OPTIMIZATION, 2017
http://dx.doi.org/10.1080/02331934.2017.1287702

An adaptive nonmonotone global Barzilai–Borwein gradient method for unconstrained optimization

Hadi Nosratipour, Omid Solaymani Fard and Akbar Hashemi Borzabadi

Department of Applied Mathematics, School of Mathematics and Computer Science, Damghan University, Damghan, Iran

ABSTRACT
The Barzilai–Borwein (BB) gradient method has received many studies due to its simplicity and numerical efficiency. By incorporating a nonmonotone line search, Raydan (SIAM J Optim. 1997;7:26–33) has successfully extended the BB gradient method for solving general unconstrained optimization problems so that it is competitive with conjugate gradient methods. However, the numerical results reported by Raydan are poor for very ill-conditioned problems because the effect of the degree of nonmonotonicity may be noticeable. In this paper, we focus more on the nonmonotone line search technique used in the global Barzilai–Borwein (GBB) gradient method. We improve the performance of the GBB gradient method by proposing an adaptive nonmonotone line search based on the morphology of the objective function. We also prove the global convergence and the R-linear convergence rate of the proposed method under reasonable assumptions. Finally, we give some numerical experiments made on a set of unconstrained optimization test problems of the CUTEr collection. The results show the efficiency of the proposed method in the sense of the performance profile introduced (Math Program. 2002;91:201–213) by Dolan and Moré.

ARTICLE HISTORY
Received 13 July 2016
Accepted 13 January 2017

KEYWORDS
Unconstrained optimization; Barzilai–Borwein gradient method; nonmonotone line search; global convergence

1. Introduction

Consider the unconstrained optimization problem

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & x \in \mathbb{R}^n, \end{array} \tag{1}$$

where f is a non-linear continuously differentiable real-valued function, and its first-order information (function value and gradient) is available.

The unconstrained optimization problems appear in a wide range of applications, including estimation of the optical constants and the thickness of thin films [1], optimal control [2], the neural network supervised learning [3] and training algorithms for recurrent neural networks [4]. Thus, the development of numerical algorithms for solving problem (1) is very important in the theoretical and computational points of view especially when we face large-scale problems.

There are many iterative schemes for solving the problem (1) that most of them can be classified into one of three categories: zero-, first- and second-order solvers. In general, zero-order methods also called derivative-free are not efficient enough and cannot provide a rich convergence theory for general problem (1). Besides, second-order methods need to deal with the Hessian matrix and

CONTACT Omid Solaymani Fard osfard@du.ac.ir

its approximations, which require a large amount of memory (except if a limited memory version is employed). On the other hand, first-order methods take advantage of a rich convergence theory, simple structures and low memory requirement. These factors make them eligible candidates for handling large-scale optimization problems of the form (1).

Regarding the technique used to achieve the monotone convergence, nonlinear optimization methods can be organized into two boarder classes: LINE SEARCH (LS) methods and TRUST REGION methods. In LS methods, the problem is reduced to one dimension in every iteration by choosing a search direction and considering the objective function only along that line. On the other hand, trust region methods try to find a neighbourhood around the current step x_k in which a quadratic or conic model function should be agreed with the objective function.

Let us denote by x_k and d_k the current iteration and a descent search direction constructed by first-order information, respectively. The basic structure of the kth iteration for a LS method is given as

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

where α_k is a stepsize obtained by a LS rule. The different choices of α_k and d_k seriously affect the efficiency of iterative methods.

There are effective LS methods for solving (1), for instance, the inexact Newton method, limited memory quasi-Newton method, non-linear conjugate gradient (CG) method and spectral gradient method [5–12].

The steepest descent (SD) method starts from an initial point x_0 and generates a new iteration by the rule

$$x_{k+1} = x_k - \alpha_k g_k, \tag{3}$$

where α_k is determined by the exact line search as

$$\alpha_k = \arg\min_{\alpha>0} f(x_k - \alpha g_k). \tag{4}$$

Akaike [13] shows that the SD method can be very slow when the Hessian of f is ill-conditioned at a local minimum. In this situation, the iterations slowly approach the minimum in a zigzag manner, which usually implies deteriorations in convergence. Early efforts to improve this method gave rise to the development of the CG method [7]. Another significant development of the SD method that avoids the drawbacks of SD method is due to Barzilai and Borwein [14]. They proposed two choices for the stepsize by regarding $D_k = \gamma_k I$ as an approximation to the Hessian of f at x_k and imposed a certain quasi-Newton property on D_k (approximation to the secant equation). They chose the stepsize γ_k such that

$$D_k = \arg\min_{D=\gamma I} \|D s_{k-1} - y_{k-1}\|, \tag{5}$$

and obtained

$$\gamma_k^{BB1} = \frac{y_{k-1}^T s_{k-1}}{s_{k-1}^T s_{k-1}}, \tag{6}$$

where $s_{k-1} = x_k - x_{k-1}$ and $y_{k-1} = g_k - g_{k-1}$. By regarding D_k^{-1} as an approximation to the inverse Hessian of f at x_k and choosing D_k^{-1} such that

$$D_k^{-1} = \arg\min_{D=\gamma^{-1}I} \|s_{k-1} - D^{-1} y_{k-1}\|, \tag{7}$$

Barzilai and Borwein also obtained the following choice for γ_k as

$$\gamma_k^{BB2} = \frac{y_{k-1}^T y_{k-1}}{y_{k-1}^T s_{k-1}}. \tag{8}$$

Moreover, some practical experiments have shown the choice (8) often performs worse than (6) in practical computation. The BB method performs much better than the gradient method in practice (see e.g. [15]).

Barzilai and Borwein [14] proved that the BB method is R-superlinearly convergent in the two-dimensional quadratic case. In the strictly convex quadratic case with any number of variables, it has been demonstrated in [16] that it is globally convergent and in [17] that the convergence rate is R-linear. Since the BB method is also locally R-linearly convergent for general objective functions and does not enforce decrease in the objective function, hence the BB stepsize will be accepted by nonmonotone line search rules with appropriate parameters when the iterate is close to the solution.

We also note that a nonmonotone globalization technique can be useful in difficult nonlinear problems independently of the specific local properties of the default stepsize because it may help escaping from steep sided valleys and may improve both the possibility of finding the global optimum and the rate of convergence [6,18–21].

