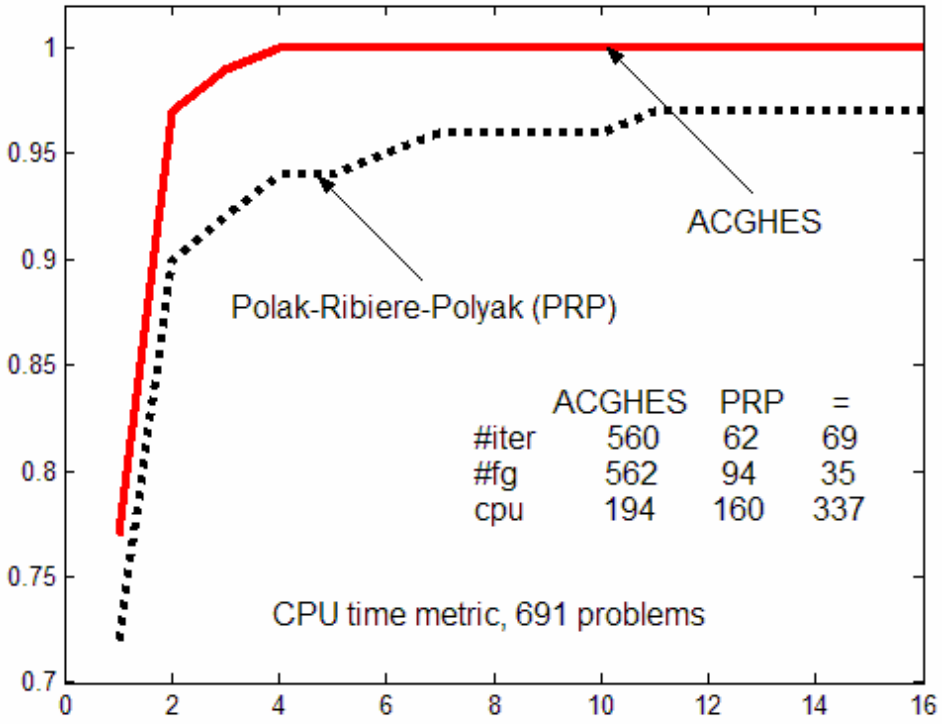

Fig. 2. ACGHES versus Polak-Ribière-Polyak.

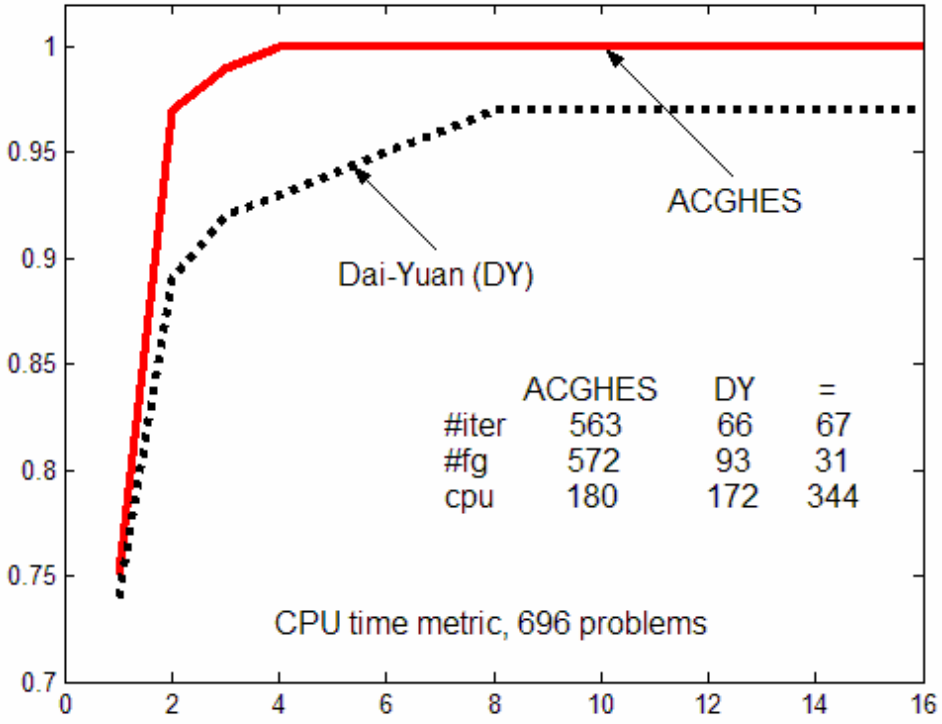

Fig. 3. ACGHES versus Dai-Yuan.

