Annals of the Academy of Romanian Scientists
Series on Science and Technology of Information
ISSN 2066-8526 Volume **8**, Number **1**/**2015**

A THREE-TERM DESCENT CONJUGATE GRADIENT ALGORITHM USING THE MINIMIZATION OF THE TWO-PARAMETRIC QUADRATIC MODEL FOR LARGE-SCALE UNCONSTRAINED OPTIMIZATION

Neculai ANDREI[1]

Abstract. *A three-term descent conjugate gradient algorithm is presented. The algorithm is obtained by minimizing the two-parameter quadratic model of the objective function in which the symmetrical approximation of the Hessian matrix satisfies the general quasi-Newton equation. Using the general quasi-Newton equation the search direction includes a parameter* ω *which is determined by the formal equality between the search direction used in the suggested algorithm and the Newton direction. It is proved that the best value of this parameter is* $\omega = 1$. *The direction satisfies both the descent and the conjugacy conditions. The new approximation of the minimum is obtained by the general Wolfe line search using by now a standard acceleration technique. Under standard assumptions, both for uniformly convex functions and for general nonlinear functions, the global convergence of the algorithm is proved. The numerical experiments using a collection of 800 large-scale unconstrained optimization test problems of different complexity show that using these ingredients we get a search direction able to define a very efficient and robust three-term conjugate gradient algorithm. Numerical comparison of this algorithm versus well known conjugate gradient algorithms ASCALCG, CONMIN, AHYBRIDM, CG-DESCENT, THREECG and TTCG as well as the limited memory quasi-Newton algorithm LBFGS (m=5) and the truncated Newton TN show that our algorithm is more efficient and more robust.*

Keywords: Large scale unconstrained optimization, Two parameters quadratic model, Generalized secant equation, Conjugate gradient algorithms, Numerical comparisons

1. Introduction

For solving large-scale unconstrained optimization problems

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

where $f: R^n \to R$ is a continuously differentiable function, supposed to be bounded from below, starting from an initial guess $x_0 \in R^n$, a three-term conjugate gradient method we want to develop in this paper, generates the sequence $\{x_k\}$ as:

[1]Senior. Res. Dr. Eng. Research Institute for Informatics, Center for Advanced Modeling and Optimization, 8-10, Averescu Avenue, Bucharest 1, Romania; corresponding member of Academy of Romanian Scientists (nandrei@ici.ro).

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

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

$$d_{k+1} = -g_{k+1} + a_k s_k + b_k y_k, \ d_0 = -g_0. \tag{1.3}$$

In (1.3), a_k and b_k are known as three-term conjugate gradient parameters or coefficients. As usual $s_k = x_{k+1} - x_k$, $g_k = \nabla f(x_k)$ and $y_k = g_{k+1} - g_k$. Observe that, the search direction d_{k+1} is computed as a linear combination of $-g_{k+1}$, s_k and y_k, in which the coefficient of g_{k+1} is -1. The line search in the conjugate gradient algorithms is often based on the general Wolfe conditions [36, 37]:

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

$$g_{k+1}^T d_k \ge \sigma g_k^T d_k, \tag{1.5}$$

where d_k is a descent direction and $0 < \rho \le \sigma < 1$. However, for some conjugate gradient algorithms stronger version of the Wolfe line search conditions, given by (1.4) and

$$\left| g_{k+1}^T d_k \right| \le -\sigma g_k^T d_k, \tag{1.6}$$

are needed to ensure the convergence and to enhance the stability.

Different three-term conjugate gradient algorithms correspond to different choices for the scalar parameters a_k and b_k. In this context the papers by Beale [10], McGuire and Wolfe [21], Deng and Li [14] and Dai and Yuan [13], Nazareth [26], Zhang, Zhou and Li [38, 39], Zhang, Xiao and Wei [40], Al-Bayati and Sharif [1], Cheng [11], Narushima, Yabe and Ford [23], Andrei [7, 8, 9] present different versions of three-term conjugate gradient algorithms together with their properties, global convergence and numerical performances. All these three-term conjugate gradient algorithms are obtained by modification of classical conjugate gradient algorithms to satisfy the descent and in some cases the conjugacy conditions. Generally, these three-term conjugate gradient algorithms are more efficient and more robust than classical conjugate gradient algorithms by Hestenes and Stiefel [19] or by Fletcher and Reeves [16], Polak-Ribière-Polyak [30, 31], Liu and Storey [20] or by Dai and Yuan [13], etc.

In this paper we suggest another way to get three-term conjugate gradient algorithms by minimizing the two-parameters quadratic model of the function f. The idea is to consider the quadratic approximation of the function f in the current point and to determine the search direction (1.3) by minimization of this

quadratic model subject to the parameters a_k and b_k. It is assumed that the symmetrical approximation of the Hessian matrix satisfies the general quasi-Newton equation which depends by a positive parameter ω. The three-term conjugate gradient parameters a_k and b_k are determined as solution of an algebraic system of two linear equation. Section 2 presents this idea, as well as the procedure for determination of the positive parameter ω. In order to determine a good value for the parameter ω the formal equality between the search direction (1.3) and the best known direction given by the Newton direction is used. In section 3 the corresponding three-term conjugate gradient TTSCAL algorithm is presented into the context of acceleration of the iterations. Section 4 is dedicated to the global convergence analysis of the algorithm. It is shown that under the standard assumptions both for uniformly convex functions and for general functions the search direction is bounded. Section 5 includes the numerical results with TTSCAL and some comparisons versus known conjugate gradient algorithms ASCALCG [2, 3], CONMIN [35], AHYBRIDM [6], CG-DESCENT [18], THREECG [9], TTCG [8] as well as versus LBFGS [27] and TN [24], on a collection of 800 large-scale unconstrained optimization test functions. It is shown that the three-term conjugate gradient algorithm TTSCAL corresponding to the minimization of the two-parameters quadratic model of the minimizing function f is more efficient and more robust then all these algorithms considered in this numerical study.

The two-parameters quadratic model of function f minimization

At the $k-$th iteration of the algorithm, let us assume that an inexact Wolfe line search was executed, that is the step-length α_k satisfying (1.4) and (1.5) was computed. With this value of α_k the following elements $s_k = x_{k+1} - x_k$ and $y_k = g_{k+1} - g_k$ can be computed. Now, let us consider the following quadratic approximate of function f in x_{k+1} as:

$$\Phi_{k+1}(d) = g_{k+1}^T d + \frac{1}{2} d^T B_{k+1} d, \tag{2.1}$$

where B_{k+1} is a symmetrical and positive definite approximation of the Hessian $\nabla^2 f(x_{k+1})$ and d is the direction which follows to be determined. The direction d_{k+1} is computed as:

$$d_{k+1} = -g_{k+1} + a_k s_k + b_k y_k, \tag{2.2}$$

where the scalars a_k and b_k are determined as solution of the following minimizing problem:

$$\min_{a_k, b_k \in R} \Phi_{k+1}(d_{k+1}). \tag{2.3}$$

Introducing d_{k+1} from (2.2) in the minimizing problem (2.3), then the parameters a_k and b_k are determined as solution of the following linear algebraic system:

$$a_k(s_k^T B_{k+1} s_k) + b_k(s_k^T B_{k+1} y_k) = g_{k+1}^T B_{k+1} s_k - s_k^T g_{k+1}, \tag{2.4a}$$

$$a_k(s_k^T B_{k+1} y_k) + b_k(y_k^T B_{k+1} y_k) = g_{k+1} B_{k+1} y_k - y_k^T g_{k+1}. \tag{2.4b}$$