Raydan [16] proposed a globalization strategy based on the nonmonotone line search of Grippo et al. [6] and extended the BB method by choosing (6) to solve a general unconstrained optimization. More precisely, it is based on an Armijo-type line search

$$f(x_k + \alpha_k d_k) \le f_{l(k)} + \delta \alpha_k g_k^T d_k, \tag{9}$$

where $\delta, \alpha_k \in (0, 1)$,

$$f_{l(k)} = \max_{0 \le j \le \min(k,M)} \{f_{k-j}\}, \quad k = 0, 1, 2, \dots, \tag{10}$$

and

$$d_k = -\frac{1}{\gamma_k^{BB1}} g_k. \tag{11}$$

The computational results show that the above global Barzilai–Borwein (GBB) algorithm is competitive with and sometimes superior to several well-known CG methods. However, as observed in [16], the GBB method is inefficient for solving ill-conditioned problems. This issue is likely to affect some drawbacks of the nonmonotone line search used in the GBB algorithm. More precisely, as mentioned in [22,23], the poor behaviour of GBB method may be due to the choice of the fixed value of the memory element M. So, the need for controlling the amount of nonmonotonicity is critical.

In recent years, several studies have focused on how to adaptively control the amount of the nonmonotonicity of line search methods. Amini et al. [24] proposed an adaptive nonmonotone line search with the following nonmonotone term

$$R_k = \eta_k f_{l(k)} + (1 - \eta_k) f_k, \tag{12}$$

where $\eta_k \in [\eta_{\min}, \eta_{\max}]$ for $\eta_{\min} \in [0, 1)$ and $\eta_{\max} \in [\eta_{\min}, 1]$. This nonmonotone term uses an adaptive determination of the convexity parameter η_k for controlling the degree of nonmonotonicity. Wan et al. [25] modified the nonmonotone line search suggested by Zhang and Hager [21] and developed an adaptive nonmonotone line search for solving smooth nonlinear equations. Furthermore, inspired from [24], Gao et al. [5] proposed an adaptive nonmonotone inexact Newton method for unconstrained optimization.

Although there is much research about nonmonotone line search methods, only a few of them focus on how to adaptively control the level of nonmonotonicity. Also, to the best of our knowledge, no study is made of the memory element M for better controlling the degree of nonmontonicity, from both the theoretical and computational points of view. It is the motivation behind the present study.

In this paper, we propose a globalization BB method equipped with an adaptive nonmonotone line search. To this end, we develop the nonmonotone line search of Grippo et al. [6] through an adaptive memory element. More precisely, considering that the first-order optimality condition is related to the morphology of a function, we propose a procedure to dynamically compute the memory element M at each iteration based on the first-order optimality condition. The global convergence to first-order stationary points and the convergence rate are established under some suitable conditions. Numerical experiments show the efficiency of the proposed nonmonotone line search technique.

The paper is organized as follows. Section 2 gives the motivations of our study and describes a new GBB-type algorithm. In Section 3, We prove the global convergence and the R-linear convergence rate of the proposed method under mild and reasonable assumptions. We report the results of our numerical experiments for the proposed algorithm in Section 4. Finally, we give some conclusions in Section 5.

Notation. Throughout this paper $g(x) = \nabla f(x)$ denotes the gradient of $f(x)$. We write $\|.\|$ and $\|.\|_\infty$ for the Euclidean and infinity norms of a vector, respectively. Furthermore, For all values, a subscript k means that this is the evaluation at x_k or the value in the kth iteration, e.g. f_k, g_k.

2. An adaptive GBB gradient method

We here give the motivation of our study and propose a procedure to dynamically compute the memory element M of the nonmonotone line search (9) and (10) at each iteration. By incorporating the adaptive nonmonotone line search in BB method, we then introduce a new GBB-type algorithm to solve unconstrained optimization problems.

The proposed global BB method [6] has several practical benefits. However, as pointed out in the introduction to this paper, the GBB method is inefficient for solving ill-conditioned problems. This issue is likely to affect some drawbacks of the nonmonotone line search used in the GBB algorithm. More precisely, as mentioned in [22,26], the poor behaviour of GBB method may be due to the choice of the fixed value of the memory element M. So, how efficiently to control the amount of nonmonotonicity through an adaptive memory element can be useful.

Plagianakos et al. proposed a novel and interesting work on how to use an adaptive memory element called PMV method for training of multilayer perceptrons in the field of neural networks [27]. They introduced a procedure to finding estimates of the memory element (also named nonmonotone learning horizon) M at each iteration based on the concept of the Lipschitz constant.

As mentioned in [27,28], it is reputed that the Lipschitz constant of the derivative of the objective function is nearly relevant to the morphology of a function in the sense that when a function has steep regions, the Lipschitz constant is large and when the function is flat, the Lipschitz constant is small. So, it can provide an adaptive way to control the magnitude of the memory element at each iteration. Since the Lipschitz constant is usually unknown a priori in practical computation and it plays a major role in the algorithm design, one needs to estimate it. In [27] a local estimation of the Lipschitz constant has been proposed as

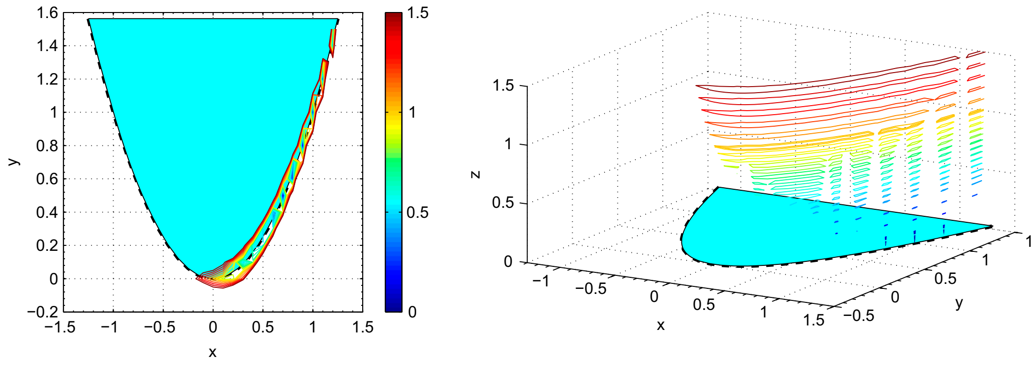
$$L_k = \frac{\|g_k - g_{k-1}\|}{\|x_k - x_{k-1}\|}, \tag{13}$$

which provides information related to the local shape of function. Accordingly, Plagianakos et al. [27] proposed the following procedure to adapt the value of the memory element at the each iteration:

$$M_k = \begin{cases} M_{k-1} + 1 & L_k < L_{k-1} < L_{k-2}, \\ M_{k-1} - 1 & L_k > L_{k-1} > L_{k-2}, \\ M_{k-1} & \text{otherwise}, \end{cases} \tag{14}$$

where L_k is the local estimation of the Lipschitz constant achieved through (13) at the kth iteration.