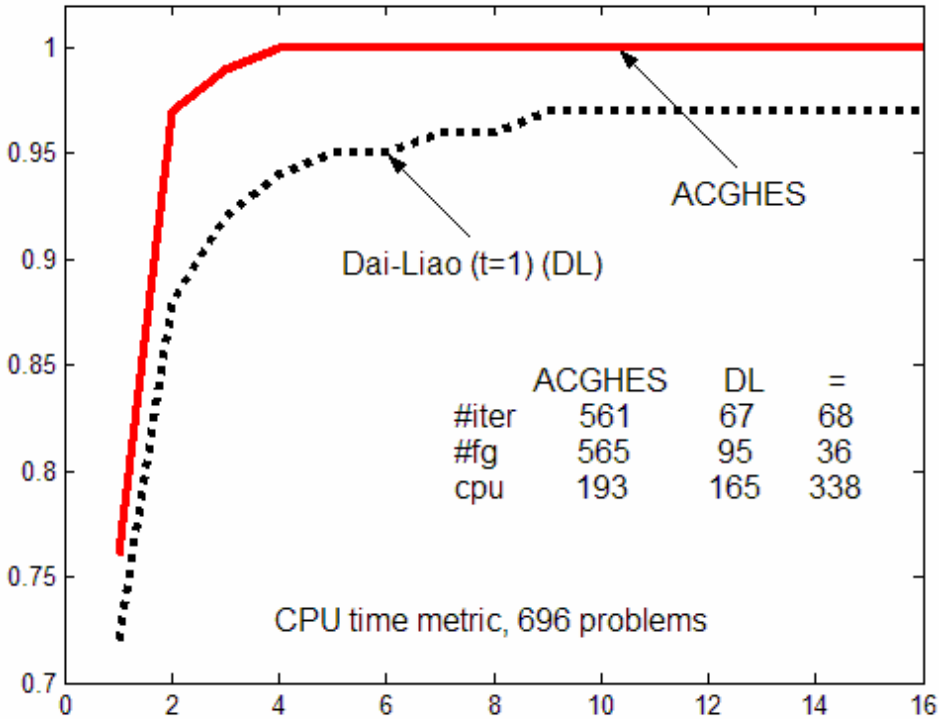

Fig. 4. ACGHES versus Dai-Liao (t=1).

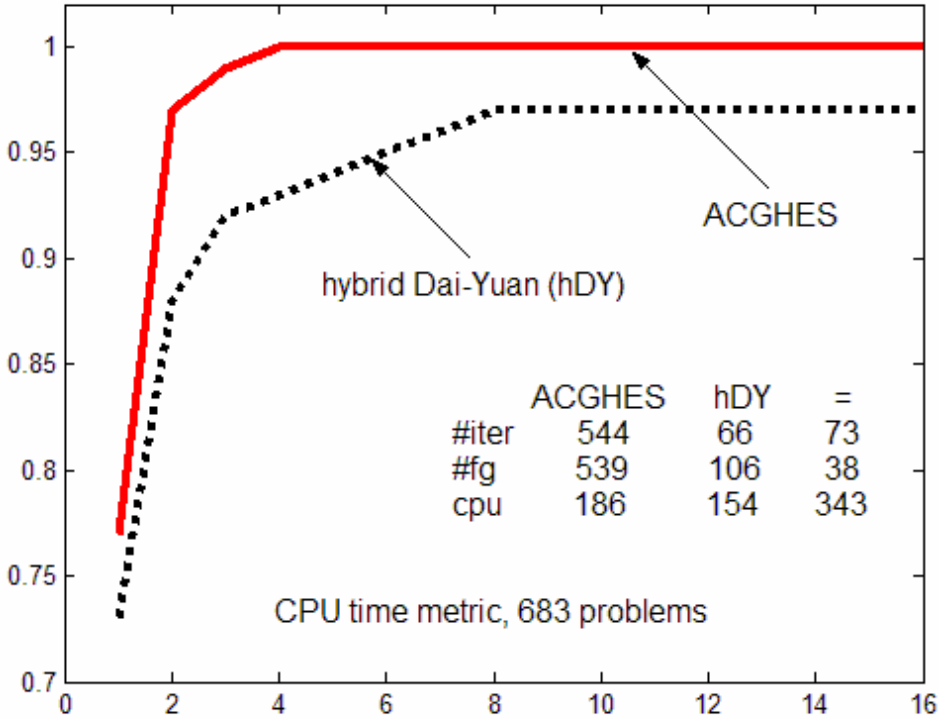

Fig. 5. ACGHES versus hybrid Dai-Yuan (hDY).