Suppose that the symmetrical and positive definite matrix B_{k+1} is an approximation of the Hessian $\nabla^2 f(x_{k+1})$ such that $B_{k+1} s_k = \omega^{-1} y_k$, with $\omega > 0$, known as the general quasi-Newton equation. With this the linear algebraic system (2.4) becomes:

$$a_k(y_k^T s_k) + b_k \|y_k\|^2 = y_k^T g_{k+1} - \omega s_k^T g_{k+1}, \tag{2.5a}$$

$$a_k \|y_k\|^2 + b_k \omega(y_k^T B_{k+1} y_k) = \omega g_{k+1}^T B_{k+1} y_k - \omega y_k^T g_{k+1}. \tag{2.5b}$$

In order to solve the linear system (2.5) we must evaluate the quantities: $\eta_k \equiv y_k^T B_{k+1} y_k$ and $\theta_k \equiv g_{k+1}^T B_{k+1} y_k$. Suppose that B_{k+1} is positive definite. Now, using the classical quasi-Newton equation $B_{k+1} s_k = y_k$ we have:

$$\eta_k = y_k^T B_{k+1} y_k = \frac{y_k^T B_{k+1} y_k s_k^T B_{k+1} s_k}{(y_k^T B_{k+1} s_k)^2} \frac{(y_k^T B_{k+1} s_k)^2}{s_k^T B_{k+1} s_k}$$

$$= \frac{y_k^T B_{k+1}^{\frac{1}{2}} B_{k+1}^{\frac{1}{2}} y_k s_k^T B_{k+1}^{\frac{1}{2}} B_{k+1}^{\frac{1}{2}} s_k}{(y_k^T B_{k+1}^{\frac{1}{2}} B_{k+1}^{\frac{1}{2}} s_k)^2} \frac{(y_k^T y_k)^2}{y_k^T s_k}$$

$$= \frac{\left\| B_{k+1}^{\frac{1}{2}} y_k \right\|^2 \left\| B_{k+1}^{\frac{1}{2}} s_k \right\|^2}{\left((B_{k+1}^{\frac{1}{2}} y_k)^T (B_{k+1}^{\frac{1}{2}} s_k) \right)^2} \frac{(y_k^T y_k)^2}{y_k^T s_k} = \frac{1}{\cos^2 < B_{k+1}^{\frac{1}{2}} y_k, B_{k+1}^{\frac{1}{2}} s_k >} \frac{(y_k^T y_k)^2}{y_k^T s_k}. \tag{2.6}$$

Since B_{k+1} is unknown, it follows that the quantity $\cos^2 < B_{k+1}^{\frac{1}{2}} y_k, B_{k+1}^{\frac{1}{2}} s_k >$ in (2.6) is unknown. However, since the mean value of $\cos^2 \xi = 1/2$, then in (2.6) it seems

reasonable to replace the above quantity $\cos^2 < B_{k+1}^{\frac{1}{2}} y_k, B_{k+1}^{\frac{1}{2}} s_k >$ by 1/2. Therefore, η_k can be computed as:

$$\eta_k = 2\frac{(y_k^T y_k)^2}{y_k^T s_k}. \tag{2.7}$$

Now, in order to compute θ_k we can use the BFGS update initialized with the identity matrix:

$$\theta_k = g_{k+1}^T B_{k+1} y_k = g_{k+1}^T \left[I + \frac{y_k y_k^T}{y_k^T s_k} - \frac{s_k s_k^T}{s_k^T s_k} \right] y_k$$

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

It is worth saying that another way to compute θ_k is to use the BFGS update initialized, for example, from the scaling matrix $((s_k^T y_k)/\|s_k\|^2)I$. However, we are interested to use (2.8) in our algorithm. With these developments the linear algebraic system (2.5) becomes:

$$a_k (y_k^T s_k) + b_k \|y_k\|^2 = y_k^T g_{k+1} - \omega s_k^T g_{k+1}, \tag{2.9a}$$

$$a_k \|y_k\|^2 + b_k \omega \eta_k = \omega \theta_k - \omega y_k^T g_{k+1}. \tag{2.9b}$$

Now, using (2.7) observe that the determinant of the matrix of the linear system (2.9) is:

$$\Delta_k = \omega \eta_k (y_k^T s_k) - (y_k^T y_k)^2 = (2\omega - 1)(y_k^T y_k)^2 > 0 \tag{2.10}$$

if $\omega > 1/2$ and of course $y_k \neq 0$.

Supposing that $\Delta_k > 0$, then from the linear algebraic system (2.9) we get:

$$a_k = \frac{1}{\Delta_k} \left[\omega \eta_k (y_k^T g_{k+1} - \omega s_k^T g_{k+1}) - \omega \|y_k\|^2 (\theta_k - y_k^T g_{k+1}) \right], \tag{2.11}$$

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

Therefore, if $\Delta_k > 0$, i.e. $y_k \neq 0$ and $\omega > 1/2$, then the search direction is computed as in (2.2), where the scalars a_k and b_k are computed as in (2.11) and (2.12) respectively.

If the line search is exact, that is $s_k^T g_{k+1} = 0,$ then from (2.11) and (2.12) we have

$$a_k = \frac{\omega}{2\omega - 1} \frac{y_k^T g_{k+1}}{y_k^T s_k}, \tag{2.13}$$

$$b_k = \frac{\omega - 1}{2\omega - 1} \frac{y_k^T g_{k+1}}{y_k^T s_k}. \tag{2.14}$$

Observe that if $\omega = 1$, then $a_k = (y_k^T g_{k+1})/(y_k^T s_k)$ and $b_k = 0,$ i.e. the search direction is computed as:

$$d_{k+1} = -g_{k+1} + \frac{y_k^T g_{k+1}}{y_k^T s_k} s_k, \tag{2.15}$$

which is exactly the Hestenes and Stiefel conjugate gradient algorithm.

Proposition 2.1. *Suppose that* $B_{k+1} > 0.$ *Then* d_{k+1} *given by (2.2) where the scalars* a_k *and* b_k *are computed as in (2.11) and (2.12) respectively is a descent direction.*

Proof. From (2.1) observe that $\Phi_{k+1}(0) = 0.$ Since $B_{k+1} > 0$ and d_{k+1} given by (2.2), (2.11) and (2.12) is the solution of (2.3), it follows that $\Phi_{k+1}(d_{k+1}) \le 0.$ Therefore,

$$g_{k+1}^T d_{k+1} \le -\frac{1}{2} d_{k+1}^T B_{k+1} d_{k+1} < 0, \tag{2.16}$$

i.e. d_{k+1} is a descent direction. ■

Proposition 2.2. *Suppose that the search direction* d_{k+1} *is given by (2.2) where the scalars* a_k *and* b_k *satisfy the linear algebraic system (2.9). Then the direction* d_{k+1} *satisfies the Dai-Liao conjugacy condition* $y_k^T d_{k+1} = -\omega s_k^T g_{k+1},$ *with* $\omega > 0.$

Proof. Since d_{k+1} is given by (2.2) it follows that $y_k^T d_{k+1}$ is given by (2.9a), which is exactly the Dai-Liao conjugacy condition [12]. ■

Observe that our algorithm in which the search direction is given by (2.2) where the scalars a_k and b_k are computed as in (2.11) and (2.12) respectively and the step-length is obtained by the Wolfe line search (1.4) and (1.5) is a conjugate gradient algorithm with three terms. Our three-term conjugate gradient algorithm

is based on the minimization of the quadratic approximation of the function f into the current point, in which the searching direction d_{k+1} is selected as a linear combination of $-g_{k+1}$, s_k and y_k where the coefficient of g_{k+1} is -1.

