A New Gradient Descent Method with an Anticipative Scalar Approximation of Hessian 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 work we present a new method for steplength selection in the frame of gradient descent methods for functions minimization. The corresponding algorithm selects the steplength, according to a backtracking procedure, along the negative gradient, in which the initial stepsize is determined using a new scalar approximation of the Hessian of the minimizing function. This approximation is based only on the function values and its gradient in two successive points along the iterations. Like other gradient descent algorithms, this one is also linear convergent. The main advantage of this computational scheme is that at every iteration we generate a positive, occasionaly greater than 1, initial stepsize in the backtracking procedure, this giving us the possibility to continue and accelerate the convergence of the algorithm. The numerical experience shows that the algorithm compares favourable with the Barzilai-Borwein approach.

Keywords: Unconstrained optimization, gradient descent methods, Barzilai-Borwein method
AMS Subject Classification: 65K05, 90C30

1. Introduction

For the unconstrained optimization problem

$$\min_{x\in R^n} \; f(x) \tag{1}$$

where $f:R^n \to R$ is a continuously differentiable function, in 1988 Barzilai and Borwein [2] proposed a gradient descent method (BB method) which essentially is a steepest descent method, where the choice of the steplength along the negative gradient direction is obtained from a two-point approximation to the secant equation underlying quasi-Newton methods. More specifically, considering $D_k = \gamma_k I$ as an approximation to the Hessian of f at x_k, they choose γ_k such that

$$D_k = argmin \left\| Ds_k - y_k \right\|_2,$$

where $s_k = x_k - x_{k-1}$ and $y_k = \nabla f(x_k) - \nabla f(x_{k-1})$, yielding

$$\gamma_k^{BB} = \frac{s_k^T y_k}{s_k^T s_k}. \tag{2}$$

With these, the method of Barzilai and Borwein is given by the following iterative scheme:

$$x_{k+1} = x_k - \frac{1}{\gamma_k^{BB}} \nabla f(x_k). \tag{3}$$

The quantity γ_k^{BB} given in (2) is frequently referred as a Rayleigh quotient. Indeed, if f is twice continuously differentiable, we have

$$y_k = \left[\int_0^1 \nabla^2 f(x_k + \tau s_k) d\tau\right] s_k .$$

Therefore:

$$\gamma_k^{BB} = s_k^T \left[\int_0^1 \nabla^2 f(x_k + \tau s_k) d\tau\right] s_k / s_k^T s_k , \tag{4}$$

which lies between the largest and the smallest eigenvalue of the Hessian average

$$\int_0^1 \nabla^2 f(x_k + \tau s_k) d\tau .$$

The scalar γ_k^{BB} has been already used as scaling factor in the context of limited memory quasi-Newton algorithms (see, for example Gilbert and Lemaréchal [9], Liu and Nocedal [14]) or conjugate gradient algorithms (Shanno and Phua [20], Andrei [1]).

Having in view its simplicity and numerical efficiency for well-conditioned problems, the BB gradient method has received a great deal of attention. In [2] Barzilai and Borwein proved that for the two-dimensional quadratic case the BB method is R-superlinear convergent. They present some numerical evidence showing that their method is remarkably superior to the classical gradient descent method for a quadratic function with four variables. Raydan [18] proved that for strictly convex quadratic case with any number of variables the BB method is globally convergent. Using a globalization strategy, based on the non-monotone line search technique introduced by Grippo, Lampariello and Lucidi [12], Raydan [19] proved the global convergence of the BB method for non-quadratic functions and reports some numerical evidence on problems up to 10^4 variables showing that the BB method is competitive with the conjugate gradient Polak-Ribière [17] and CONMIN of Shanno and Phua [20] methods. A preconditioning technique for the BB method has been considered by Molina and Raydan [15]. Under a very restrictive assumption they established the Q-linear rate of convergence of the preconditioned BB method. Some applications of preconditioned BB method on a distance matrix problem are considered by Glunt, Hayden and Raydan [10,11]. Extension of the BB method for box-constrained optimization problems have been considered by Friedlander, Martinez and Raydan [8] (for quadratic function) and by Birgin, Martinez and Raydan [3].