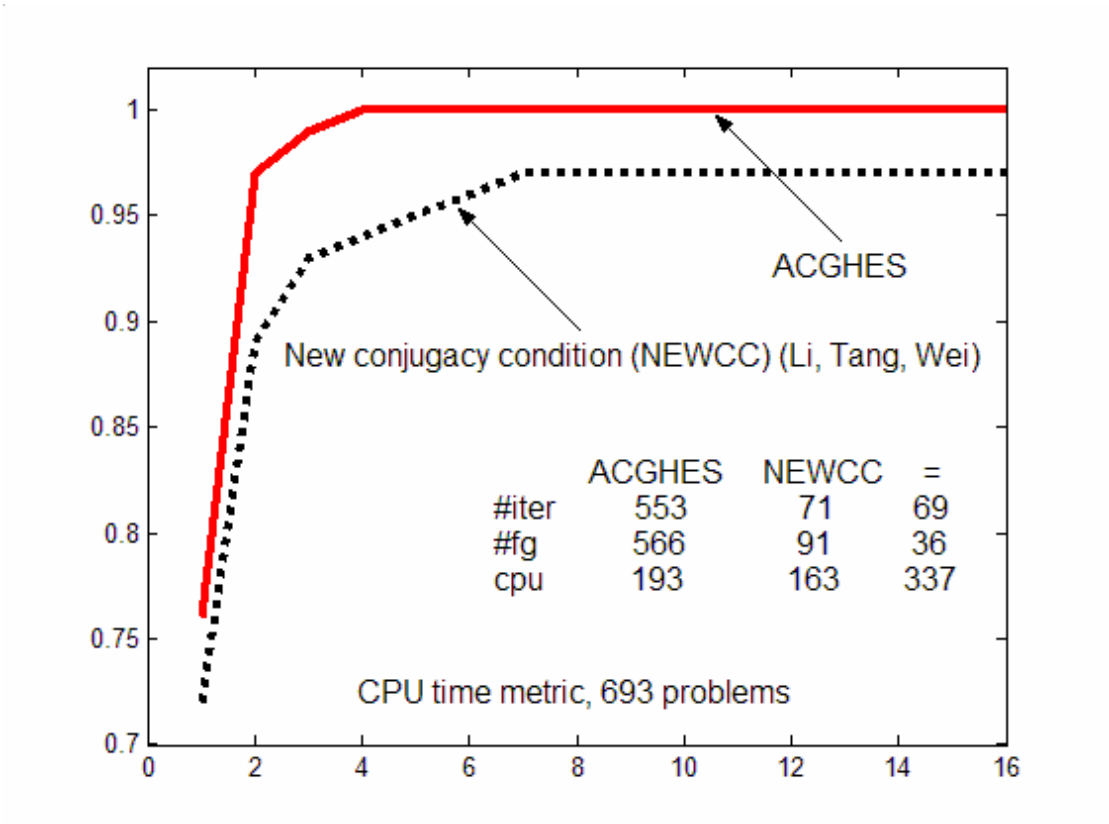

Fig. 6. ACGHES versus new conjugacy condition and related new conjugate gradient (NEWCC).

The percentage of the test problems for which a method is the fastest is given on the left axis of the plot. The right side of the plot gives the percentage of the test problems that were successfully solved by these algorithms, respectively. Mainly, the right side is a measure of the robustness of an algorithm.

When comparing ACGHES with all these conjugate gradient algorithms subject to CPU time metric we see that ACGHES is top performer, i.e. the accelerated conjugate gradient algorithm with forward-difference approximation to Hessian / vector product is more successful and more robust than the considered conjugate gradient algorithms. For example, when comparing ACGHES with Hestene-Stiefel (HS) (see Figure 1), subject to the number of iterations, we see that ACGHES was better in 545 problems (i.e. it achieved the minimum number of iterations in 545 problems). HS was better in 74 problems and they achieved the same number of iterations in 77 problems, etc. Out of 750 problems, only for 696 problems does the criterion (6.1) hold. Observe that in contrast with NEWCC which uses not only the gradient value information but also the function value information in two successive points, the ACGHES algorithm besides the gradient value information uses also the Hessian of function f in an indirect manner through a finite difference approximation of Hessian / vector product. This is the reason why ACGHES outperform NEWCC, even that NEWCC uses a new highly elaborated quasi-Newton equation.