Remark 2.1. Another possibility to define the search direction is to introduce in (2.2) a scalar or a matrix coefficient multiplying g_{k+1} as:

$$d_{k+1} = -H_{k+1}g_{k+1} + a_k s_k + b_k y_k. \tag{2.17}$$

However, this is not as good as, for example, the limited memory quasi-Newton methods since if H_{k+1} contains enough useful information about the inverse Hessian of the function f, we are better off using the search direction $d_{k+1} = -H_{k+1}g_{k+1}$. The addition of the last two terms in (2.17) may prevent d_{k+1} from being a descent direction, without saying anything about the definition of the symmetric and positive definite matrix H_{k+1} which requires several vectors, making the storage requirements similar to those of limited memory methods. This is the main reason we consider in our three-term conjugate gradient algorithm the search direction as in (2.2).

Observe that in order to define the search direction (2.2) in (2.11) and (2.12) we must establish a procedure for computation of the parameter ω. There are some possibilities, but in this paper we are interested to use the best search direction we know, i.e. the Newton direction. As a matter of fact, when the initial point x_0 is enough close to the local minimum point x^*, then the best search direction to be used in the current point x_{k+1} is the Newton direction $-\nabla^2 f(x_{k+1})^{-1} g_{k+1}$. Therefore, our motivation is to select the coefficients a_k and b_k in (2.2) in such a manner that for every $k \geq 1$ the direction d_{k+1} is the best search direction we know, i.e. the Newton direction. Hence, the parameter ω in a_k and b_k coefficients from (2.2) can be determined by the relation

$$-g_{k+1} + a_k s_k + b_k y_k = -\nabla^2 f(x_{k+1})^{-1} g_{k+1}. \tag{2.18}$$

Introducing the algebraic expressions of a_k and b_k from (2.11) and (2.12) respectively in (2.18), after some simple algebra we get the following quadratic equation:

$$2\omega^2 - 3\omega + 1 = 0, \tag{2.19}$$

which admits the following solutions: $\omega = 1$ and $\omega = 1/2$.

The solution $\omega = 1/2$ is not suitable (see the condition (2.10)). The only solution admitted is $\omega = 1$. Observe that the Newton direction is being used here only as a simple technical ingredient to compute a good value for the parameter ω.

TTSCAL algorithm

In this section we present the algorithm TTSCAL where the search direction is given by (2.2) and the coefficients a_k and b_k are computed as in (2.11) and (2.12) respectively with $\omega = 1$. In our algorithm we use an acceleration scheme we have presented in [5]. Basically the acceleration scheme modifies the step length α_k in a multiplicative manner to improve the reduction of the function values along the iterations. As in [5], in accelerated algorithm instead of (1.2) the new estimation of the minimum point is computed as

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

where

$$\xi_k = -\frac{\bar{a}_k}{\bar{b}_k}, \tag{3.2}$$

$\bar{a}_k = \alpha_k g_k^T d_k$, $\bar{b}_k = -\alpha_k (g_k - g_z)^T d_k$, $g_z = \nabla f(z)$ and $z = x_k + \alpha_k d_k$.

Hence, if $\bar{b}_k > 0$, then the new estimation of the solution is computed as $x_{k+1} = x_k + \xi_k \alpha_k d_k$, otherwise $x_{k+1} = x_k + \alpha_k d_k$. Observe that $\bar{b}_k = \alpha_k (g_z - g_k)^T d_k = \alpha_k (d_k^T \nabla^2 f(\bar{x}_k) d_k)$, where $\bar{x}_k$ is a point on the line segment connecting x_k and z. Since $\alpha_k > 0$, it follows that for convex functions $\bar{b}_k \geq 0$.

For uniformly convex functions, the linear convergence of the acceleration scheme is proved in [5].

Therefore, taking into consideration this acceleration scheme and using the definitions of g_k, s_k and y_k the following three-term conjugate gradient algorithm can be presented.

TTSCAL algorithm

Step 1.	Select a starting point $x_0 \in dom\, f$ and compute: $f_0 = f(x_0)$ and $g_0 = \nabla f(x_0)$. Select some positive values for ρ and σ. Set $d_0 = -g_0$ and $k = 0$. Consider $\omega = 1$.
Step 2.	Test a criterion for stopping the iterations. If the test is satisfied, then stop; otherwise continue with step 3.

Step 3.	Determine the steplength α_k using the Wolfe line search conditions (1.4) and (1.5).
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: $\bar{a}_k = \alpha_k g_k^T d_k$, and $\bar{b}_k = -\alpha_k y_k^T d_k$.
Step 6.	Acceleration scheme. If $\bar{b}_k > 0$, then compute $\xi_k = -\bar{a}_k / \bar{b}_k$ and update the variables as $x_{k+1} = x_k + \xi_k \alpha_k d_k$, 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.	If $y_k^T y_k > 0$, then compute a_k and b_k as in (2.11) and (2.12) respectively, otherwise set $a_k = (y_k^T g_{k+1})/(y_k^T s_k)$ and $b_k = 0$.
Step 8.	Compute the search direction as: $d_{k+1} = -g_{k+1} + a_k s_k + b_k y_k$.
Step 9.	Powell 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. ■

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 any 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 proves to be one of the best selection of the starting guess in line search.

Convergence analysis

Assume that:

(i) *The level set* $S = \{x \in R^n : f(x) \leq f(x_0)\}$ *is bounded, i.e. there exists positive constant* $B > 0$ *such that for all* $x \in S$, $\|x\| \leq B$.

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

Under these assumptions on f there exists a constant $\Gamma \geq 0$ such that $\|\nabla f(x)\| \leq \Gamma$ for all $x \in S$. Observe that the assumption that the function f is bounded below is weaker than the usual assumption that the level set is bounded.

Although the search directions generated by (2.2), (2.11) and (2.12) are always descent directions, to ensure convergence of the algorithm we need to constrain the choice of the step-length α_k. The following proposition shows that the Wolfe line search always gives a lower bound for the step-length α_k.

Proposition 4.1. *Suppose that d_k is a descent direction and the gradient ∇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 Wolfe conditions (1.4) and (1.5), then

$$\alpha_k \geq \frac{(1-\sigma)\left|g_k^T d_k\right|}{L\|d_k\|^2}. \tag{4.1}$$

Proof. Subtracting $g_k^T d_k$ from both sides of (1.5) and using the Lipschitz continuity we get

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

Since d_k is a descent direction and $\sigma < 1$, (4.1) follows immediately. ■

The following proposition proves that in the above three-term conjugate gradient method, under the general Wolfe line search (1.4) and (1.5), the Zoutendijk [41] condition holds.

