Scaled memoryless BFGS preconditioned conjugate gradient algorithm for unconstrained optimization

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

Abstract. A scaled memoryless BFGS preconditioned conjugate gradient algorithm for solving unconstrained optimization problems is presented. The basic idea is to combine the scaled memoryless BFGS method and the preconditioning technique in the frame of the 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 spectral gradient. In very mild conditions it is shown that, for strongly convex functions, the algorithm is globally convergent. Computational results for a set consisting of 750 unconstrained optimization test problems, show that this new scaled conjugate gradient algorithm substantially outperforms known conjugate gradient methods including: the spectral conjugate gradient by Birgin and Martínez [2], the conjugate gradient by Polak and Ribière [13], as well as the most recent CG_DESCENT conjugate gradient method with guaranteed descent by Hager and Zhang [7,8].

Keywords: Unconstrained optimization; conjugate gradient method; spectral gradient method; Wolfe line search; BFGS preconditioning
2000 Mathematics Subject Classification: 49M07; 49M10; 90C06; 65K

1. Introduction

In this paper we consider the following unconstrained optimization problem:

$$min\ f(x) \tag{1}$$

where $f:R^n \to R$ is continuously differentiable and its gradient is available. We are interested in elaborating an algorithm for solving large-scale cases for which the Hessian of f is either not available or requires a large amount of storage and computational costs.
The paper presents a conjugate gradient algorithm based on a combination of the scaled memoryless BFGS method and the preconditioning technique. For general nonlinear functions a good preconditioner is any matrix that approximates $\nabla^2 f(x^*)^{-1}$, where x^* is the solution of (1). In this algorithm the preconditioner is a scaled memoryless BFGS matrix which is reset when the Powell restart criterion holds. The scaling factor in the preconditioner is selected as spectral gradient.

The algorithm uses the conjugate gradient direction where the famous parameter β_k is obtained by equating the conjugate gradient direction with the direction corresponding to the Newton method. Thus, we get a general formula for the direction computation, which could be particularized to include the Polak and Ribiére [13] and the Fletcher and Reeves [6] conjugate gradient algorithms, the spectral conjugate gradient (SCG) by Birgin and Martínez

[1] The author was awarded the Romanian Academy Grant 168/2003.

[2] or the algorithm of Dai and Liao [4], for $t = 1$. This direction is then modified in a canonical manner as it was considered earlier by Oren and Luenberger [10], Oren and Spedicato [11], Perry [12] and Shanno [17, 18], by means of a scaled, memoryless BFGS preconditioner placed into the Beale-Powell restart technology. The scaling factor is computed in a spectral manner based on the inverse Rayleigh quotient, as suggested by Raydan [16]. The method is an extension of the spectral conjugate gradient (SCG) by Birgin and Martínez [2] or of a variant of the conjugate gradient algorithm by Dai and Liao [4] (for $t = 1$) to overcome the lack of positive definiteness of the matrix defining their search direction.

The paper is organized as follows: In section 2 we present the method. Section 3 is dedicated to the SCALCG algorithm. The algorithm performs two types of steps: a standard one in which a double quasi-Newton updating scheme is used and a restart one where the current information is used to define the search direction. The convergence of the algorithm for strongly convex functions is proved in section 4. Finally, in section 5 we present computational results on a set of 750 unconstrained optimization problems from the CUTE [3] collection along with some other large-scale unconstrained optimization problems, and compare the performance of the new algorithm to SCG conjugate gradient method by Birgin and Martínez [2], the Polak and Ribiére conjugate gradient algorithm [13], as well as the recent CG_DESCENT by Hager and Zhang [7, 8].