An analysis of the BB method stressing the importance of non-monotone line search as well as some open problems are presented by Fletcher [7] and Dai and Fletcher [5]. Dai and Liao [6] refined the analysis in Raydan [18] and proved that the convergence rate is R-linear. New globalization strategies for BB method, based on relaxations of the monotonicity requirements, are considered by Grippo and Sciandrone [13] where the nonmonotone watchdog technique with nonmonotone linesearch rules are combined. Their algorithms are very sophisticated and dependent of a number of parameters. Numerical experience and comparisons with E04DGF routine of NAG library on some collections of problems, including CUTE [4], shows that their globalization strategy for the BB algorithm compares favorables with E04DGF algorithm. (However, for some difficult ill-conditioned problems, algorithm E04DGF is more efficient.)

The purpose of this paper is to present a new algorithm of gradient descent type, in which the initial step size in a backtracking procedure is computed by means of a simple approximation to the Hessian of the minimizing function. In contrast with the Barzilai and Borwein approach in which the steplength is computed from a simple interpretation of the secant equation, the new proposed algorithm considers another approximation of the Hessian based on the function values and its gradients in two successive points along the iterations. The corresponding algorithm belongs to the same class of linear convergent descent methods and compares favourable with the Barzilai-Borwein approach. The conclusion is that using only the local information given by the gradient, any procedure for step size computation, of any sophistication, does not change the linear convergence class of algorithms.

2. Gradient Descent Method with an Anticipative Scalar Approximation of Hessian

In the following we suggest another procedure for computing a scalar approximation of the Hessian of the function f at x_k which can be used to get the stepsize along the nagative gradient. Let us consider the initial point x_0 where $f(x_0)$ and $g_0 = \nabla f(x_0)$ can immediately be computed. Using the backtracking procedure (initialized with $t = 1$) we can compute the steplength $t_0 = \underset{t<1}{argmin}\, f(x_0 - tg_0)$, with which the next estimate $x_1 = x_0 - t_0 g_0$ is computed, where again we can determine $f(x_1)$ and $g_1 = \nabla f(x_1)$. So, the first step is computed using the backtracking along the negative gradient. Now, at point $x_{k+1} = x_k - t_k g_k$, $k = 0,1,\ldots$ we have

$$f(x_{k+1}) = f(x_k) - t_k g_k^T g_k + \frac{1}{2} t_k^2 g_k^T \nabla^2 f(z) g_k, \tag{5}$$

where z is on the line segment connecting x_k and x_{k+1}. Having in view the local character of the searching procedure and that the distance between x_k and x_{k+1} is enough small we can choose $z = x_{k+1}$ and consider γ_{k+1} as a scalar approximation of the $\nabla^2 f(x_{k+1})$, where $\gamma_{k+1} \in R$. This is an anticipative view point, in which a scalar approximation to the Hessian at point x_{k+1} is computed using only the local information from two successive points: x_k and x_{k+1}. Therefore, we can write:

$$\gamma_{k+1} = \frac{2}{g_k^T g_k} \frac{1}{t_k^2} \left[f(x_{k+1}) - f(x_k) + t_k g_k^T g_k \right]. \tag{6}$$

Observe that at x_{k+1} , $k = 0,1,\ldots$ we know $f(x_{k+1})$, g_{k+1} and an approximation of $\nabla^2 f(x_{k+1})$ as $\gamma_{k+1} I$. Now, in order to compute the next estimation $x_{k+2} = x_{k+1} - t_{k+1} g_{k+1}$ we must consider a procedure to determine the steplength t_{k+1}. For this let us consider the function:

$$\Phi_{k+1}(t) = f(x_{k+1}) - t g_{k+1}^T g_{k+1} + \frac{1}{2} t^2 \gamma_{k+1} g_{k+1}^T g_{k+1}. \tag{7}$$