Proposition 4.2. *Suppose that the assumptions (i) and (ii) hold. Consider the algorithm (1.2) and (2.2) with (2.11) and (2.12) where d_k is a descent direction and α_k is computed by the general Wolfe line search (1.4) and (1.5). Then*

$$\sum_{k=0}^{\infty} \frac{(g_k^T d_k)^2}{\|d_k\|^2} < +\infty. \tag{4.2}$$

Proof. From (1.4) and proposition 4.1 we get

$$f_k - f_{k+1} \geq -\rho\alpha_k g_k^T d_k \geq \rho \frac{(1-\sigma)(g_k^T d_k)^2}{L\|d_k\|^2}.$$

Therefore, from assumption (i) we get the Zoutendijk condition (4.2). ■

In [32] Powell proved that in conjugate gradient algorithms the iteration can fail, in the sense that $\|g_k\| \geq \gamma > 0$ for all k, only if $\|d_k\| \to \infty$ sufficiently rapidly.

More exactly, the sequence of gradient norms $\|g_k\|$ can be bounded away from zero only if $\sum_{k\geq 0} 1/\|d_k\| < \infty$. This observation is fundamental and can be used for global convergence analysis of nonlinear conjugate gradient algorithms. For any conjugate gradient method with strong Wolfe line search (1.4) and (1.6) the following general result holds [28].

Proposition 4.3. *Suppose that the assumptions (i) and (ii) hold and consider any conjugate gradient algorithm (1.2) where* d_k *is a descent direction and* α_k *is obtained by the strong Wolfe line search (1.4) and (1.6). If*

$$\sum_{k\geq 1} \frac{1}{\|d_k\|^2} = \infty, \tag{4.3}$$

then

$$\liminf_{k\to\infty} \|g_k\| = 0. \tag{4.4}$$

For *uniformly convex functions* we can prove that the norm of the direction d_{k+1} generated by (2.2) and (2.11)-(2.12) is bounded above. Therefore by proposition 4.3 we can prove the following result.

Theorem 4.1. *Suppose that the assumptions (i) and (ii) hold and consider the algorithm (1.2) and (2.2) with (2.11) and (2.12) with* $\omega = 1$, *where* d_k *is a descent direction and* α_k *is computed by the Wolfe line search (1.4) and (1.5). Suppose that* ∇f *satisfies the Lipschitz condition and* f *is a uniformly convex function on* S, *i.e. there exists a constant* $\mu > 0$ *such that*

$$(\nabla f(x) - \nabla f(y))^T (x - y) \geq \mu \|x - y\|^2 \tag{4.5}$$

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

$$\lim_{k\to\infty} \|g_k\| = 0. \tag{4.6}$$

Proof. From Lipschitz continuity we have $\|y_k\| \leq L\|s_k\|$. On the other hand, from uniform convexity it follows that $y_k^T s_k \geq \mu \|s_k\|^2$. Now, using the Cauchy inequality, from Lipschitz continuity and uniform convexity we have:

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

$$\le \frac{\Gamma\|y_k\|^3}{\mu\|s_k\|^2} + \frac{\Gamma\|s_k\|^2\|y_k\|}{\|s_k\|^2} \le \frac{\Gamma L^3\|s_k\|}{\mu} + \Gamma L\|s_k\|$$

$$= \left(\frac{\Gamma L^3}{\mu} + \Gamma L\right)\|s_k\| = M_1\|s_k\|. \tag{4.7}$$

On the other hand for $\omega = 1$ we have

$$\left|y_k^T g_{k+1} - \omega s_k^T g_{k+1}\right| \le \left|y_k^T g_{k+1}\right| + \left|s_k^T g_{k+1}\right| \le (\Gamma L + \Gamma)\|s_k\| \equiv M_2\|s_k\|. \tag{4.8}$$

From uniform convexity and Cauchy inequality observe that $\mu\|s_k\|^2 \le y_k^T s_k \le \|y_k\|\|s_k\|$, i.e.

$$\mu\|s_k\| \le \|y_k\|. \tag{4.9}$$

From (2.11) using (4.7), (4.8) and (4.9) with $\omega = 1$ we get:

$$|a_k| \le \frac{1}{(y_k^T y_k)^2}\left[\frac{2(y_k^T y_k)^2}{\left|y_k^T s_k\right|}\left|y_k^T g_{k+1} - s_k^T g_{k+1}\right| + (y_k^T y_k)\left|\theta_k - y_k^T g_{k+1}\right|\right]$$

$$\le \frac{2}{\left|y_k^T s_k\right|}M_2\|s_k\| + \frac{1}{y_k^T y_k}M_1\|s_k\| \le \frac{2}{\mu\|s_k\|^2}M_2\|s_k\| + \frac{1}{\mu^2\|s_k\|^2}M_1\|s_k\|$$

$$= \left[\frac{2}{\mu}M_2 + \frac{1}{\mu^2}M_1\right]\frac{1}{\|s_k\|} \equiv M_3\frac{1}{\|s_k\|}. \tag{4.10}$$

Now, from (2.12) using (4.7), (4.8) and (4.9) with $\omega = 1$ we get:

$$|b_k| \le \frac{1}{\|y_k\|^4}\left[\left|y_k^T s_k\right|\left|\theta_k - y_k^T g_{k+1}\right| + \|y_k\|^2\left|y_k^T g_{k+1} - s_k^T g_{k+1}\right|\right]$$

$$\le \frac{1}{\|y_k\|^4}\|y_k\|\|s_k\|M_1\|s_k\| + \frac{1}{\|y_k\|^2}M_2\|s_k\| \le \left[\frac{1}{\mu^2\|s_k\|^2}M_1\|s_k\|^2 + \frac{1}{\mu\|s_k\|}M_2\|s_k\|\right]\frac{1}{\|y_k\|}$$

$$= \left[\frac{M_1}{\mu^2} + \frac{M_2}{\mu}\right]\frac{1}{\|y_k\|} \equiv M_4\frac{1}{\|y_k\|}. \tag{4.11}$$

Therefore, from (2.2)

$$\|d_{k+1}\| \le \|g_{k+1}\| + |a_k|\|s_k\| + |b_k|\|y_k\| \le \Gamma + M_3 + M_4,$$

showing that (4.3) is true. From proposition 4.3 it follows that (4.4) is true, which for uniformly convex functions is equivalent to (4.6). ■

Convergence analysis for *general nonlinear functions* exploits the assumptions (i) and (ii), as well as the fact that by the Wolfe line search $y_k^T s_k > 0$ (strictly) and therefore it can be bounded from below by a positive constant, i.e. there exists $\tau > 0$ such that $y_k^T s_k \ge \tau$.

Proposition 4.4. *If the search direction d_k is a descent one, then by the Wolfe line search condition (1.5) there exists a constant $\tau > 0$ such that $y_k^T s_k \ge \tau$.*

Proof. Since d_k is a descent direction then $\nabla f(x_k)^T d_k \le 0$. Consider that α_k is chosen to satisfy the second Wolfe line search condition (1.5). Then

$$\nabla f(x_k + \alpha_k d_k)^T d_k \ge \sigma \nabla f(x_k)^T d_k > \nabla f(x_k)^T d_k, \tag{4.12}$$

since

$$0 < \sigma < 1 \text{ and } \nabla f(x_k)^T d_k \le 0.$$

Accordingly,

$$\nabla f(x_k + \alpha_k d_k)^T d_k - \nabla f(x_k)^T d_k \ge (\sigma - 1)\nabla f(x_k)^T d_k. \tag{4.13}$$

Therefore, from (4.13) we get:

$$(\nabla f(x_k + \alpha_k d_k) - \nabla f(x_k))^T d_k > 0 \tag{4.14}$$

i.e. $y_k^T d_k > 0$.