The proposed method in [27] has several practical benefits. However, there is a certain drawback associated with the use of the local estimation of the Lipschitz constant (13) in the structures of procedure (14). Indeed, the procedure (14) is not so suitable for controlling the degree of the



(a) 2D contour plot. (b) 3D contour plot.

Figure 1. 2D and 3D contour plots of the Rosenbrock function and plot of $y = x^2 + 0.005$.

nonmonotonicity in the presence of narrow curved valley, a common event in difficult non-linear problems. To support this claim, let us consider the following example.

Example: [Two-dimensional (2D) Rosenbrock function in [29]]

$$f(x, y) = 100(y - x^2)^2 + (1 - x^2)^2, \tag{15}$$

with the initial point $x_0 = [-1.2, 1]^T$.

This problem is probably the most famous of all test functions in mathematical optimization. It is also known as Rosenbrock valley or Rosenbrock banana function due to the shape of its contour lines (see Figure 1). The global minimum is at the point $\left[\,1,\,1\,\right]^T$ that lies inside a long, narrow valley. The floor of the valley follows approximately the parabola $y = x^2 + 0.005$ shown by dash line in Figure 1. The gradient of Rosenbrock function is

$$\nabla f(x, y) = \begin{bmatrix} 400x^3 + 2x - 2 - 400xy \\ -200x^2 + 200y \end{bmatrix}. \tag{16}$$

It follows from (16) that the gradient vector in the floor of the valley is approximately $\left[\,2x - 4,\,1\,\right]^T$. The local estimation of the Lipschitz constant achieved through (13) is thus nearly a fixed value 2 in the floor of the valley. Now, it is worth noting that when the sequence of iterations in the intermediate stage of the minimization process enter a valley such as the Rosenbrock valley, the third condition (14) is possibly satisfied because of the fixed value of the Lipschitz constant in the bottom of the valley. Therefore, due to the decreasing M_k before entering to the valley, the minimization process will be continued with a weaker nonmonotone strategy. It contrasts with the viewpoint mentioned in [6,30,31] which states that a stronger nonmonotone strategy can improve the convergence rate in case where the sequence of iterations follows the bottom of a curved narrow valley.

Similarly to the Lipschitz constant of the derivative of the objective function, the norm of the gradient is closely related to the morphology of a function. Unlike the Lipschitz constant, the norm of the gradient is not necessary to be approximated locally at each iteration. On the other hand, it is helpful to control adaptively the size of memory element in the case where the sequence of iterations follows the bottom of a curved narrow valley, especially when the value of the Lipschitz constant is approximately fixed in successive iterations.

Motivated by the above observations, a good way to avoid the use of the Lipschitz constant is to use the norm of the gradient instead. Hence, we propose an adaptive nonmonotone strategy based on the first-order optimality condition and incorporate it in the BB method for unconstrained optimization (1).

Now, we propose the following procedure to adapt the value of the memory element at the each iteration:

$$M_k = \begin{cases} M_{k-1}+1 & 10^{-1} \le \|g_k\|_\infty, \\ M_{k-1} & 10^{-3} \le \|g_k\|_\infty \le 10^{-1}, \\ M_{k-1}-1 & \text{otherwise.} \end{cases} \tag{17}$$

If the norm of the gradient is large, i.e. the first condition is satisfied in (17) just like the case that the sequence of iterations follows the bottom of a narrow valley, then the value of M_k should be increased to avoid creeping along the bottom of a narrow curved valley. On the other hand, when the second condition is satisfied in (17), i.e. the norm of the gradient is moderate, then the value of M_k is better to be unchanged. Finally, if the norm of the gradient is small, i.e. the third condition is satisfied in (17), it shows that the current iteration is in a flat area, hence, to further decrease the function value, the value of M_k should be decreased. Thus, an adaptive nonmonotone Armijo-type line search can be defined by

$$f(x_k + \alpha_k d_k) \le \hat{f}_{l(k)} + \delta \alpha_k g_k^T d_k, \tag{18}$$

where

$$\hat{f}_{l(k)} = \max_{0\le j\le \min(k, M_k)} \{f_{k-j}\}, \quad k = 0, 1, 2, \ldots, \tag{19}$$

where M_k is computed by (17).

We now present Algorithm 1 to describe the steps of our GBB-type method as follows.

Algorithm 1: Adaptive GBB-Type algorithm

```
Input: $x_0 \in \mathbb{R}^n$, constants $0 < \delta < 1, 0 < \rho < 1, \lambda_{\max} \ge \lambda_{\min} > 0, 0 < M_{\min} \le M_0 \le M_{\max}, \varepsilon > 0$;
Output: $x_b, f_b$;
1  begin
2      compute $f_0$ and $g_0$;
3      $\hat{f}_{l(0)} \leftarrow f(x_0), \lambda_0 \leftarrow 1, k \leftarrow 0$;
4      while $\|g_k\| \ge \varepsilon$ do
5          $d_k \leftarrow -\lambda_k g_k$;
6          $\alpha \leftarrow 1$;
7          $\hat{x}_k \leftarrow x_k + \alpha d_k$;
8          while $f(\hat{x}_k) > \hat{f}_{l(k)} + \delta \alpha g_k^T d_k$ do
9              $\alpha \leftarrow \rho\alpha$;
10             $\hat{x}_k \leftarrow x_k + \alpha d_k$;
11         end
12         $x_{k+1} \leftarrow \hat{x}_k$; $f_{k+1} \leftarrow f(\hat{x}_k)$;
13         compute stepsize $\gamma_k$ by (6);
14         if $\gamma_k < 0$ then
15             $\lambda_k \leftarrow \frac{1}{\lambda_{\max}}$;
16         else
17             $\lambda_{k+1} \leftarrow \min\{\lambda_{\max}, \max\{\lambda_{\min}, \frac{1}{\gamma_k}\}\}$;
18         end
19         compute $g_{k+1}$;
20         compute $M_{k+1}$ by (17) in $[M_{\min}, M_{\max}]$;
21         update $\hat{f}_{l(k+1)}$ by (19);
22         $k \leftarrow k + 1$;
23     end
24     $x_b \leftarrow x_k$; $f_b \leftarrow f_k$;
25 end
```