Observe that $\Phi_{k+1}(0) = f(x_{k+1})$, $\Phi_{k+1}(2/\gamma_{k+1}) = f(x_{k+1})$ and $\Phi'_{k+1}(0) = -g_{k+1}^T g_{k+1} < 0$. Therefore, $\Phi_{k+1}(t)$ is a convex function for all $t \geq 0$. To have a minimum for $\Phi_{k+1}(t)$ we must have $\gamma_{k+1} > 0$. Considering for the moment that $\gamma_{k+1} > 0$, then from $\Phi'_{k+1}(t) = 0$ we get

$$\bar{t}_{k+1} = \frac{1}{\gamma_{k+1}}, \tag{8}$$

as the minimum point of $\Phi_{k+1}(t)$. Observe that

$$\Phi_{k+1}(\bar{t}_{k+1}) = f(x_{k+1}) - \frac{1}{2\gamma_{k+1}} \left\| g_{k+1} \right\|_2^2, \tag{9}$$

showing that, if $\gamma_{k+1} > 0$, then at every iteration the value of function f is reduced. This suggests us to determine the stepsize t_{k+1} as:

$$t_{k+1} = \underset{t \leq \bar{t}_{k+1}}{argmin}\, f(x_{k+1} - t g_{k+1}), \tag{10}$$

using for example a backtracking procedure and to consider $x_{k+2} = x_{k+1} - t_{k+1} g_{k+1}$. To complete the algorithm we must consider the situation when $\gamma_{k+1} < 0$ and to detail the backtracking procedure.

Concerning the backtracking procedure we consider the Armijo's rule. This procedure selects two scalars $0 < \alpha \leq 0.5,\ 0 < \beta < 1$ and takes the following steps:

Backtracking procedure

Step 1. Set: $t = \bar{t}_{k+1}$ and $f_{\min} = min\{f(x_j), j = 0,1,\ldots,k+1\}$.

Setp 2. While $f(x_{k+1} - tg_{k+1}) > f_{\min} - \alpha t g_{k+1}^T g_{k+1}$, set $t = t\beta$.

Step 3. Set $t_{k+1} = t$.

Therefore, the initialization of the backtracking procedure is given by $\bar{t}_{k+1}$ and in Armijo rule the function values are compared with $f_{\min}$. Typically, $\alpha = 0.0001$ and $\beta = 0.8$, meaning that we accept a small decrease in f of the prediction based on the linear extrapolation.

Observe that for convex functions $\gamma_{k+1} > 0$. If $f(x_{k+1}) - f(x_k) + t_k g_k^T g_k < 0$, then the reduction $f(x_{k+1}) - f(x_k)$ in function value is smaller than $-t_k g_k^T g_k$. In these cases the idea is to change a little the stepsize t_k as $t_k + \eta_k$, maintaining the other quantities at their values, in such a manner that γ_{k+1} to be positive. To get a value for η_k let us select a real $\delta > 0$, "enough small", but comparable with the current value of the function, ($\delta = \varepsilon_a |f(x_{k+1})|,\ \varepsilon_a = 10^{-2}$) and consider

$$\eta_k = \frac{1}{g_k^T g_k}\left[f(x_k) - f(x_{k+1}) - t_k g_k^T g_k + \delta\right], \tag{11}$$

with which a new value for γ_{k+1} can be computed as:

$$\gamma_{k+1} = \frac{2}{g_k^T g_k}\frac{1}{(t_k + \eta_k)^2}\left[f(x_{k+1}) - f(x_k) + (t_k + \eta_k) g_k^T g_k\right]. \tag{12}$$

With these, the corresponding Anticipative Algorithm is as follows:

The Anticipative Algorithm (AA)

Step 1. Select $x_0 \in dom\, f$ and compute f_0, g_0 and $t_0 = \underset{t<1}{argmin}\, f(x_0 - tg_0)$. Set $k = 0$ and $f_{\min} = f_0$.

Step 2. Compute $x_{k+1} = x_k - t_k g_k$, f_{k+1} and g_{k+1}. If $f_{k+1} < f_{\min}$, then $f_{\min} = f_{k+1}$.

Step 3. Compute γ_{k+1} as in (6). If $\gamma_{k+1} < 0$, then select $\delta > 0$ and compute a new value for γ_{k+1} as in (12), where η_k is given by (11).

Step 4. Compute the initial stepsize $\bar{t}_{k+1} = 1/\gamma_{k+1}$.

Step 5. Using the above backtracking procedure determine the steplength t_{k+1}.

Step 6. Test for continuation. If

$$\left\|g_{k+1}\right\|_\infty \leq \varepsilon_g \quad \text{or} \quad t_{k+1}\left|g_{k+1}^T g_{k+1}\right| \leq \varepsilon_f \left|f_{k+1}\right|, \tag{13}$$

then stop; otherwise set $k = k+1$ and continue with step 2.