2. The Method

The algorithm generates a sequence x_k of approximations to the minimum x^* of f, in which

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

$$d_{k+1} = -\theta_{k+1} g_{k+1} + \beta_k s_k, \tag{3}$$

where $g_k = \nabla f(x_k)$, α_k is selected to minimize $f(x)$ along the search direction d_k, β_k is a scalar parameter, $s_k = x_{k+1} - x_k$ and θ_{k+1} is a parameter to be determined. The iterative process is initialized with an initial point x_0 and $d_0 = -g_0$.

Observe that if $\theta_{k+1} = 1$, then we get the classical conjugate gradient algorithms according to the value of the scalar parameter β_k. On the other hand, if $\beta_k = 0$, then we get another class of algorithms according to the selection of the parameter θ_{k+1}. Considering $\beta_k = 0$, there are two possibilities for θ_{k+1}: a positive scalar or a positive definite matrix. If $\theta_{k+1} = 1$, then we have the steepest descent algorithm. If $\theta_{k+1} = \nabla^2 f(x_{k+1})^{-1}$, or an approximation of it, then we get the Newton or the quasi-Newton algorithms, respectively. Therefore, we see that in the general case, when $\theta_{k+1} \neq 0$ is selected in a quasi-Newton manner, and $\beta_k \neq 0$, (3) represents a combination between the quasi-Newton and the conjugate gradient methods. However, if θ_{k+1} is a matrix containing some useful information about the inverse Hessian of function f, we are better off using $d_{k+1} = -\theta_{k+1} g_{k+1}$ since the addition of the term $\beta_k s_k$ in (3) may prevent the direction d_k from being a descent direction unless the line search is sufficiently accurate. Therefore, in this paper we shall consider θ_{k+1} as a positive scalar which contains some useful information to the inverse Hessian of function f.

To determine β_k consider the following procedure. As we know, the Newton direction for solving (1) is given by $d_{k+1} = -\nabla^2 f(x_{k+1})^{-1} g_{k+1}$. Therefore, from the equality

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

we get:

$$\beta_k = \frac{s_k^T \nabla^2 f(x_{k+1}) \theta_{k+1} g_{k+1} - s_k^T g_{k+1}}{s_k^T \nabla^2 f(x_{k+1}) s_k}. \tag{4}$$

Using the Taylor development, after some algebra we obtain:

$$\beta_k = \frac{(\theta_{k+1} y_k - s_k)^T g_{k+1}}{y_k^T s_k}, \tag{5}$$

where $y_k = g_{k+1} - g_k$. Birgin and Martínez [2] arrived at the same formula for β_k, but using a geometric interpretation of quadratic function minimization. The direction corresponding to β_k given in (5) is as follows:

$$d_{k+1} = -\theta_{k+1} g_{k+1} + \frac{(\theta_{k+1} y_k - s_k)^T g_{k+1}}{y_k^T s_k} s_k. \tag{6}$$

This direction is used by Birgin and Martínez [2] in their SCG (spectral conjugate gradient) package for unconstrained optimization, where θ_{k+1} is selected in a spectral manner, as suggested by Raydan [16]. The following particularizations are obvious. If $\theta_{k+1} = 1,$ then (6) is the direction considered by Perry [12]. At the same time we see that (6) is the direction given by Dai and Liao [4] for $t = 1,$ obtained this time by an interpretation of the conjugacy condition. Additionally, if $s_j^T g_{j+1} = 0, \quad j = 0,1,\ldots,k,$ then from (6) we get:

$$d_{k+1} = -\theta_{k+1} g_{k+1} + \frac{\theta_{k+1} y_k^T g_{k+1}}{\alpha_k \theta_k g_k^T g_k} s_k, \tag{7}$$

which is the direction corresponding to a *generalization of the Polak and Ribière* formula. Of course, if $\theta_{k+1} = \theta_k = 1$ in (7), we get the *classical Polak and Ribière* formula [13]. If $s_j^T g_{j+1} = 0, \quad j = 0,1,\ldots,k,$ and additionally the successive gradients are orthogonal, then from (6)

$$d_{k+1} = -\theta_{k+1} g_{k+1} + \frac{\theta_{k+1} g_{k+1}^T g_{k+1}}{\alpha_k \theta_k g_k^T g_k} s_k, \tag{8}$$

which is the direction corresponding to a *generalization of the Fletcher and Reeves* formula [6]. Therefore, (6) is a general formula for direction computation in a conjugate gradient manner including the classical Fletcher and Reeves [6], and Polak and Ribière [13] formulas.