Now, if in the above inequality (4.13) we let α_k approach zero, than the left-hand side approaches zero, while the right-hand side remains constant at the value $(\sigma - 1)\nabla f(x_k)^T d_k > 0$, which is impossible. Thus, the Wolfe line search prevents arbitrarily small choices of α_k. Therefore, from (4.14), since $s_k = x_{k+1} - x_k = \alpha_k d_k$, it follows that $y_k^T s_k > 0$ (strictly). Besides, by the Archimedean property of the real numbers, there always exists a positive constant τ, arbitrarily small, such that $y_k^T s_k \ge \tau$. ■

Theorem 4.2. *Suppose that the assumptions (i) and (ii) hold and consider the algorithm (1.2) and (2.2) with (2.11) and (2.12) with* $\omega = 1$, *where* d_k *is a descent direction,* α_k *is computed by the Wolfe line search (1.4)-(1.5) and there exists a constant* $\tau > 0$ *such that* $y_k^T s_k \geq \tau$ *for any* $k \geq 1$. *Then*

$$\liminf_{k\to\infty} \|g_k\| = 0. \tag{4.15}$$

Proof. Since $g_k^T s_k < 0$ for any k, it follows that $s_k^T g_{k+1} = y_k^T s_k + g_k^T s_k < y_k^T s_k$. By the assumptions (i) and (ii) it follows that

$$\|y_k\| = \|g_{k+1} - g_k\| = \|\nabla f(x_k + \alpha_k d_k) - \nabla f(x_k)\| \leq L\|s_k\| \leq 2BL. \tag{4.16}$$

Suppose that $g_k \neq 0$ for all $k \geq 1$, otherwise a stationary point is obtained. Now, using the standard assumptions (i) and (ii) we have

$$\begin{aligned} \left|\theta_k - y_k^T g_{k+1}\right| &\leq \frac{\left|y_k^T g_{k+1}\right| \left|y_k^T y_k\right|}{\left|y_k^T s_k\right|} + \frac{\left|s_k^T g_{k+1}\right| \left|y_k^T s_k\right|}{\left|s_k^T s_k\right|} \leq \frac{\|y_k\| \Gamma \|y_k\|^2}{\tau} + \frac{\left|y_k^T s_k\right| \left|y_k^T s_k\right|}{\|s_k\|^2} \\ &\leq \frac{(2BL)\Gamma \|y_k\|^2}{\tau} + \frac{\|y_k\|^2 \|s_k\|^2}{\|s_k\|^2} = \left(\frac{2BL\Gamma}{\tau} + 1\right)\|y_k\|^2 \\ &\equiv M_5 \|y_k\|^2. \end{aligned} \tag{4.17}$$

On the other hand,

$$\begin{aligned} \left|y_k^T g_{k+1} - s_k^T g_{k+1}\right| &\leq \left|y_k^T g_{k+1}\right| + \left|s_k^T g_{k+1}\right| \leq \left|y_k^T g_{k+1}\right| + \left|y_k^T s_k\right| \\ &\leq (\Gamma + 2B)\|y_k\|. \end{aligned} \tag{4.18}$$

From (2.11) using (4.17) and (4.18) we get:

$$\begin{aligned} |a_k| &\leq \frac{1}{(y_k^T y_k)^2}\left[\frac{2(y_k^T y_k)^2}{\left|y_k^T s_k\right|}\left|y_k^T g_{k+1} - s_k^T g_{k+1}\right| + (y_k^T y_k)\left|\theta_k - y_k^T g_{k+1}\right|\right] \\ &\leq \frac{2}{\left|y_k^T s_k\right|}(\Gamma + 2B)\|y_k\| + \frac{1}{\|y_k\|^2} M_5 \|y_k\|^2 \\ &\leq \frac{2}{\tau}(\Gamma + 2B)L2B + M_5 \equiv M_6. \end{aligned} \tag{4.19}$$

On the other hand, from (2.12) using again (4.17) and (4.18) we have:

$$|b_k| \le \frac{1}{\|y_k\|^4}\left[|y_k^T s_k|\,|\theta_k - y_k^T g_{k+1}| + \|y_k\|^2 |y_k^T g_{k+1} - s_k^T g_{k+1}|\right]$$

$$\le \frac{1}{\|y_k\|^4}|y_k^T s_k| M_5 \|y_k\|^2 + \frac{1}{\|y_k\|^2}(\Gamma + 2B)\|y_k\| \le \frac{1}{\|y_k\|^2}\|y_k\|\|s_k\| M_5 + \frac{1}{\|y_k\|}(\Gamma + 2B)$$

$$= (2BM_5 + \Gamma + 2B)\frac{1}{\|y_k\|} = M_7 \frac{1}{\|y_k\|} \tag{4.20}$$

Therefore, from (2.2)

$$\|d_{k+1}\| \le \|g_{k+1}\| + |a_k|\|s_k\| + |b_k|\|y_k\| \le \Gamma + 2BM_6 + M_7. \tag{4.21}$$

Now, from proposition 4.3 it follows that (4.15) is true. ■

Numerical experiments and discussions

In this section we report some numerical results obtained with an implementation of the TTSCAL 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 80 large-scale unconstrained optimization test functions in generalized or extended form we presented in [4]. For each test function we have taken ten numerical experiments with the number of variables increasing as $n = 1000, 2000, ..., 10000.$ The TTSCAL algorithm implements the Wolfe line search conditions with cubic interpolation, $\rho = 0.0001,$ $\sigma = 0.8$ and the same stopping criterion $\|g_k\|_\infty \le 10^{-6}$, where $\|.\|_\infty$ is the maximum absolute component of a vector. In all the algorithms we considered in this numerical study the maximum number of iterations is limited to 10000. All algorithms implement the Powell restart technology, i.e. when $|g_{k+1}^T g_k| > 0.2\|g_{k+1}\|^2$, then the search direction is set to the negative gradient.

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 $i = 1, \ldots, 800,$ respectively. We say that, in the particular problem $i,$ the performance of ALG1 was better than the performance of ALG2 if:

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

and the number of iterations (#iter), or the number of function-gradient evaluations (#fg), or the CPU time of ALG1 was less than the number of

iterations, or the number of function-gradient evaluations, or the CPU time corresponding to ALG2, respectively. In this numerical study we declare that an algorithm solved a particular problem if the final point obtained had the lowest functional value among the tested algorithms (up to 10^{-3} tolerance as it was specified in (5.1). This criterion is acceptable for users who are interested in minimizing functions and not in finding critical points.

In the first set of numerical experiments we compare TTSCAL versus ASCALCG [2, 3], CONMIN [35], AHYBRIDM [6], CG-DESCENT [18], THREECG [9] and TTCG [8].