Numerical experiments proved that for the vast majority of iterations $\gamma_k = -a_k / b_k < 1$, i.e. the acceleration scheme has the propensity to reduce the values of the step lengths.

In the second set of numerical experiments, Figures 7 and 8 we compare ACGHES to the conjugate gradient algorithms SCALCG by Andrei [3-5], and CONMIN by Shanno and Phua [22].

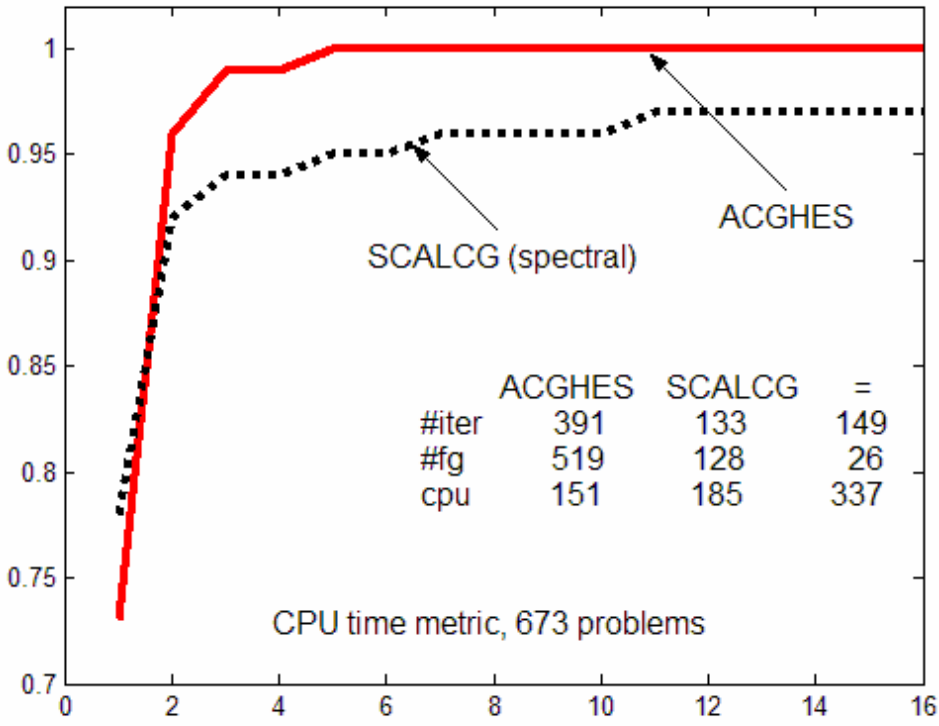

Fig. 7 ACGHES versus SCALCG spectral (Andrei).