There is a result by Shanno [17, 18] that says that the conjugate gradient method is precisely the BFGS quasi-Newton method for which the initial approximation to the inverse of the Hessian, at every step, is taken as the identity matrix. The extension to the scaled conjugate gradient is very simple. Using the same methodology as considered by Shanno [17] we get the following direction d_{k+1} :

$$d_{k+1} = -\theta_{k+1} g_{k+1} + \theta_{k+1} \left(\frac{g_{k+1}^T s_k}{y_k^T s_k} \right) y_k - \left[\left(1 + \theta_{k+1} \frac{y_k^T y_k}{y_k^T s_k} \right) \frac{g_{k+1}^T s_k}{y_k^T s_k} - \theta_{k+1} \frac{g_{k+1}^T y_k}{y_k^T s_k} \right] s_k, \tag{9}$$

involving only 4 scalar products. Again observe that if $g_{k+1}^T s_k = 0,$ then (9) reduces to:

$$d_{k+1} = -\theta_{k+1} g_{k+1} + \theta_{k+1} \frac{g_{k+1}^T y_k}{y_k^T s_k} s_k. \tag{10}$$

Thus, in this case, the effect is simply one of multiplying the Hestenes and Stiefel [9] search direction by a positive scalar.

In order to ensure the convergence of the algorithm (2), with d_{k+1} given by (9), we need to constrain the choice of α_k. We consider line searches that satisfy the Wolfe conditions [20, 21]:

$$f(x_k + \alpha_k d_k) - f(x_k) \le \sigma_1 \alpha_k g_k^T d_k, \tag{11}$$

$$\nabla f(x_k + \alpha_k d_k)^T d_k \ge \sigma_2 g_k^T d_k, \tag{12}$$

where $0 < \sigma_1 \le \sigma_2 < 1.$

Theorem 1. *Suppose that α_k in (2) satisfies the Wolfe conditions (11) and (12), then the direction d_{k+1} given by (9) is a descent direction.*

Proof: Since $d_0 = -g_0,$ we have $g_0^T d_0 = -\|g_0\|^2 \le 0.$ Multiplying (9) by $g_{k+1}^T,$ we have

$$g_{k+1}^T d_{k+1} = \frac{1}{(y_k^T s_k)^2}\Big[-\theta_{k+1}\|g_{k+1}\|^2 (y_k^T s_k)^2 + 2\theta_{k+1}(g_{k+1}^T y_k)(g_{k+1}^T s_k)(y_k^T s_k) - (g_{k+1}^T s_k)^2 (y_k^T s_k) - \theta_{k+1}(y_k^T y_k)(g_{k+1}^T s_k)^2\Big].$$

Applying the inequality $u^T v \le \frac{1}{2}(\|u\|^2 + \|v\|^2)$ to the second term of the right hand side of the above equality, with $u = (s_k^T y_k) g_{k+1}$ and $v = (g_{k+1}^T s_k) y_k$ we get:

$$g_{k+1}^T d_{k+1} \le -\frac{(g_{k+1}^T s_k)^2}{y_k^T s_k}. \tag{13}$$

But, by Wolfe condition (12), $y_k^T s_k > 0.$ Therefore, $g_{k+1}^T d_{k+1} < 0$ for every $k = 0,1,\ldots$ ■

Observe that the second Wolfe condition (12) is crucial for the descent character of direction (9). Besides, we see that the estimation (13) is independent of the parameter θ_{k+1}.