ASCALCG, elaborated by Andrei [2, 3], is an accelerated scaled conjugate gradient algorithm using a double update scheme embedded in the restart philosophy of Beale-Powell. The basic idea of ASCALCG is to combine the scaled memoryless BFGS method and the preconditioning technique in the frame of conjugate gradient method. The preconditioner, which is also a scaled memoryless BFGS matrix is reset when the Beale-Powell restart criterion holds. The parameter scaling the gradient is selected as a spectral gradient $\theta_{k+1} = s_k^T s_k / y_k^T s_k$. The search direction is computed as a double quasi-Newton updating scheme as:

$$d_{k+1} = -v + \frac{(g_{k+1}^T s_k)w + (g_{k+1}^T w)s_k}{y_k^T s_k} - \left(1 + \frac{y_k^T w}{y_k^T s_k}\right)\frac{g_{k+1}^T s_k}{y_k^T s_k} s_k, \tag{5.2}$$

where $v = H_{r+1} g_{k+1}$ and $w = H_{r+1} y_k$ and H_{r+1} is the BFGS approximation to the inverse Hessian initialized with the identity matrix and scaled by the scalar θ_{r+1} at the $r-$th iteration where the Beale-Powell restart test is satisfied:

$$H_{r+1} = \theta_{r+1} I - \theta_{r+1}\frac{y_r s_r^T + s_r y_r^T}{y_r^T s_r} + \left(1 + \theta_{r+1}\frac{y_r^T y_r}{y_r^T s_r}\right)\frac{s_r s_r^T}{y_r^T s_r}. \tag{5.3}$$

The restart direction is computed as $d_{k+1} = -Q_{k+1}^* g_{k+1}$, where Q_{k+1}^* is exactly the BFGS quasi-Newton matrix, and at every step the approximation of the inverse Hessian is the identity matrix multiplied by the scalar θ_{k+1}, i.e.

$$d_{k+1} = -\theta_{k+1} g_{k+1} + \theta_{k+1}\left(\frac{g_{k+1}^T s_k}{y_k^T s_k}\right) y_k - \left[\left(1 + \theta_{k+1}\frac{y_k^T y_k}{y_k^T s_k}\right)\frac{g_{k+1}^T s_k}{y_k^T s_k} - \theta_{k+1}\frac{g_{k+1}^T y_k}{y_k^T s_k}\right] s_k. \tag{5.4}$$

For the step-length computation the algorithm implements the Wolfe line search conditions in the same manner as in CONMIN.

CONMIN, established by Shanno and Phua [35], (see also [33, 34]) is a conjugate gradient algorithm which may be interpreted as a memoryless BFGS quasi-Newton algorithm optimally scaled in the sense of Oren and Spedicato [29]. In CONMIN the scaling is combined with Beale-Powell's restart criterion. The direction d_{k+1} in CONMIN is computed as:

$$d_{k+1} = -H_{k+1}g_{k+1} + A_k y_k - B_k s_k, \tag{5.5}$$

where H_{k+1} is BFGS approximation of the inverse Hessian which at every iteration is initialized with identity matrix, and A_k and B_k are specific matrices. The main drawback of this method is that if H_{k+1} contains useful information about the inverse Hessian of the function f, then we are better off using the search direction $d_{k+1} = -H_{k+1}g_{k+1}$ since the addition of the last terms in (5.5) may prevent the direction d_{k+1} from being a descent direction unless the line search is sufficiently accurate.

AHYBRIDM, elaborated by Andrei [6], is an accelerated hybrid conjugate gradient algorithm in which the search direction is computed as a convex combination of Hestenes-Stiefel [19] and Dai-Yuan [13] conjugate gradient algorithms. The parameter in this convex combination is computed in such a way the direction corresponding to the conjugate gradient algorithm is the best direction we know, i.e. the Newton direction, while the pair (s_k, y_k) satisfies the modified secant equation $B_{k+1}s_k = z_k$, where

$$z_k = y_k + (\eta_k / \|s_k\|^2)s_k,\ \eta_k = 2(f_k - f_{k+1}) + (g_k + g_{k+1})^T s_k. \tag{5.6}$$

CG-DESCENT was elaborated by Hager and Zhang [18] in order to ensure sufficient descent, independent by the accuracy of the line search. In CG_DESCENT the search direction $d_{k+1} = -g_{k+1} + \beta_k^{HZ} s_k$, where

$$\beta_k^{HZ} = \left(y_k - 2\frac{\|y_k\|^2}{y_k^T s_k} s_k \right)^T \frac{g_{k+1}}{y_k^T s_k}, \tag{5.7}$$

satisfies the sufficient descent condition $g_k^T d_k \le -(7/8)\|g_k\|^2$. Mainly, CG-DESCENT is a modification of the HS algorithm in such a way when iterates jam the expression $(\|y_k\|^2 (s_k^T g_{k+1}))/(y_k^T s_k)^2$ in the formulation of β_k^{HZ} from (5.7)

becomes negligible. This modification of the HS scheme makes CG-DESCENT to perform better than HS [18].

THREECG, written by Andrei [9], is a simple three-term conjugate gradient algorithm which consists of a modification of the HS or of CG-DESCENT in such a way that the search direction is descent and it satisfies the conjugacy condition. These properties are independent by the line search. The direction d_{k+1} is computed as $d_{k+1} = -g_{k+1} - \delta_k s_k - \eta_k y_k$, where

$$\delta_k = \left(1 + \frac{\|y_k\|^2}{y_k^T s_k}\right)\frac{s_k^T g_{k+1}}{y_k^T s_k} - \frac{y_k^T g_{k+1}}{y_k^T s_k}, \quad \eta_k = \frac{s_k^T g_{k+1}}{y_k^T s_k}. \tag{5.8}$$

Also, the algorithm could be considered as a simple modification of the memoryless BFGS quasi-Newton method [9].

TTCG, also written by Andrei [8], is a three-term conjugate gradient algorithm, which is a modification of the Hestenes and Stiefel [19] or a modification of the CG_DESCENT by Hager and Zhang [18] algorithms, for which both the descent condition and the conjugacy condition are simultaneously satisfied.

The algorithm is given by (1.2) where the direction d_{k+1} is computed as $d_{k+1} = -g_{k+1} - \delta_k s_k - \eta_k y_k$, where

$$\delta_k = \left(1 + 2\frac{\|y_k\|^2}{y_k^T s_k}\right)\frac{s_k^T g_{k+1}}{y_k^T s_k} - \frac{y_k^T g_{k+1}}{y_k^T s_k}, \quad \eta_k = \frac{s_k^T g_{k+1}}{y_k^T s_k}. \tag{5.9}$$

Intensive numerical experiments showed that TTCG is clearly more efficient and slightly more robust than THREECG [8].

Figure 1 shows the Dolan and Moré [15] CPU performance profile of TTSCAL versus these conjugate gradient algorithms.

In a performance profile plot, the top curve corresponds to the method that solved the most problems in a time that was within a given factor of the best time.

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 TTSCAL with all these conjugate gradient algorithms subject to CPU time metric we see that TTSCAL is top performer.

The three-term accelerated conjugate gradient algorithm TTSCAL is more successful and more robust than all these conjugate gradient algorithms. For example, comparing TTSCAL versus CG-DESCENT (see Figure 1), subject to the number of iterations, we see that TTSCAL was better in 645 problems (i.e. it achieved the minimum number of iterations in 645 problems).