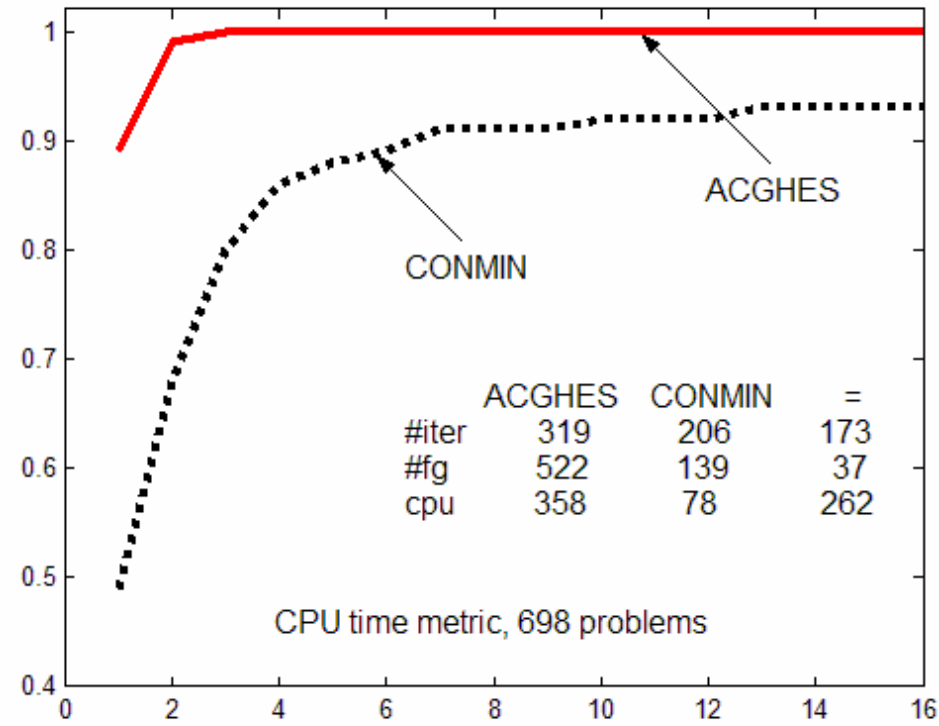

Fig. 8 ACGHES versus CONMIN (Shanno-Phua).

In Figures 7 and 8 we have computational evidence that the ACGHES algorithm is more robust than the BFGS preconditioned conjugate gradient algorithms SCALCG and CONMIN. Even though SCALCG and CONMIN take a lot from the quasi-Newton methods we see that

this conjugate gradient algorithm with forward difference approximation of Hessian / vector product and acceleration scheme is far more efficient.
Finally, in the third set of numerical experiments, in Figure 9, we compare ACGHES to TN by Nash [18] where again a finite difference approximation of Hessian / vector is used.

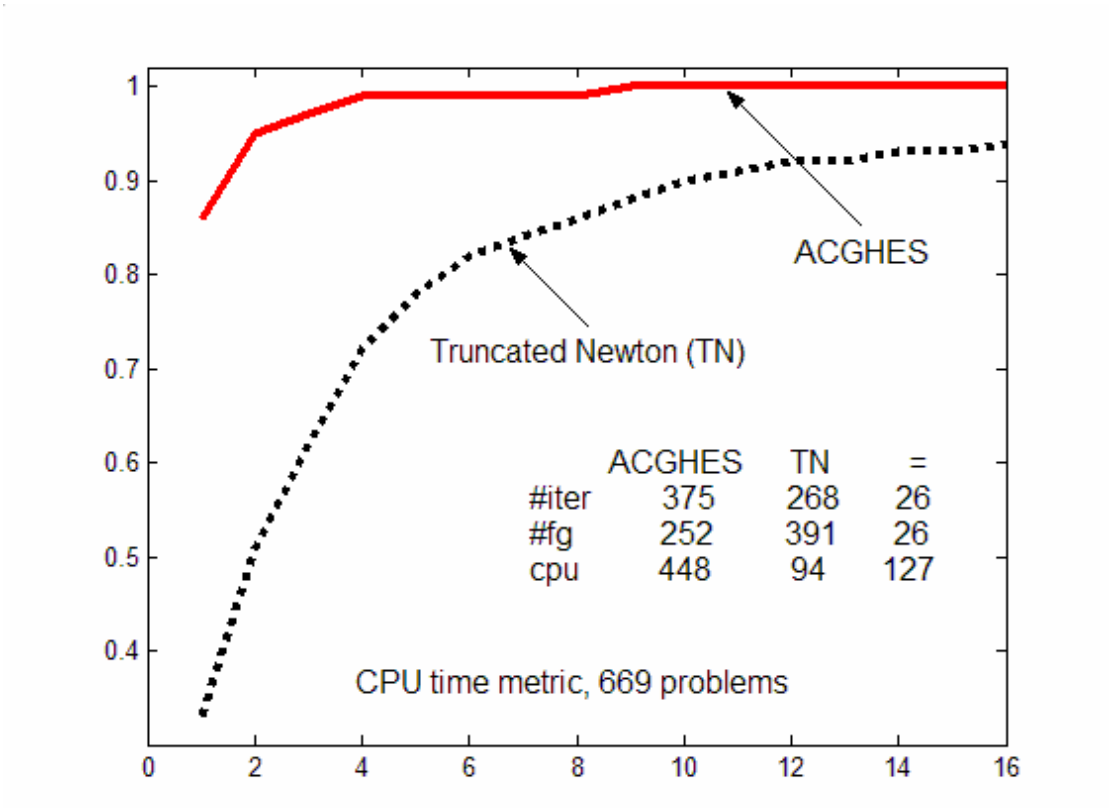

Fig. 9 ACGHES versus Truncated Newton TN (Nash).