If in step 3 of algorithm AA instead of γ_{k+1} we consider γ_{k+1}^{BB}, then we get a variant of Barzilai-Borwein algorithm with line search, called algorithm BB.

The analysis of convergence of this algorithm is given by the following results.

Proposition 1. *For every* $k = 0,1,\ldots$ γ_k *generated by the algorithm AA is bounded away from zero.*

Proof. For every $k = 0,1,\ldots$ we know that $f(x_{k+1}) - f(x_k) + t_k g_k^T g_k > 0$. Therefore, $f(x_k) - f(x_{k+1}) < t_k g_k^T g_k$. With this we have:

$$\gamma_{k+1} = \frac{2}{t_k} - \frac{2(f(x_k) - f(x_{k+1}))}{t_k^2 (g_k^T g_k)} > \frac{2}{t_k} - \frac{2t_k (g_k^T g_k)}{t_k^2 (g_k^T g_k)} = 0. \quad \blacksquare$$

Therefore, the step 4 of AA is well defined. However, towards the final iterations of the algorithm, especially when the accuracy requirements are too high, it is possible that $(f(x_{k+1}) + t_k g_k^T g_k) - f(x_k) < 0,$ but very close to zero. This is because $g_k^T g_k$ is too small. That means the reduction in function values is too small. The remedy, we have suggested here in this situations, is to increase a little the steplength in order to compensate the accuracy requirements.

In the following let us assume that f is strongly convex and the sublevel set $S = \{x \in dom\, f : f(x) \le f(x_0)\}$ is closed. Strong convexity of f on S involves that there exists the constants m and M such that $mI \le \nabla^2 f(x) \le MI,$ for all $x \in S.$ A consequence of strong convexity of f is that we can bound f^*, the value of function f in local minimum, as:

$$f(x) - \frac{1}{2m}\|\nabla f(x)\|_2^2 \le f^* \le f(x) - \frac{1}{2M}\|\nabla f(x)\|_2^2. \tag{14}$$

In this circumstances the following theorem can be proved.

Theorem 1. *For strongly convex functions the AA algorithm with backtracking is linear convergent and*

$$f(x_k) - f^* \le \left(\prod_{i=0}^{k-1} c_i\right)(f(x_0) - f^*),$$

where $c_i = 1 - min\{2m\alpha \bar{t}_{i+1}, 2m\alpha \bar{t}_{i+1}\beta^{p_i}\} < 1$ *and* $p_i \ge 0$ *is an integer* ($p_i = 0,1,2,\ldots$ *given by the backtracking procedure).*

Proof. We can write:

$$f(x_{k+1}) = f(x_k) - \left(t - \frac{1}{2}t^2\gamma_{k+1}\right)\|g_k\|_2^2.$$

But, $t - t^2\gamma_{k+1}/2$ is a concave function, and for all $0 \le t \le 1/\gamma_{k+1},$ it follows that $t - t^2\gamma_{k+1}/2 \ge t/2.$ Hence,

$$f(x_{k+1}) \le f(x) - \frac{t}{2}\|g_k\|_2^2 \le f(x_k) - \alpha t\|g_k\|_2^2,$$

since $\alpha \le 1/2.$ The backtracking linesearch procedure terminates either with $t = \bar{t}_{k+1}$ $(p_k = 0)$ or a value $t = \bar{t}_{k+1}\beta^{p_k},$ where $p_k > 1$ is an integer given by the backtracking procedure. With this, at step k we can get a lower bound on the decrease of the function. In the first case we have:

$$f(x_{k+1}) \le f(x_k) - \alpha \bar{t}_{k+1}\|g_k\|_2^2,$$

and in the second one

$$f(x_{k+1}) \le f(x_k) - \alpha \bar{t}_{k+1}\beta^{p_k}\|g_k\|_2^2.$$

Therefore,

$$f(x_{k+1}) \le f(x_k) - min\{\alpha \bar{t}_{k+1}, \alpha \bar{t}_{k+1}\beta^{p_k}\}\|g_k\|_2^2.$$

Hence

$$f(x_{k+1}) - f^* \le f(x_k) - f^* - min\{\alpha \bar{t}_{k+1}, \alpha \bar{t}_{k+1}\beta^{p_k}\}\|g_k\|_2^2.$$