Usually, all conjugate gradient algorithms are periodically restarted. The Powell restarting procedure [14, 15] is to test if there is very little orthogonality left between the current gradient and the previous one. At step r when:

$$\left|g_{r+1}^T g_r\right| \ge 0.2\|g_{r+1}\|^2, \tag{14}$$

we restart the algorithm using the direction given by (9).

At step r we know s_r, y_r and θ_{r+1}. If (14) is satisfied, then a restart step is considered, i.e. the direction is computed as in (9). For $k \ge r+1,$ we consider the same philosophy used by Shanno [17, 18], where the gradient g_{k+1} is modified by a positive definite matrix which best estimates the inverse Hessian without any additional storage requirements, i.e. we compute:

$$v = \theta_{r+1} g_{k+1} - \theta_{r+1}\left(\frac{g_{k+1}^T s_r}{y_r^T s_r}\right) y_r + \left[\left(1 + \theta_{r+1}\frac{y_r^T y_r}{y_r^T s_r}\right)\frac{g_{k+1}^T s_r}{y_r^T s_r} - \theta_{r+1}\frac{g_{k+1}^T y_r}{y_r^T s_r}\right] s_r, \tag{15}$$

and

$$w = \theta_{r+1} y_k - \theta_{r+1}\left(\frac{y_k^T s_r}{y_r^T s_r}\right) y_r + \left[\left(1 + \theta_{r+1}\frac{y_r^T y_r}{y_r^T s_r}\right)\frac{y_k^T s_r}{y_r^T s_r} - \theta_{r+1}\frac{y_k^T y_r}{y_r^T s_r}\right] s_r, \tag{16}$$

involving 6 scalar products. With these, at any nonrestart step, the direction d_{k+1} for $k \ge r+1,$ is computed using a double update scheme as in Shanno [17]:

$$d_{k+1} = -v + \frac{(g_{k+1}^T s_k) w + (g_{k+1}^T w) s_k}{y_k^T s_k} - \left(1 + \frac{y_k^T w}{y_k^T s_k}\right)\frac{g_{k+1}^T s_k}{y_k^T s_k} s_k, \tag{17}$$

involving only 4 scalar products. Observe that $y_k^T s_k > 0$ is sufficient to ensure that the direction d_{k+1} given by (17) is well defined and it is always a descent direction.

Motivated by the efficiency of the spectral gradient method introduced by Raydan [16] and used by Birgin and Martínez [2] in their spectral conjugate gradient method for unconstrained optimization, in our algorithm θ_{k+1} is defined as a scalar approximation to the inverse Hessian. This is given as the inverse of the Rayleigh quotient:

$$s_k^T \left[\int_0^1 \nabla^2 f(x_k + t s_k) dt \right] s_k \; / \; s_k^T s_k ,$$

i.e.

$$\theta_{k+1} = \frac{s_k^T s_k}{y_k^T s_k}. \tag{18}$$

The inverse of Rayleigh quotient lies between the smallest and the largest eigenvalue of the Hessian average $\int_0^1 \nabla^2 f(x_k + t s_k) dt$. Again observe $y_k^T s_k > 0$ is sufficient to ensure that θ_{k+1} in (18) is well defined.

3. SCALCG Algorithm

Having in view the above developments and the definitions of g_k, s_k and y_k, as well as the selection procedure for θ_{k+1} computation, the following scaled conjugate gradient algorithm can be presented.