From Figure 9 we see that the truncated Newton algorithm in TN implementation given by Nash [18] is clearly outperformed by ACGHES. The Hessian / vector product in TN is approximated by the forward finite difference in which δ is computed as $\delta = \sqrt{\varepsilon_m}(1+\|x_{k+1}\|)$.

7. Conclusion

We have presented a new conjugate gradient algorithm for solving large-scale unconstrained optimization problems. The algorithm exploits the presence of the Hessian in the formula for β_k computation as well as the fact that the step lengths in conjugate gradient algorithms differ from 1 in the vast majority of iterations. The algorithm approximates the Hessian / vector product by means of the forward finite difference in combination with a careful choice of the finite difference interval. It modifies the step length by an acceleration scheme which proved to be very efficient in reducing the values of the minimizing function along the iterations. We proved that the direction is a descent one, and the algorithm is globally convergent. For uniformly convex functions the convergence of the accelerated scheme is still linear, but the reduction in function values is significantly improved. For a test set consisting of 750 problems with dimensions ranging between 1000 and 10,000, the CPU time performance profiles of ACGHES was higher than those of HS, PRP, DY, DL (t=1), hDY, NEWCC, SCALCG, CONMIN and TN. The acceleration scheme is an important ingredient for the efficiency of the algorithm.

References

[1] N. Andrei, An unconstrained optimization test functions collection. Advanced Modeling and Optimization. An Electronic International Journal, 10 (2008) 147-161.

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

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

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

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

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

[7] I. Bongartz, A.R. Conn, N.I.M. Gould, Ph.L. Toint, CUTE: constrained and unconstrained testing environments, ACM Trans. Math. Software 21 (1995) 123-160.

[8] Y.H. Dai, L.Z. Liao, New conjugacy conditions and related nonlinear conjugate gradient methods. Appl. Math. Optim., 43 (2001) 87-101.

[9] Y.H. Dai, Y. Yuan, An efficient hybrid conjugate gradient method for unconstrained optimization, Ann. Oper. Res., 103 (2001) 33-47.

[10] Y.H. Dai, J.Y. Han, G.H. Liu, D.F. Sun, X. Yin, Y. Yuan, Convergence properties of nonlinear conjugate gradient methods. SIAM Journal on Optimization 10 (1999) 348-358.

[11] J.W. Daniel, The conjugate gradient method for linear and nonlinear operator equations. SIAM J. Numer. Anal., 4 (1967) 10-26.

[12] E.D. Dolan, J.J. Moré, Benchmarking optimization software with performance profiles, Math. Programming, 91 (2002) 201-213.

[13] A.A. Goldstein, On steepest descent, SIAM J. Control, 3 (1965) 147-151.

[14] W.W. Hager, H. Zhang, A survey of nonlinear conjugate gradient methods. Pacific journal of Optimization, 2 (2006) 35-58.

[15] M.R. Hestenes, E.L. Stiefel, Methods of conjugate gradients for solving linear systems, J. Research Nat. Bur. Standards, 49 (1952) 409-436.

[16] G. Li, C. Tang, Z. Wei, New conjugacy condition and related new conjugate gradient methods for unconstrained optimization. Journal of Computational and Applied Mathematics, 202 (2007) 523-539.

[17] D.C. Liu, J. Nocedal, On the limited memory BFGS method for large scale optimization methods. Mathematical Programming, 45 (1989), pp. 503-528.

[18] S.G. Nash, Preconditioning of truncated-Newton methods. SIAM J. on Scientific and Statistical Computing, 6 (1985) 599-616.

[19] J. Nocedal, Conjugate gradient methods and nonlinear optimization. In Linear and nonlinear Conjugate Gradient related methods, L. Adams and J.L. Nazareth (eds.), SIAM, 1996, pp.9-23.

[20] E. Polak, G. Ribière, Note sur la convergence de directions conjuguée, Rev. Francaise Informat Recherche Operationelle, 3e Année 16 (1969) 35-43.

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

[22] D.F. Shanno, K.H. Phua, Algorithm 500, Minimization of unconstrained multivariate functions, ACM Trans. on Math. Soft., 2 (1976) 87-94.

[23] P. Wolfe, Convergence conditions for ascent methods, SIAM Rev. 11 (1969) 226-235.

[24] P. Wolfe, Convergence conditions for ascent methods II: some corrections, SIAM Rev. 13 (1971) 185-188.

March 4, 2008