The sufficient descent condition is very important to the global convergence of the unconstrained optimization methods. Therefore, in Algorithm 1, the aim of lines 14–18 is to avoid uphill directions and keep the sequence $\{\lambda_k\}$ uniformly bounded. In fact, for all $k, \lambda_{\min} \le \lambda_k \le \lambda_{\max}$, this and

$d_k = -\lambda_k g_k$ ensure that there exist two positive numbers c_1 and c_2 such that the search direction d_k satisfies

$$g_k^T d_k \leq -c_1 \|g_k\|^2, \tag{20}$$

and

$$\|d_k\| \leq c_2 \|g_k\|. \tag{21}$$

It is clear that M_k must be positive. On other hand numerical results indicate that the measure of the upper bound of M_k is not critical. Hence, as seen in the line 19 of Algorithm 1, we determine upper and lower bounds on M_k to control its measure.

3. Convergence analysis

In this section, we study the global convergence and convergence rate of Algorithm 1. To verify the convergence analysis of the algorithm, the following assumptions are required:

(H1) The objective function $f(x)$ is continuously differentiable and bounded below on $\mathbb{R}^n$.

(H2) The gradient $g(x)$ of $f(x)$ is Lipschitz continuous over an open convex S that contains $L(x_0) = \{x \in \mathbb{R}^n | f(x) \leq f(x_0),\ x_0 \in \mathbb{R}^n\}$; i.e. there exists a positive constant L such that

$$\|g(x) - g(y)\| \leq L\|x - y\|, \quad \forall x, y \in S.$$

Theorem 3.1: *Suppose that* (H1) *and* (H2) *hold, and the sequence* $\{x_k\}$ *is generated by Algorithm* 1. *Then there exists* $\beta > 0$ *such that*

$$f_{k+1} \leq \hat{f}_{l(k)} - \beta \|g_k\|^2. \tag{22}$$

***Proof*:** Let $\beta_0 = \inf_{\forall k}\{\alpha_k\}$. If $\beta_0 > 0$, then it follows from (18) and (20) that

$$f_{k+1} \leq \hat{f}_{l(k)} - c_1\beta_0 \|g_k\|^2.$$

Setting $\beta = c_1\beta_0$, we can complete the proof by verifying $\beta_0 > 0$. On the contrary, suppose that $\beta_0 = 0$, hence there exists an infinite subset $K \subseteq \{0, 1, 2, \ldots,\}$ such that

$$\lim_{k \in K,\ k \to \infty} \alpha_k = 0. \tag{23}$$

Therefore, (23) implies that there is a $\acute{k}$ such that

$$\alpha_k/\rho \leq 1, \quad \forall k > \acute{k} \text{ and } k \in K.$$

Thus, setting $\alpha = \alpha_k \rho$, the inequality (18) is not satisfied, i.e. we have

$$f(x_k + \alpha d_k) > \hat{f}_{l(k)} + \delta \alpha g_k^T d_k. \tag{24}$$

It follows from (24) that

$$f_k - f(x_k + \alpha d_k) < -\delta \alpha g_k^T d_k.$$

Using the mean value theorem on the left-hand side of the above inequality, there exists $\theta_k \in [0, 1]$ such that

$$-\alpha g(x_k + \theta_k \alpha d_k)^T d_k < -\delta \alpha g_k^T d_k.$$

Thus

$$g(x_k + \theta_k \alpha d_k)^T d_k > \delta g_k^T d_k. \tag{25}$$

We conclude from (H2), Cauchy–Schwartz inequality, (20) and (25) that

$$\begin{aligned} L\alpha\|d_k\|^2 &\geq \|g(x_k+\alpha\theta_k d_k)-g_k\|\|d_k\| \\ &\geq (g(x_k+\alpha\theta_k d_k)-g_k)^T d_k \\ &\geq -(1-\delta)g_k^T d_k \\ &\geq c_1(1-\delta)\|g_k\|^2. \end{aligned}$$

We can obtain from (21) that

$$\alpha_k \geq \frac{c_1(1-\delta)}{\rho L}\frac{\|g_k\|^2}{\|d_k\|^2} \geq \frac{c_1(1-\delta)}{c_2^2\rho L} > 0, \quad k > \acute{k} \text{ and } k \in K,$$

which contradicts (23). So, $\beta_0 > 0$ and the proof is complete. □

Before stating the following lemma, for the convenience of the reader we define $\widehat{m}(k) = \min\{k, M_k\}$ and $N \equiv M_{\max}$.

Lemma 3.2: *If the conditions of Theorem 3.1 hold, then*

$$\max_{1\leq j\leq N} f(x_{Nl+j}) \leq \max_{1\leq j\leq N} f(x_{N(l-1)+j}) - \beta \min_{1\leq j\leq N}\|g_{Nl+j-1}\|^2, \tag{26}$$

and

$$\sum_{l=1}^{\infty}\min_{1\leq i\leq N}\|g_{Nl+i-1}\|^2 < +\infty. \tag{27}$$

***Proof*:** To prove (26), it is sufficient to show the following inequality holds for $j = 1, 2, \ldots, N$,

$$f(x_{Nl+j}) \leq \max_{1\leq j\leq N} f(x_{N(l-1)+j}) - \beta\|g_{Nl+j-1}\|^2. \tag{28}$$

Theorem 3.1 implies

$$f(x_{Nl+j}) \leq \max_{1\leq j\leq \widehat{m}\{Nl\}} f(x_{Nl-i}) - \beta\|g_{Nl}\|^2. \tag{29}$$