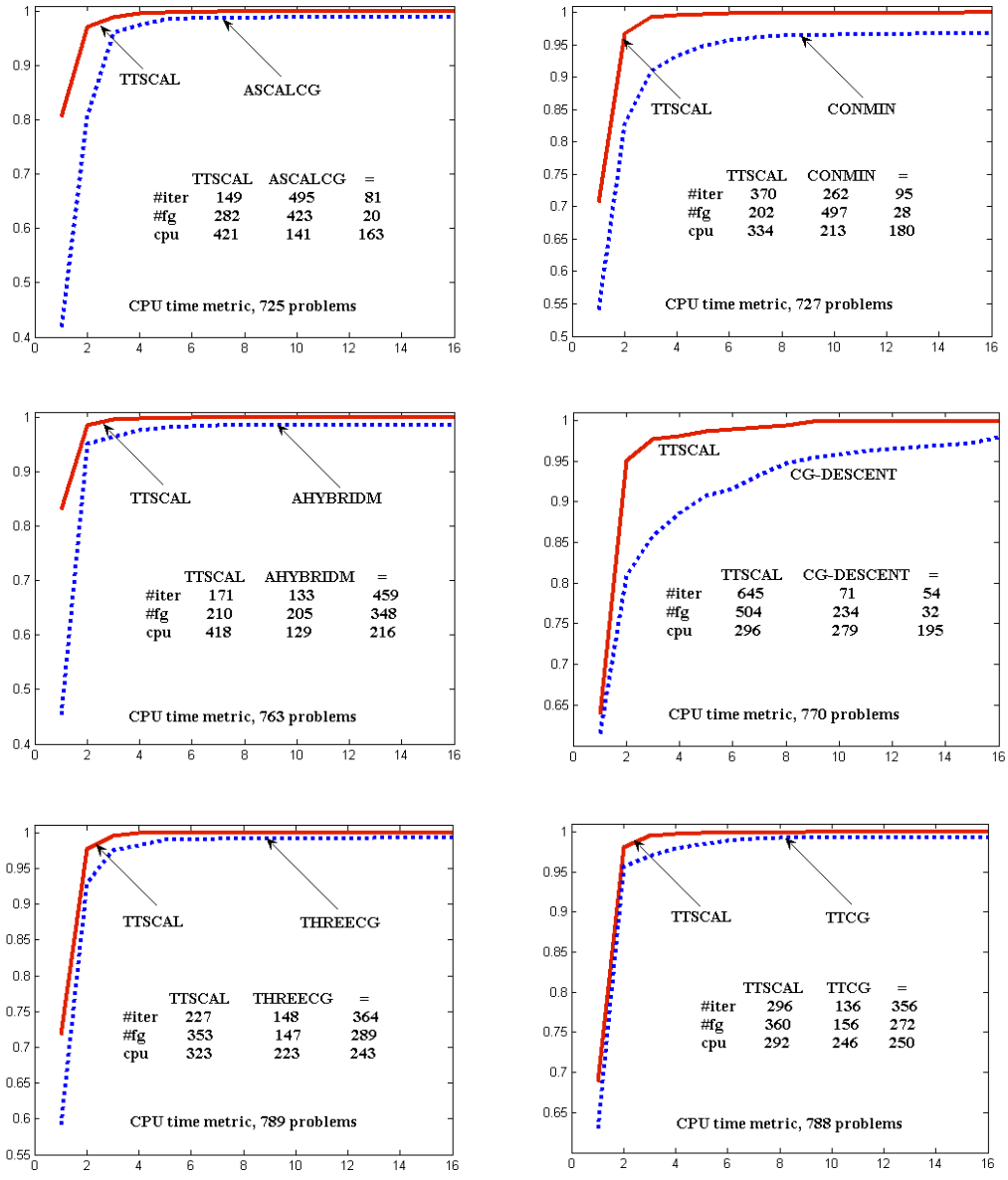

Fig. 1. TTSCAL versus ASCALCG, CONMIN, AHYBRIDM, CG-DESCENT, THREECH and TTCG, subject to CPU time metric.

CG-DESCENT was better in 71 problems and they achieved the same number of iterations in 54 problems, etc.

Out of 800 problems, only for 770 problems does the criterion (5.1) hold. Therefore in comparison with CG-DESCENT, TTSCAL appears to generate the best search direction and the best step-length, on average. It is known that, besides the conjugate gradient methods, for large-scale unconstrained optimization, two other methods can be successfully tried: the *limited memory BFGS method* (L-BFGS) and the *discrete truncated-Newton method* (TN). Both these methods use a low and predictable amount of storage, requiring only the function and its gradient values at each iterate. Both methods have been intensive tested on large problems of different types and their performance appears to be satisfactory [25].

Therefore, in the second set of numerical experiments we compare TTSCAL versus LBFG (m = 5) [27] and TN [24].

L-BFGS is an adaptation of the BFGS method for solving large-scale problems. In BFGS method the search direction is computed as $d_{k+1} = -H_{k+1}g_{k+1}$, where H_{k+1} is an approximation to the inverse Hessian matrix of f, updated as

$$H_{k+1} = V_k^T H_k V_k + \frac{s_k s_k^T}{y_k^T s_k}, \tag{5.10}$$

and $V_k = I - (y_k s_k^T)/(y_k^T s_k)$.

In L-BFGS method, instead of forming the matrices H_k, a number of m vectors s_k and y_k that define them implicitly are saved, as is described in [27]. The numerical experience with L-BFGS method, reported in [17], indicates that values of m in the range $3 \le m \le 7$ give the best results. Therefore, in this paper we consider the value $m = 5$. The line-search is performed by means of the routine CVSRCH by Moré and Thuente [22] which uses cubic interpolation.

The TN method is described by Nash [24]. At each *outer* iteration of the TN method, for determination of the search direction d_k an approximate solution of the Newton system $\nabla^2 f(x_k)d_k = -g_k$ is found using a number of *inner* iterations based on a preconditioned linear conjugate gradient method. The matrix-vector products required by the inner conjugate gradient algorithm are computed by finite differencing, i.e. the Hessian matrix is not explicitly computed.

Besides, the conjugate gradient inner iteration is preconditioned by a scaled two-step limited memory BFGS method with Powell's restarting strategy used to reset the preconditioner periodically. In TN the line search is performed using the strong Wolfe conditions.

Figure 2 presents the Dolan and Moré CPU performance profiles of TTSCAL versus L-BFGS (m = 5) and TN, respectively.

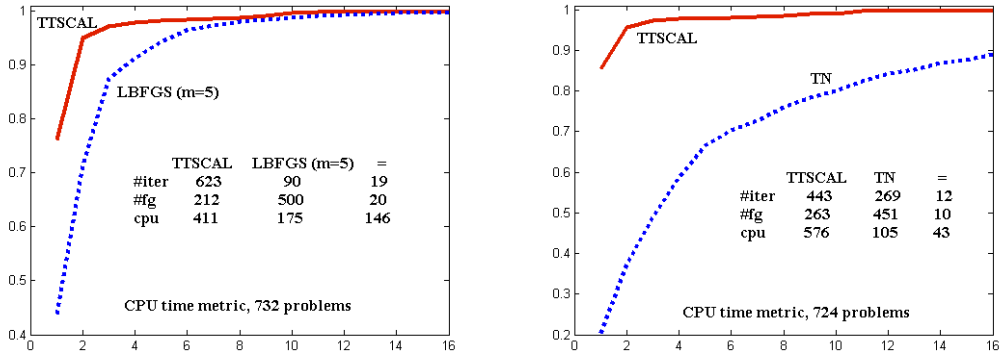

Fig. 2. TTSCAL versus LBFGS (m=5) and TN, subject to CPU time metric.