Having in view the strong convexity, from (14), we have $\|g_k\|_2^2 \geq 2m\left(f(x_k)-f^*\right)$. Hence,

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

where

$$c_k = 1-min\left\{2m\alpha\bar{t}_{k+1}, 2m\alpha\bar{t}_{k+1}\beta^{p_k}\right\}.$$

Since $c_k < 1$, the sequence $f(x_k)$ is linear convergent, like a geometric series, to f^*. ■

3. Numerical Results

In this section we present the performance of a Fortran implementation of these algorithms. All codes are written in standard Fortran and compiled with f77 (default compiler settings) on a Pentium 1.5Ghz. For the very beginning, in order to illustrate the behaviour of the Anticipative Algorithm (AA) in comparison with the Barzilai-Borwein algorithm (BB) in this formulation, let us consider the Freudenstein & Roth function:

$$f(x)=\sum_{i=1}^{n/2}\left(-13+x_{2i-1}+((5-x_{2i})x_{2i}-2)x_{2i}\right)^2+$$
$$\left(-29+x_{2i-1}+((x_{2i}+1)x_{2i}-14)x_{2i}\right)^2.$$

Considering $x_0=[0.5,-2,\ldots,0.5,-2]$, $\alpha=0.0001$ and $\beta=0.8$, in backtracking procedure, as well as $\varepsilon_g=10^{-6}$ and $\varepsilon_f=10^{-20}$ in the criteria (13) for stopping the iterations, than for $n=1000$, the evolution of $\left|f(x_k)-f^*\right|$ given by the AA and BB algorithms are presented in Figure 1.

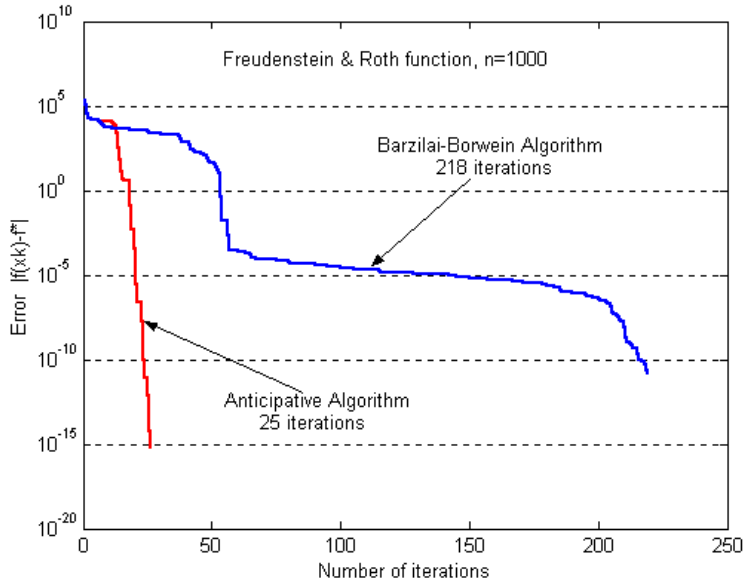

Fig. 1. AA versus BB algorithms.

We see that, for this example, subject to the number of iterations, the AA algorithm is about 8.7 times more performant than BB. It worth saying that, for algorithm AA we have $0.0002269 \leq \bar{t}_k = 1/\gamma_k \leq 0.34479$, for $k=1,2,\ldots,25$. At the same time for algorithm BB $0.00023029 \leq \bar{t}_k = 1/\gamma_k^{BB} \leq 0.34477$, for $k=1,2,\ldots,218$, showing that the initial steplength used in backtracking procedure has an acceptable value. However, for general functions the numbers γ_k and γ_k^{BB} can be unacceptable large or small, or even $\gamma_k^{BB}<0$ for nonconvex functions. Therefore, we must assume that the initial steplength computed through $\bar{t}_k = 1/\gamma_k$ or $\bar{t}_k = 1/\gamma_k^{BB}$ is modified so as to satisfy a bounding condition of the

form $0 < \bar{t}_l \le \bar{t}_k \le \bar{t}_u$ for all k, where $\bar{t}_l$ and $\bar{t}_u$ are numbers specified by the user. The Armijo line search, used in the backtracking, does not permit any increase of the stepsize. Therefore, it is necessary to implement some rules that occasionally admit increases in the stepsize. Such a procedure, for example, was proposed by Grippo and Sciandrone [13] based on a combination of nonmonotone watchdog techniques with nonmonotone linesearches. However, in this paper we did not consider this approach of line search because the main interests were directed to suggest and to present the numerical evidence of an alternative scalar approximation of the Hessian to that of Barzilai and Borwein's, in the frame of gradient descent algorithm.