It follows from (29) and $0 \leq \widehat{m}\{Nl\} \leq N$ that (28) holds for $j = 1$. Let (28) hold for any $1 \leq j \leq N-1$. It follows from the induction hypothesis and the nonnegative term $\|g_{Nl+j-1}\|^2$ on the right of (28) that

$$\max_{1\leq i\leq j} f(x_{Nl+i}) \leq \max_{1\leq i\leq N} f(x_{N(l-1)+i}). \tag{30}$$

It concludes from the induction hypothesis, (18), (30) and $0 \leq \widehat{m}\{Nl\} \leq N$ that

$$\begin{aligned} f(x_{Nl+j+1}) &\leq \max_{1\leq i\leq \widehat{m}\{Nl+j\}} f(x_{N(l-1)+j+1}) - \beta\|g_{Nl+j}\|^2 \\ &\leq \max\left\{\max_{1\leq i\leq N} f(x_{N(l-1)+i}), \max_{1\leq i\leq j} f(x_{Nl+i})\right\} - \beta\|g_{Nl+j}\|^2 \\ &\leq \max_{1\leq i\leq N} f(x_{N(l-1)+i}) - \beta\|g_{Nl+j}\|^2. \end{aligned}$$

Thus (28) is satisfied for $j+1$, and by induction, (28) holds for $1 \leq j \leq N$. Hence (26) holds.

Since $f(x)$ is bounded from below by (H1), it follows that

$$\max_{1\leq i\leq N} f(x_{Nl+i}) > -\infty.$$

It follows by summing (26) over l that

$$\sum_{l=1}^{\infty} \min_{1\le i\le N} \|g_{Nl+i-1}\|^2 < +\infty.$$

Therefore (27) holds and the proof is complete. □

Theorem 3.3: *If the conditions of Theorem 3.1 hold, then*

$$\lim_{k\to\infty} \|g_k\| = 0. \tag{31}$$

Proof: We begin by proving the fact that there exists a constant c_3 such that

$$\|g_{k+1}\| \le c_3\|g_k\|. \tag{32}$$

We obtain from (H2) and (21) that

$$\begin{aligned}\|g_{k+1}\| &\le \|g_{k+1} - g_k + g_k\| \\ &\le \|g_{k+1} - g_k\| + \|g_k\| \\ &\le L\alpha\|d_k\| + \|g_k\| \\ &\le (1 + Lc_2)\|g_k\|.\end{aligned}$$

Setting $c_3 = 1 + Lc_2$, we obtain

$$\|g_{k+1}\| \le c_3\|g_k\|. \tag{33}$$

We now set

$$\phi(l) = \arg\min_{0\le i\le N-1} \|g_{Nl+i}\|. \tag{34}$$

Lemma 3.2 shows that

$$\lim_{l\to\infty} \|g_{Nl+\phi(l)}\| = 0. \tag{35}$$

From (33), we deduce that

$$\|g_{N(l+1)+i}\| \le c_3^{2N}\|g_{Nl+\phi(l)}\|, \quad i = 0, 1, \ldots, N-1. \tag{36}$$

Therefore, it follows from (35) and (36) that (31) holds. □

We conclude this section by proving the R-linear convergence rate of the sequence generated by Algorithm 1. Dai [30] established the R-linear convergence of the nonmonotone line search of Grippo et al. [6] for strongly convex functions. Similar to [30], the R-linear convergence of Algorithm 1 for strongly convex functions can be established. To this end, recall that the objective function f is a strongly convex function if there exists a scalar ω such that

$$f(x) \ge f(y) + \nabla f(y)^T(x-y) + \frac{1}{2\omega}\|x-y\|^2, \tag{37}$$

for all $x, y \in \mathbb{R}^n$. In order to establish the R-linear convergence rate, we need the following lemma.

Lemma 3.4: *Suppose that* (H1) *and* (H2) *hold, $f(x)$ is a strongly convex function and the sequence $\{x_k\}$ is generated by Algorithm* 1. *Then there exist constants $c_4 > 0$ and $c_5 \in (0,1)$ such that*

$$f(x_k) - f(x_*) \le c_4c_5^k\left[f(x_1) - f(x_*)\right], \tag{38}$$

for all $k \in \mathbb{N}$.

***Proof*:** Lemma 3.2 and (33) provide all necessary assumptions for Theorem 3.1 of [30] to hold. Therefore, the conclusion can be proved in a similar way. For more details we refer the reader to [30]. □

Theorem 3.5: *If the conditions of Lemma 3.4 hold, then the sequence $\{x_k\}$ converges to the stationary point x_* at least R-linearly.*

***Proof*:** Recall that the sequence $\{x_k\}$ converges to x_* R-linearly if there exists a sequence of nonnegative scalars $\{\nu_k\}$ such that, for all $k \in \mathbb{N}$,

$$\|x_k - x_*\| \leq \nu_k, \tag{39}$$

where the sequence $\{\nu_k\}$ converges Q-linearly to zero. We first introduce a sequence $\{\nu_k\}$, then prove its Q-linear convergence. Theorem 3.3 together with substituting $y = x_*$ and $x = x_k$ in (37) imply that

$$\|x_k - x_*\|^2 \leq 2\omega\left(f(x_k) - f(x_*)\right) \leq \left[2\omega c_4\left(f(x_1) - f(x_*)\right)\right] c_5{}^k = rc_5{}^k, \tag{40}$$

where $r = \left[2\omega c_4\left(f_1 - f_*\right)\right]$. By setting $\nu_k = rc_5{}^k$, we get that $\nu^* = 0$. We also have

$$\lim_{k\to\infty} \frac{\nu_{k+1} - \nu_*}{\nu_k - \nu_*} = c_5 < 1. \tag{41}$$

Therefore, the sequence $\{x_k\}$ converges to x_* at least R-linearly. □

4. Numerical experiments

In this section, some computational performances of the proposed algorithm are reported to illustrate the numerical efficiency of the proposed nonmonotone line search (HNS). To this end, we firstly compare our algorithm with the other algorithms associated with different line search rules. For this purpose, we replace the HNS rule used in Algorithm 1 by the standard Armijo line search in [28] (MA), the nonmonotone line search of Grippo et al. [6] (GLL) and the adaptive nonmonotone line search of Plagianakos et al. [27] (PMV). In the second comparison, we then compare the performance of our method with two CG methods equipped with modified Armijo line search rules. The first one is a modified Hestenes–Stiefel (MHS) CG method [32] which has a similar form as the CG-DESCENT method proposed by Hager and Zhang [33] and the second is the MPRPA method which is a modified Polak–Ribiere–Polyak (PRP) CG method proposed in [12].