Step 1. Initialization. Select $x_0 \in R^n$, and the parameters $0 < \sigma_1 \le \sigma_2 < 1$. Compute $f(x_0)$ and $g_0 = \nabla f(x_0)$. Set $d_0 = -g_0$ and $\alpha_0 = 1 / \|g_0\|$. Set $k = 0$.
Step 2. Line search. Compute α_k satisfying the Wolfe conditions (11) and (12). Update the variables $x_{k+1} = x_k + \alpha_k d_k$. Compute $f(x_{k+1}), g_{k+1}$ and $s_k = x_{k+1} - x_k$, $y_k = g_{k+1} - g_k$.
Step 3. Test for continuation of iterations. If this test is satisfied the iterations are stopped, else set $k = k + 1$.
Step 4. Scaling factor computation. Compute θ_k using (18).
Step 5. Restart direction. Compute the (restart) direction d_k as in (9).
Step 6. Line search. Compute the initial guess: $\alpha_k = \alpha_{k-1} \|d_{k-1}\|_2 / \|d_k\|_2$. Using this initialization compute α_k satisfying the Wolfe conditions. Update the variables $x_{k+1} = x_k + \alpha_k d_k$. Compute $f(x_{k+1})$, g_{k+1} and $s_k = x_{k+1} - x_k$, $y_k = g_{k+1} - g_k$.
Step 7. Store: $\theta = \theta_k$, $s = s_k$ and $y = y_k$.
Step 8. Test for continuation of iterations. If this test is satisfied the iterations are stopped, else set $k = k + 1$.
Step 9. Restart. If the Powell restart criterion (14) is satisfied, then go to step 4 (a restart step); otherwise continue with step 10 (a standard step).
Step 10. Standard direction. Compute the direction d_k as in (17), where v and w are computed as in (15) and (16) with saved values θ, s and y.
Step 11. Line search. Compute the initial guess: $\alpha_k = \alpha_{k-1} \|d_{k-1}\|_2 / \|d_k\|_2$. Using this initialization compute α_k satisfying the Wolfe conditions. Update the variables $x_{k+1} = x_k + \alpha_k d_k$. Compute $f(x_{k+1})$, g_{k+1} and $s_k = x_{k+1} - x_k$, $y_k = g_{k+1} - g_k$.
Step 12. Test for continuation of iterations. If this test is satisfied the iterations are stopped, else set $k = k + 1$ and go to step 9. ■

It is well known that if f is bounded below along the direction d_k, then there exists a step length α_k satisfying the Wolfe conditions. The initial selection of the step length crucially affects the practical behaviour of the algorithm. At every iteration $k \geq 1$ the starting guess for the step α_k in line search is computed as $\alpha_{k-1}\|d_{k-1}\|_2 / \|d_k\|_2$. This procedure was considered for the first time by Shanno and Phua in CONMIN [19]. The same one is taken by Birgin and Martínez in SCG [2].

Concerning the stopping criterion to be used in steps 3, 8 and 12 we consider:

$$\|g_k\|_\infty \leq \varepsilon_g, \tag{19}$$

where $\|.\|_\infty$ denotes the maximum absolute component of a vector and ε_g is a tolerance specified by the user.

4. Convergence analysis for strongly convex functions

Throughout this section we assume that f is strongly convex and Lipschitz continuous on the level set

$$L_0 = \{x \in R^n : f(x) \leq f(x_0)\}. \tag{20}$$

That is, there exists constants $\mu > 0$ and L such that

$$(\nabla f(x) - \nabla f(y))^T (x - y) \geq \mu \|x - y\|^2 \tag{21}$$

and

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \tag{22}$$

for all x and y from L_0. For the convenience of the reader we include here the following lemma (see [7]).

Lemma 1. *Assume that d_k is a descent direction and ∇f satisfies the Lipschitz condition*

$$\|\nabla f(x) - \nabla f(x_k)\| \leq L\|x - x_k\|, \tag{23}$$

for every x on the line segment connecting x_k and x_{k+1}, where L is a constant. If the line search satisfies the second Wolfe condition (12), then

$$\alpha_k \geq \frac{1-\sigma_2}{L} \frac{|g_k^T d_k|}{\|d_k\|^2}. \tag{24}$$

Proof: Subtracting $g_k^T d_k$ from both sides of (12) and using the Lipschitz condition we have

$$(\sigma_2 - 1) g_k^T d_k \leq (g_{k+1} - g_k)^T d_k \leq L\alpha_k \|d_k\|^2. \tag{25}$$