The number of iterations (iter), function and its gradient evaluations (fg) and cpu time in centeseconds (cpu(c)) for minimizing this function, for $n = 1000,2000,\ldots,10000$, corresponding to AA and BB algorithms are presented in Table 1.

Table 1. AA versus BB algorithms for function $f(x)$ with $n = 1000,2000,\ldots,10000$.

	AA			BB		
n	**iter**	**fg**	**cpu(c)**	**iter**	**fg**	**cpu(c)**
1000	25	194	22	218	1615	204
2000	25	194	50	240	1762	445
3000	25	194	71	240	1771	670
4000	25	194	99	145	1086	554
5000	25	194	121	138	1028	654
6000	25	194	148	285	2096	1615
7000	25	194	171	175	1302	1170
8000	25	194	192	295	2173	2246
9000	25	194	225	170	1254	1456
10000	25	194	247	140	1047	1356
TOTAL	**250**	**1940**	**1346**	**2046**	**15134**	**10370**

In the following we present the performance of these algorithms, subject to cpu time metric, on a number of 310 unconstrained optimization test functions. The test problems are the unconstrained problems in the CUTE library [4], along with other large-scale optimization test problems. We selected 31 large-scale unconstrained optimization test problems, in extended or generalized form, and for each test function we have considered 10 numerical experiments with number of variables $n = 1000,2000,\ldots,10000$. The numerical results concerning the number of iterations, number of function and gradient evaluations, cpu time in seconds, the minimum and maximum values of the initial stepsize, for each of these algorithms are posted at the following web site: http://www.ici.ro/camo/neculai/ansoft.htm/asa. The main indicator of performance is the relative cpu time of algorithms AA and BB for solving a problem, as sugested by Morales [16], and measured by

$$time_i = -\log_2\left(\frac{time_{AA}^i}{time_{BB}^i}\right)$$

where $time_{AA}^i$ is the cpu time for solving the $i-$th problem by algorithm AA. The sign of $time_i$ indicates the winer. In all cases in which AA wins $time_i$ has a positive value. The number of times by which the winer outperforms the loser is $2^{|time_i|}$. We refer this number as the outperforming factor. In Figure 2 we display the values of $time_i$ for the set of problems considered in this experiment, where the problems have been placed in decreasing order with respect to their values of $|time_i|$.

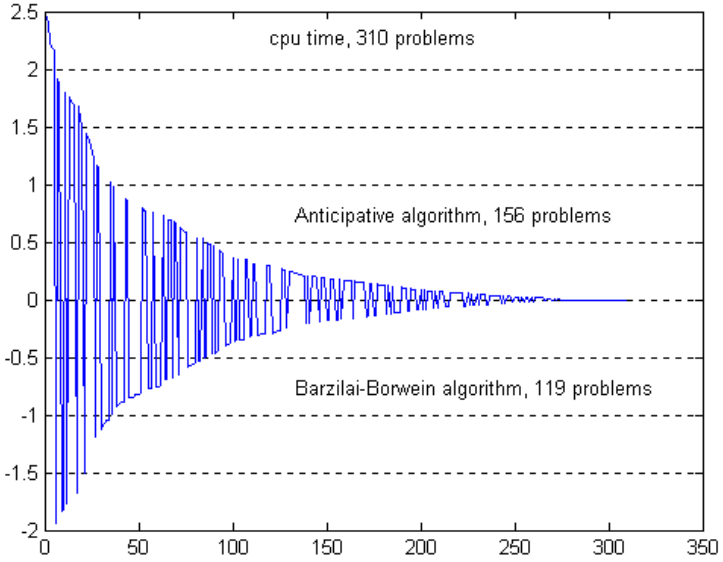

Fig.2. Cpu time performance profile for 310 problems. AA versus BB.

We observe that for this set of experiments the AA algorithm outperforms BB, out of 310 problems, for 156 the AA algorithm was faster than BB. However, the gain is only marginal, showing that the anticipative procedure for a scalar approximation of Hessian compares favourable to that of Barzilai and Borwein's.