Our experiments are performed in double precision arithmetic format in MATLAB 7.14.0.739 (R2012a) programming environment on a computer with 2.6 gigahertz of CPU, 1 gigabytes of RAM and Centos 6.2 server Linux operation system. They are performed on a set of 147 unconstrained optimization test problems of the CUTEr collection [34], with default dimensions. The test problems data have been clarified in Table 1.

We used the following stopping criterion:

$$\|g_k\| \leq 10^{-5}. \tag{42}$$

The process is also stopped if the number of gradient evaluations exceeds 20,000, or the number of function evaluations reaches 50,000. In order to have a proper comparison, all algorithms were implemented with the parameters $\lambda_{\max} = 10^{30}$, $\lambda_{\min} = 10^{-30}$, $\delta = 10^{-4}$ as suggested in [15]. We used the value $M = 10$ for the GLL algorithm. Like the GLL algorithm, the PMV and HNS algorithms take advantage of the initial value $M_0 = 10$. Numerical results indicate the measure of the upper bound of M_k is not critical, on other hand M_k has to be positive. Hence, the boundaries of M_k is set to $3 \leq M_k \leq 15$, for the PMV and HNS algorithms. We consider the number of gradient and function evaluations and CPU time in seconds (T) to compare the algorithms.

Table 1. Specifications of the test problems.

Problem	Dim.	Problem	Dim.	Problem	Dime.
HIMMELBH	2	PALMER3C	8	SROSENBR	5000
ROSENBR	2	STRATEC	10	ARWHEAD	5000
HILBERTA	2	HILBERTB	10	BDQRTIC	5000
HIMMELBB	2	OSBORNEB	11	BROYDN7D	5000
MEXHAT	2	WATSON	12	BRYBND	5000
S308	2	PARKCH	15	CRAGGLVY	5000
SISSER	2	ARGLINB	50	DQRTIC	5000
SNAIL	2	ARGLINC	50	ENGVAL1	5000
ZANGWIL2	2	NCB20B	50	FLETBV3M	5000
AKIVA	2	TOINTGOR	50	POWELLSG	5000
BEALE	2	TOINTPSP	50	SCHMVETT	5000
BRKMCC	2	TOINTQOR	50	SPARSINE	5000
BROWNAL	2	VAREIGVL	50	SSBRYBND	5000
BROWNBS	2	DIXON3DQ	100	TESTQUAD	5000
CUBE	2	FREUROTH	100	TOINTGSS	5000
DENSCHNA	2	MANCINO	100	DQRTIC	5000
DENSCHNB	2	MSQRTBLS	100	FMINSRF2	5625
DENSCHNC	2	NONCVXU2	100	FMINSURF	5625
DENSCHNF	2	PENALTY2	100	DIXMAANA	9000
EXPFIT	2	SENSORS	100	DIXMAANB	9000
HAIRY	2	ARGLINA	200	DIXMAANC	9000
HIELOW	2	PENALTY3	200	DIXMAAND	9000
HIMMELBG	2	VARDIM	200	DIXMAANE	9000
GULF	3	PENALTY1	1000	DIXMAANG	9000
HELIX	3	EG2	1000	DIXMAANH	9000
YFITU	3	EXTROSNB	1000	DIXMAANI	9000
HATFLDE	3	FLETCBV2	1000	DIXMAANIJ	9000
HATFLDFL	3	FLETCBV3	1000	DIXMAANK	9000
BARD	3	FLETCHCR	1000	DIXMAANL	9000
BOX2	3	GENHUMPS	1000	DIXMAANM	9000
BOX3	3	INDEF	1000	DIXMAANN	9000
DENSCHNE	3	NCB20	1000	DIXMAANO	9000
ENGVAL2	3	SINQUAD	1000	DIXMAANP	9000
GROWTHLS	3	SSCOSINE	1000	LIARWHD	10000
KOWOSB	4	MODBEALE	2000	POWER	10000
MSQRTALS	4	BOXPOWER	2000	QUARTC	10000
ALLINITU	4	EDENSCH	2000	CURLY10	10000
BROWNDEN	4	EIGENALS	2550	CURLY20	10000
OSBORNEA	5	EIGENBLS	2550	CURLY30	10000
GENROSE	5	JIMACK	3549	FLETCHBV	10000
PALMER5C	6	CHAINWOO	4000	INDEFM	10000
BIGGS3	6	WOOD	4000	NONDIA	10000
BIGGS5	6	MOREBV	5000	NONDQUAR	10000
PALMER4C	8	TQUARTIC	5000	OSCIGRAD	10000
VIBRBEAM	8	TRIDIA	5000	SCURLY10	10000
PALMER2C	8	DQDRTIC	5000	SCURLY20	10000
AIRCRFTB	8	POWELLSG	5000	SCURLY30	10000
PALMER1C	8	SPMSRTLS	5000	SPARSQUR	10000

Here, we analyse the performance data using the profiles of Dolan and Moré [35]. In fact, for each method, the fraction P of problems for which the method is within a factor τ of the best time is plotted. The left side of the figure gives the percentage of the test problems for which a method is the fastest, and the right side gives the percentage of the test problems that are successfully solved by each of the methods. In essence, the right side is as a measure for the robustness of an algorithm. The top curve is the method that solved the most problems with a performance that is a coefficient τ of the best performance.

Figures 2–4 give the performance profiles of the algorithms based on the number of gradient evaluations, the number of function evaluations and CPU time, respectively. Also, Table 2 shows the

Table 2. Percentage of the test problems each method was fastest.