Since d_k is a descent direction and $\sigma_2 < 1$, (24) follows immediately from (25). ■

Lemma 2. *Assume that ∇f is strongly convex and Lipschitz continuous on L_0. If θ_{k+1} is selected by spectral gradient, then the direction d_{k+1} given by (9) satisfies:*

$$\|d_{k+1}\| \leq \left(\frac{2}{\mu} + \frac{2L}{\mu^2} + \frac{L^2}{\mu^3}\right)\|g_{k+1}\|. \tag{26}$$

Proof: By Lipschitz continuity (22) we have

$$\|y_k\| = \|g_{k+1} - g_k\| = \|\nabla f(x_k + \alpha_k d_k) - \nabla f(x_k)\| \leq L\alpha_k \|d_k\| = L\|s_k\|. \tag{27}$$

On the other hand, by strong convexity (21)

$$y_k^T s_k \geq \mu \|s_k\|^2. \tag{28}$$

Selecting θ_{k+1} as in (18), it follows that

$$\theta_{k+1} = \frac{s_k^T s_k}{y_k^T s_k} \le \frac{\|s_k\|^2}{\mu \|s_k\|^2} = \frac{1}{\mu}. \tag{29}$$

Now, using the triangle inequality and the above estimates (27)-(29), after some algebra on $\|d_{k+1}\|$, where d_{k+1} is given by (9), we get (26). ■

The convergence of the scaled conjugate gradient algorithm (SCALCG) when f is strongly convex is given by

Theorem 2. *Assume that f is strongly convex and Lipschitz continuous on the level set L_0. If at every step of the conjugate gradient (2) with d_{k+1} given by (9) and the step length α_k selected to satisfy the Wolfe conditions (11) and (12), then either $g_k = 0$ for some k, or $\lim_{k\to\infty} g_k = 0$.*

Proof: Suppose $g_k \neq 0$ for all k. By strong convexity we have

$$y_k^T d_k = (g_{k+1} - g_k)^T d_k \ge \mu \alpha_k \|d_k\|^2. \tag{30}$$

By theorem 1, $g_k^T d_k < 0$. Therefore, the assumption $g_k \neq 0$ implies $d_k \neq 0$. Since $\alpha_k > 0$, from (30) it follows that $y_k^T d_k > 0$. But f is strongly convex over L_0, therefore f is bounded from below. Now, summing over k the first Wolfe condition (11) we have

$$\sum_{k=0}^{\infty} \alpha_k g_k^T d_k > -\infty.$$

Considering the lower bound for α_k given by (24) in lemma 1 and having in view that d_k is a descent direction it follows that

$$\sum_{k=1}^{\infty} \frac{|g_k^T d_k|^2}{\|d_k\|^2} < \infty. \tag{31}$$

Now, from (13), using the inequality of Cauchy and (28) we get

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

Therefore, from (31) it follows that

$$\sum_{k=0}^{\infty} \frac{\|g_k\|^4}{\|d_k\|^2} < \infty. \tag{32}$$

Now, inserting the upperbound (26), for d_k in (32) yields

$$\sum_{k=0}^{\infty} \|g_k\|^2 < \infty,$$

which completes the proof. ■

For general functions the convergence of the algorithm is coming from theorem 1 and the restart procedure. Therefore, for strongly convex functions and under inexact line search it is global convergent. To a great extent, however, the SCALCG algorithm is very close to the Perry/Shanno computational scheme [17, 18]. SCALCG is a scaled memoryless BFGS preconditioned algorithm where the scaling factor is the inverse of a scalar approximation of the Hessian. If the Powell restart criterion (14) is used, for general functions f bounded from below with bounded second partial derivatives and bounded level set, using the same arguments considered by Shanno in [18] it is possible to prove that the iterates either converge to a point x^* satisfying $\|g(x^*)\| = 0$, or the iterates cycle. It remains

for further study to determine a complete global convergence result and whether cycling can occur for general functions with bounded second partial derivatives and bounded level set.