4. Conclusion

The paper proposes a new scalar approximation to the Hessian of a minimizing function which is based on the function values and its gradient in two successive points along the iterations. This approximation, computed at step k is used to determine the initial stepsize for a backtracking procedure which compute the steplength for the next iteration $k+1$. The anticipative scheme, suggested here, is more robust than that of Barzilai and Brown which often hapens to generate too small or even negative estimates for the initial steplength. The advantage of this computational scheme is that at every iteration we generate a positive scalar approximation of the Hessian, this giving us the possibility to continue the algorithm. Like BB method, the anticipative strategy for steplength computation has a limited value in the practice of gradient descent algorithms. The convergence of resulting algorithms, even for strongly convex functions, is only linear. However, this idea of anticipative approach for scalar approximation of Hessian is more profitable in the frame of scaled conjugate gradient computational scheme [1].

References

[1] N. Andrei. Scaled Conjugate gradient algorithms for unconstrained optimization. ICI Technical Report, March 11, 2005.

[2] J. Barzilai and J.M. Borwein. Two point step size gradient method. *IMA J. Numer. Anal.,* **8** (1988) pp.141-148.

[3] E.G. Birgin, J.M. Martinez and M. Raydan. Nonmonotone spectral projected gradient methods on convex sets. *SIAM J. Optim.*, **10** (2000) pp.1196-1211.

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

[5] Y.H. Dai and R. Fletcher. On the asymptotic behaviour of some new gradient methods. *Math. Programming, Series A, Published online*: February 9, 2005.

[6] Y.H. Dai and L.Z. Liao. R-linear convergence of the Barzilai and Borwein gradient method. *IMA J. Numer. Anal.*, 22 (2002) pp.1-10.
[7] R. Fletcher. On the Barzilai-Borwein method. Numerical Analysis Report NA/207, 2001.
[8] A. Friedlander, J.M. Martinez and M. Raydan. A new method for large-scale box constrained convex quadratic minimization problems. *Optimization Methods and Software*, **5** (1995) pp.55-74.
[9] J. Gilbert and C. Lemaréchal. Some numerical experiments with variable-storage quasi-Newton algorithms. *Math. Programming, Series B*, **45** (1989) pp.407-435.
[10] W. Glunt, T.L. Hayden and M. Raydan. Molecular conformations from distance matrices, *J. Comput. Chem.*, 14 (1993) pp.114-120.
[11] W. Glunt, T.L. Hayden and M. Raydan. Preconditioners for distance matrix algorithms. *J. Comput. Chem.,* **15** (1994) pp.227-232.
[12] L. Grippo, F. Lampariello and S. Lucidi. A nonmonotone line search technique for Newtons method. *SIAM J. Numer. Anal.*, **23** (1986) pp.707-716.
[13] L. Grippo and M. Sciandrone. Nonmonotone globalization techniques for the Barzilai-Borwein gradient method. *Computational Optimization and Applications*, **23** (2002) pp.143-169.
[14] D.C. Liu and J. Nocedal. On the limited-memory BFGS method for large scale optimization. *Math. Programming*, **45** (1989) pp.503-528.
[15] B. Molina and M. Raydan. Preconditioned Barzilai-Borwein method for the numerical solution of partial differential equations. *Numerical Algorithms*, 13 (1996) pp.45-60.
[16] J.L. Morales, A numerical study of limited memory BFGS methods. *Applied Mathematics Letters,* **15** (2002) pp. 481-487.
[17] E. Polak and G. Ribière. Note sur la convergence de methodes de directions conjugat. *Revue Francaise d'Informatique et Recherche Operationnelle*, **16** (1969) pp.35-43.
[18] M. Raydan. On the Barzilai and Borwein choice of the steplength for the gradient method. *IMA J. Numer. Anal.*, **13** (1993) pp.618-622.
[19] M. Raydan. The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem. *SIAM J. Optim.*, **7** (1997) pp.26-33.
[20] D.F. Shanno and K.H. Phua. Remark on Algorithm 500: Minimization of unconstrained multivariate functions. *ACM Trans. Math. Software*, **6** (1980) pp.618-622.

March 25, 2005

-----ooooo0ooooo-----