Algorithm	Gradient evaluations (%)	Function evaluations (%)	CPU time (%)
The first comparison			
MA	50	41.6	22.1
GLL	53.7	56.5	27.8
PMV	50	43.5	29
HNS	62	67.6	50.9
The second comparison			
MHS	58.3	57.4	40.4
MPRPA	49	50	33.7
HNS	64.8	65.7	43.3

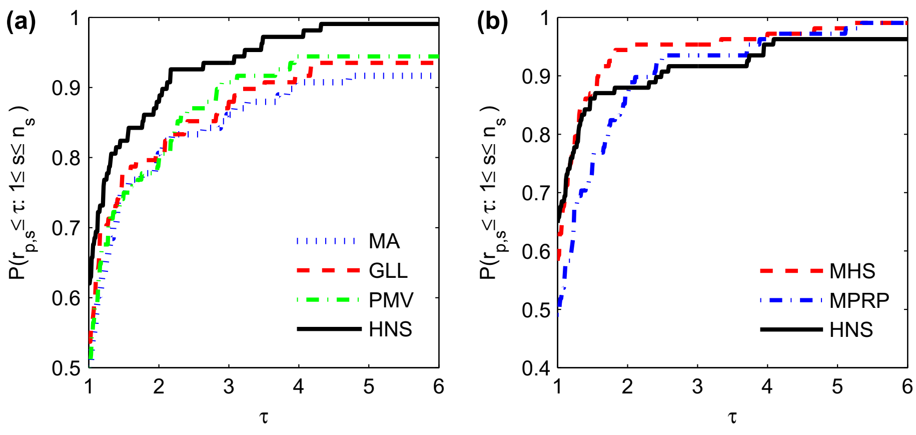


Figure 2. Performance profiles for the number of gradient evaluations.

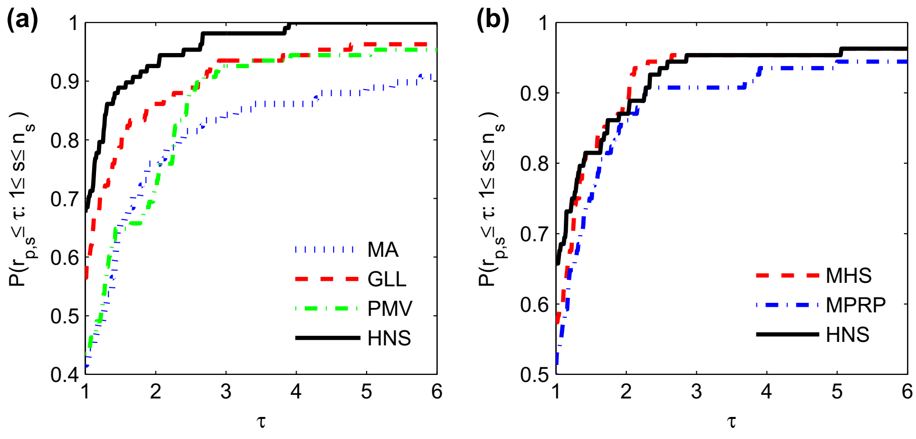


Figure 3. Performance profiles for the number of function evaluations.

percentage of the test problems for which a method has the best performance. In each comparison, the column total in Table 2 exceeds 100 due to ties for some test problems.

We now analyse the performance of algorithms as follows. In Figure 2, the subfigure (a) shows that HNS attains most wins with about 62% score and GLL, PMV and MA are almost in the competition.

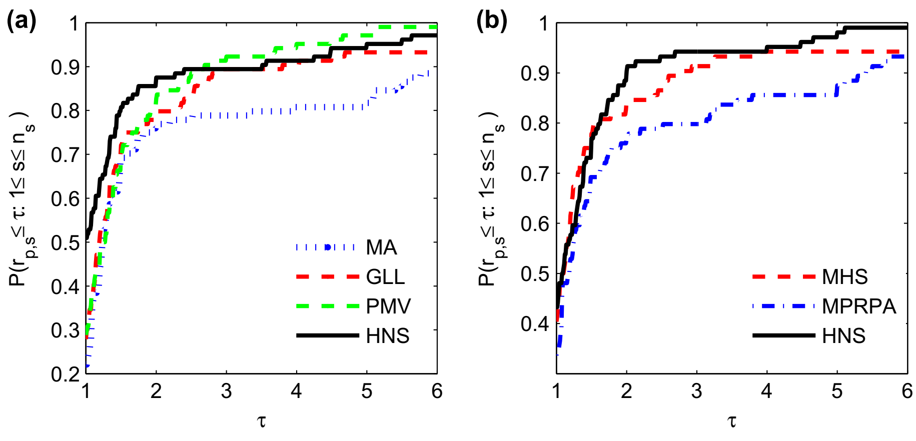


Figure 4. Performance profiles for CPU time.

Meanwhile, in the sense of the ability to complete a run successfully, HNS is the best among considered algorithms because it grows up faster than the others. The horizontal axis in the figure stops at $\tau = 6$ since the plots are essentially flat for larger values of τ. From the subfigure (b), we can see that HNS attains most wins with about 64.8% score. It is competitive with MHS and performs better than MPRPA.

In Figure 3, the subfigure (a) clearly indicates that HNS outperforms the other algorithms with respect to the number of function evaluations. In particular, HNS attains the most wins with about 67.6% score. Moreover, the slope of its curve shows that HNS is the best algorithm in the sense of the ability to complete a run successfully. From the subfigure (b), we can see that HNS attains most wins with about 65.7% score. It is competitive with MHS and slightly better than MPRPA.

As can be seen in Figure 4, the subfigure (a) shows that HNS attains about 50.9% of wins and followed by GLL and PMV. Also, the subfigure (b) shows that HNS obtains about 43.3% of wins and is competitive with MHS and slightly better than MPRPA.

On the whole, Figures 2–4 and Table 2 show that HNS has a promising performance for solving unconstrained optimization problems of the form (1).

5. Conclusions

Based on a study of the morphology of the objective function just like PMV in [27], an adaptive nonmonotone line search by developing the nonmonotone line search of Grippo et al. [6] has been suggested. Unlike what was proposed by Plagianakos et al. [27], our procedure that adapts the value of the memory element at the each iteration has been based on the norm of gradient to overcome drawbacks of PMV algorithm. By utilizing the proposed adaptive nonmonotone line search, we have developed an adaptive GBB algorithm for unconstrained optimization problems.