More sophisticated reasons for restarting the algorithms have been proposed in the literature, but we are interested in the performance of an algorithm that uses the Powell restart criterion, associated with the scaled memoryless BFGS preconditioned direction choice for restart. Additionally, some convergence analysis with Powell restart criterion was given by Dai and Yuan [5] and can be used in this context of the preconditioned and scaled memoryless BFGS algorithm.

5. Computational results and comparisons

In this section we present the performance of a Fortran implementation of the *SCALCG - scaled conjugate gradient algorithm* on a set of 750 unconstrained optimization test problems. At the same time, we compare the performance of SCALCG with *the best spectral conjugate gradient algorithm, SCG* (betatype=1,Perry-M1), by Birgin and Martínez [2]; with the *CG_DESCENT - a conjugate gradient method with guaranteed descent* by Hager and Zhang [7, 8] and with the *Polak-Ribière algorithm - PR*.

The SCALCG code is authored by Andrei, while the SCG and Polak-Ribière are co-authored by Birgin and Martínez and CG_DESCENT is co-authored by Hager and Zhang. All codes are written in Fortran and compiled with f77 (default compiler settings) on an Intel Pentium 4, 1.5Ghz. The CG_DESCENT code contains the variant implementing the Wolfe line search (W) and the variant corresponding to the approximate Wolfe conditions (aW). All algorithms implements the same stopping criterion as in (19), where $\varepsilon_g = 10^{-6}$.

The test problems are the unconstrained problems in the CUTE [3] collection, along with other large-scale optimization problems. We selected 75 large-scale unconstrained optimization problems in extended or generalized form. For each function we have considered 10 numerical experiments with number of variables $n = 1000,2000,\ldots,10000.$

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:

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

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.

The numerical results concerning the number of iterations, the number of restart iterations, the number of function and gradient evaluations, cpu time in seconds, for each of these methods can be found in [1].

Tables 1-4 show the number of problems, out of 750, for which SCALCG versus SCG, PR, CG_DESCENT(W) or CG_DESCENT(aW) achieved the minimum number of iterations (#iter), the minimum number of function evaluations (#fg) and the minimum cpu time (CPU), respectively.

Table 1. Performance of SCALCG versus SCG. 750 problems.

	SCALCG	SCG	=
# iter	486	98	89
# fg	425	128	120
CPU	592	43	38

Table 2. Performance of SCALCG versus PR. 750 problems.

	SCALCG	PR	=
# iter	497	93	81
# fg	415	134	122
CPU	580	54	37

Table 3. Performance of SCALCG versus CG_DESCENT(W). 750 problems.

	SCALCG	CG_DESCENT(W)	=
# iter	440	173	28
# fg	444	164	33
CPU	515	91	35

Table 4. Performance of SCALCG versus CG_DESCENT(aW). 750 problems.

	SCALCG	CG_DESCENT(aW)	=
# iter	445	168	28
# fg	431	175	35
CPU	507	97	37

For example, when comparing SCALCG and SCG (Table 1), subject to the number of iterations, SCALCG was better in 486 problems (i.e. it achieved the minimum number of iterations in 486 problems), SCG was better in 98 problems, and they had the same number of iterations in 89 problems, etc. From these Tables we see that, at least for this set of 750 problems, the top performer is SCALCG. Since these codes uses the same Wolfe line search, excepting CG_DESCENT(aW) which implements an approximate Wolfe line search, and the same stopping criterion they differ in their choice of the search direction. Hence, among these conjugate gradient algorithms, SCALCG appears to generate the best search direction, on average.