Observe that TTSCAL is more efficient and more robust versus both LBFGS (m=5) and TN algorithms. These algorithms are different in many respects. The principles on which these algorithms are based are very different. The linear algebra in LBFGS and TN codes to update the search direction is more time consuming than the linear algebra in TTSCAL. This is the main reason why TTSCAL is more efficient and more robust in this numerical study.

2. Conclusions

Three-term conjugate gradient algorithms represent one of the most important developments in large scale unconstrained optimization. In this paper the search direction is selected as a linear combination of $-g_{k+1}$, s_k and y_k, where the coefficients in this combination are selected to minimize the quadratic model of the minimizing function in which the symmetrical approximation of the Hessian matrix satisfies the general quasi-Newton equation. The parameter in general quasi-Newton equation is determined by the formal equality between the search direction used in the algorithm and the Newton direction. The algebraic developments prove that the best value of this parameter is equal to 1. This mechanism for the search direction computation proved to be very effective both subject to the efficiency and to the robustness of the algorithm. Numerical experiments using a large collection of 800 large-scale unconstrained optimization test problems showed that the suggested three-term conjugate gradient algorithm is both more efficient and more robust than some known conjugate gradient algorithms as well as than the limited memory quasi-Newton LBFGS and the discrete truncated-Newton methods.

R E F E R E N C E S

[1] Al-Bayati, A.Y., Sharif, W.H., *A new three-term conjugate gradient method for unconstrained optimization*. Canadian Journal on Science and Engineering Mathematics, vol. **1**, No. 5, October **2010**, pp. 108-124.

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

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

[4] Andrei, N., *Another collection of large-scale unconstrained optimization test functions*. ICI Technical Report, January 30, 2013.

[5] Andrei, N., *Acceleration of conjugate gradient algorithms for unconstrained optimization*. Applied Mathematics and Computation, **213** (**2009**), 361-369.

[6] Andrei, N., *Accelerated hybrid conjugate gradient algorithm with modified secant condition for unconstrained optimization*. Numerical Algorithms, **54** (**2010**), pp. 23-46.

[7] Andrei, N., *A modified Polak-Ribière-Polyak conjugate gradient algorithm for unconstrained optimization*. Optimization, **60** (**2011**) pp. 1457-1471.

[8] Andrei, N., *On three-term conjugate gradient algorithms for unconstrained optimization.* Applied Mathematics and Computation, vol. **210**, Issue 11, 1 February **2013**, pp. 6316-6327.

[9] Andrei, N., *A simple three-term conjugate gradient algorithm for unconstrained optimization.* Journal of Computational and Applied Mathematics. vol. **241** (**2013**), pp. 19-29.

[10] Beale, E.M.L., *A derivative of conjugate gradients*. In Lootsma, F.A., (Ed.) Numerical Methods for Nonlinear Optimization, Academic Press, London, 1972, pp.39-43.

[11] Cheng, W., *A two-term PRP-based descent method*. Numerical Functional Analysis and Optimization, **28** (2007), 1217-1230.

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

[13] Dai, Y.H., Yuan, Y., *A nonlinear conjugate gradient method with a strong global convergence property*, SIAM J. Optim., **10** (**1999**), pp. 177-182.

[14] Deng, N.Y., Li, Z., *Global convergence of three terms conjugate gradient methods*. Optim. Method and Software, **4** (**1995**), 273-282.

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

[16] Fletcher, R. and Reeves, C.M., *Function minimization by conjugate gradients* Comput. J. 7, 149-154 (**1964**).

[17] Gilbert, J.C., Lemaréchal, C., *Some numerical experiments with variable storage quasi-Newton algorithm*. Math. Programming, **45** (**1989**) pp. 407-435.

[18] Hager, W.W. and Zhang, H., *A new conjugate gradient method with guaranteed descent and an efficient line search*, SIAM Journal on Optimization, Vol. **16**, pp. 170-192, **2005**.

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

[20] Liu, Y., Storey, C., *Efficient generalized conjugate gradient algorithms, Part 1: Theory*. JOTA, **69** (**1991**), pp. 129-137.

[21] McGuire, M.F., Wolfe, P., *Evaluating a restart procedure for conjugate gradients*. Report RC-4382, IBM Research Center, Yorktown Heights, **1973**.

[22] Moré, J.J., Thuente, D.J., *On linesearch algorithms with guaranteed sufficient decrease*. Mathematics and Computer Science Division Preprint MCS-P153-0590, Argonne National Laboratory, Argonne, IL, **1990**.

[23] Narushima, Y., Yabe, H., Ford, J.A., *A three-term conjugate gradient method with sufficient descent property for unconstrained optimization*. SIAM J. on Optimization, vol. **21**, No. 1, (**2011**), 212-230.

[24] Nash, S.G., *User's guide for TN-TNBC: Fortran routines for nonlinear optimization*. Report 397, Mathematical Sciences Department, The John Hopkins University, Baltimore, MD.

[25] Nash, S.G., Nocedal, J., *A numerical study of the limited memory BFGS method and the truncated-Newton method for large scale optimization*. SIAM J. Optimization, **1** (**1991**), pp. 358-372.

[26] Nazareth, L., *A conjugate direction algorithm without line search*. J. Optim. Theort Appl., **23** (**1977**), pp. 373-387.

[27] Nocedal, J., *Updating quasi-Newton matrices with limited storage*. Math. Comp., **35** (**1980**), pp. 773-782.

[28] Nocedal, J., *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.

[29] Oren, S.S., Spedicato, E., *Optimal conditioning of self-scaling variable metric algorithms*. Math. Programming, **10** (**1976**) 70-90.

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

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

[32] Powell, M.J.D., *Nonconvex minimization calculations and the conjugate gradient method*. Numerical Analysis (Dundee, 1983), Lecture Notes in Mathematics, Vol. 1066, Springer, Berlin, **1984**, pp. 122-141.

[33] Shanno, D.F., *On the convergence of a new conjugate gradient algorithm*. SIAM J. Numer. Anal., **15** (1978), pp. 1247-1257.

[34] Shanno, D.F. *Conjugate gradient methods with inexact searches*. Mathematics of Operations Research, vol.**3**, No.3, (**1978**), 244-256.

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

[36] Wolfe, P., *Convergence conditions for ascent methods*, SIAM Rev., Vol. **11**, pp. 226-235, **1968**.

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

[38] Zhang, L., Zhou, W., Li, D.H., *A descent modified Polak-Ribière-Polyak conjugate gradient method and its global convergence*. IMA J. Numer. Anal., **26** (**2006**), pp. 629-640.

[39] Zhang, L., Zhou, W., Li, D.H., *Some descent three-term conjugate gradient methods and their global convergence*. Optimization Methods and Software, **22** (**2007**), pp. 697-711.

[40] Zhang, J., Xiao, Y., Wei, Z., *Nonlinear conjugate gradient methods with sufficient descent condition for large-scale unconstrained optimization*. Mathematical Problems in Engineering, vol. 2009 (**2009**), Article ID 243290. DOI: 10.1155/2009/243290.

[41] Zoutendijk, G., *Nonlinear programming, computational methods.* In J. Abadie (Ed.) *Integer and Nonlinear Programming*. North-Holland, Amsterdam, **1970**, pp. 38-86.