The global convergence and the rate of convergence of the proposed method have been discussed. Numerical comparisons have been made in two parts on a set of 147 unconstrained optimization test problems of the CUTEr collection [34]. The first comparison is among the implementations of the proposed method and MA, GLL, PMV and the second is among the proposed method and two CG methods MHS and MPRPA. The results showed the efficiency of the proposed method in the sense of the performance profile introduced by Dolan and Moré [35].

Disclosure statement

No potential conflict of interest was reported by the authors.

References

[1] Birgin EG, Chambouleyron I, Martinez JM. Estimation of the optical constants and the thickness of thin films using unconstrained optimization. J Comput Phys. 1999;151:862–880.

[2] Ng CK, Liao LZ, Li D. A globally convergent and efficient method for unconstrained discrete-time optimal control. J Global Optim. 2002;22:401–421.

[3] Magoulas G, Varhatis MN. Adaptive algorithms for neural network supervised learning: a deterministic optimization approach. Int J Bifurcation Chaos. 2006;16(7):1929–1950.

[4] Peng CC, Magoulas G. Advanced adaptive nonmonotone conjugate gradient training algorithms for recurrent neural networks. Int J Artif Intell Tools. 2008;17(5):963–984.

[5] Gao H, Zhang HB, Li ZB, et al. A nonmonotone inexact Newton method for unconstrained optimization. Optim Lett. 2015. DOI:10.1007/s11590-015-0976-2.

[6] Grippo L, Lampariello F, Lucidi S. A nonmonotone line search technique for Newton's method. SIAM J Numer Anal. 1986;23:707–716.

[7] Hestenes MR, Stiefel EL. Methods of conjugate gradients for solving linear systems. J Res Nat Bureau Stand. 1952;1952:409–436.

[8] Yuan G, Meng Z, Li Y. A modified Hestenes and Stiefel conjugate gradient algorithm for large-scale nonsmooth minimizations and nonlinear equations. J Optim Theory Appl. 2016;168:129–152.

[9] Yuan G, Zhang M. A three-terms Polak–Ribière–Polyak conjugate gradient algorithm for large-scale nonlinear equations. J Comput Appl Math. 2015;286:186–195.

[10] Yuan G, Wei Z, Li G. A modified Polak–Ribière–Polyak conjugate gradient algorithm for nonsmooth convex programs. J Comput Appl Math. 2014;255:86–96.

[11] Yuan G, Wei Z. The Barzilai and Borwein gradient method with nonmonotone line search for nonsmooth convex optimization problems. Math Model Anal. 2012;17(2):203–216.

[12] Zhang L, Zhou W, Li DH. A descent modified Polak–Ribière–Polyak conjugate gradient method and its global convergence. IMA J Numer Anal. 2006;26:629–640.

[13] Akaike H. On a successive transformation of probability distribution and its application to the analysis of the optimum gradient method. Ann Inst Stat Math. 1959;11:1–17.

[14] Barzilai B, Borwein JM. Methods of conjugate gradients for solving linear systems. IMA J Numer Anal. 1988;8: 141–148.

[15] Xiao Y, Wang Q, Wang D. Notes on the Dai–Yuan–Yuan modified spectral gradient method. J Comput Appl Math. 2010;234:2986–2992.

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

[17] Dai YH, Liao LZ. R-linear convergence of the Barzilai and Borwein gradient method. IMA J Numer Anal. 2002;22:1–10.

[18] Ahookhosh M, Amini K, Bahrami S. A class of nonmonotone Armijo-type line search method for unconstrained optimization. Optim: J Math Program Oper Res. 2012;61(4):387–404.

[19] Huang S, Wan Z, Chen Z. A new nonmonotone line search technique for unconstrained optimization. Numer Algorithms. 2015;68:671–689.

[20] Tang J, Dong L, Zhou J, et al. A smoothing-type algorithm for the second-order cone complementarity problem with a new nonmonotone line search. Optim: J Math Program Oper Res. 2015;64(9):1935–1955.

[21] Zhang HC, Hager WW. A nonmonotone line search technique and its application to unconstrained optimization. SIAM J Optim. 2004;14(4):1043–1056.

[22] Grippo L, Sciandrone M. Nonmonotone globalization techniques for the Barzilai–Borwein gradient method. Comput Optim Appl. 2002;23:143–169.

[23] Fletcher R. Low storage methods for unconstrained optimization. In: Allgower EL, Georg K, editors. Computational solution of nonlinear systems of equations. Vol. 26, Lectures in Applied Mathematics (AMS); 1990;26:165–179.

[24] Amini K, Ahookhosh M, Nosratipour H. An inexact line search approach using modified nonmonotone strategy for unconstrained optimization. Numer Algorithms. 2014;60(1):49–78.

[25] Wan Z, Chen Y, Huang S, et al. A modified nonmonotone BFGS algorithm for solving smooth nonlinear equations. Optim Lett. 2004;8(6):11845–1860.

[26] Qi L, Teo K, Yang X. Optimization and control with applications. New York (NY): Springer; 2005.

[27] Plagianakos VP, Magoulas G, Vrahatis MN. Deterministic nonmonotone strategies for effective training of multilayer perceptrons. IEEE Trans Neural Networks. 2002;13(6):1268–1284.

[28] Armijo L. Minimization of functions having Lipschitz-continuous first partial derivatives. Pac J Math. 1966;16: 1–3.

[29] Rosenbrock HH. An automatic method for finding the greatest or least value of a function. Comput J. 1960;3: 175–184.

[30] Dai H. On the nonmonotone line search. J Optim Theory Appl. 2002;112(2):315–330.

[31] Toint PL. An assessment of non-monotone line search techniques for unconstrained optimization. SIAM J Sci Comput. 1996;17:725–739.
[32] Dai Z, Wen F. Global convergence of a modified Hestenes–Stiefel nonlinear conjugate gradient method with Armijo line search. Numer Algorithms. 2012;59:79–93.
[33] Hager WW, Zhang H. A new conjugate gradient method with guaranteed descent and an efficient line search. SIAM J Optim. 2005;16:170–192.
[34] Gould NIM, Orban D, Toint PL. CUTEr: a constrained and unconstrained testing environment, revisited. ACM Trans Math Softw. 2003;29:373–394.
[35] Dolan E, Moré JJ. Benchmarking optimization software with performance profiles. Math Program. 2002;91: 201–213.