6. Conclusion

We have presented a scaled memoryless BFGS preconditioned conjugate gradient algorithm, *SCALCG - scaled conjugate gradient algorithm*, for solving large-scale unconstrained optimization problems. SCALCG can be considered as a modification of the best algorithm by Birgin and Martínez [2], which is mainly a scaled variant of Perry's [12], and of the Dai and Liao [4] $(t = 1)$, in order to overcome the lack of positive definiteness of the matrix defining the search direction. This modification takes the advantage of the quasi-Newton BFGS updating formula. Using the restart technology of Beale-Powell, we get a scaled conjugate gradient algorithm in which the parameter scaling the gradient is selected as spectral gradient. Although the update formulas (9) and (15)-(17) are more complicated the scheme proved to be efficient and robust in numerical experiments. The algorithm implements the Wolfe conditions and we have proved that the steps are along the descent directions.

The performances for our scaled conjugate gradient algorithm were higher than those of *SCG - spectral conjugate gradient method* by Birgin and Martínez, *CG_DESCENT - conjugate gradient with guaranteed descent* by Hager and Zhang and *scaled Polak-Ribière* for a set of 750 unconstrained optimization problems with dimensions ranging between 10^3 and 10^4.

As each of these codes considered here are different, mainly in the amount of linear algebra required at each step, it is quite clear that different codes will be superior in different

problem sets. Generally, one needs to solve thousands different problems before trends begin to emerge. For any particular problem almost any method can win. However, we have strong computational evidence that our scaled memoryless BFGS preconditioned conjugate gradient algorithm is the top performer among these conjugate gradient algorithms.

References

[1] Andrei, N., http://www.ici.ro/camo/neculai/anpaper.htm/

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

[3] Bongartz, I, Conn, A.R., Gould, N.I.M. and Toint, P.L., 1995, *CUTE: constrained and unconstrained testing environments*, ACM Trans. Math. Software, 21, pp.123-160.

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

[5] Dai Y.H. and Yuan, Y., 1998, *Convergence properties of the Beale-Powell restart algorithm*, Science in China (series A) 41, (11), 1142-1150.

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

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

[8] Hager, W.W. and Zhang, H., 2004, *CG-DESCENT, A conjugate gradient method with guaranteed descent (algorithm details and comparisons)*, University of Florida, Department of Mathematics, January 15, 2004.

[9] Hestenes, M.R. and Stiefel, E., 1952, *Methods of conjugate gradients for solving linear systems*, J. Research Nat. Bur. Standards Sec. B. 48, pp. 409-436.

[10] Oren, S.S. and Luenberger, D.G., 1976, *Self-scaling variable metric algorithm. Part I*, Management Sci., 20, pp.845-862.

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

[12] Perry, J.M., 1977, *A class of conjugate gradient algorithms with a two step variable metric memory*, Discussion paper 269, Center for Mathematical Studies in Economics and Management Science, Northwestern University, 1977.

[13] Polak E. and Ribière, G., 1969, *Note sur la convergence de methods de directions conjugres*, Revue Francaise Informat. Reserche Opérationnelle 16, pp. 35-43.

[14] Powell, M.J.D., 1976, *Some convergence properties of the conjugate gradient method*. Math. Programming, 11, pp.42-49.

[15] Powell, M.J.D., 1977, *Restart procedures for the conjugate gradient method*, Math. Programming, 12, pp.241-254.

[16] Raydan, M., 1997, *The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem*, SIAM J. Optim., 7, 26-33.

[17] Shanno, D.F., 1978, *Conjugate gradient methods with inexact searches*, Mathematics of Operations Research, vol. 3, pp.244-256.

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

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

[20] Wolfe, P., 1969, *Convergence conditions for ascent methods*, SIAM Rev., 11, pp.226-235.

[21] Wolfe, P., 1971, *Convergence conditions for ascent methods II: some corrections*, SIAM Rev. 13, pp.185-188.

May 23, 2006