Applied Mathematical Modelling 36 (2012) 478–487

Contents lists available at ScienceDirect

Applied Mathematical Modelling

journal homepage: www.elsevier.com/locate/apm

A nonmonotone trust-region line search method for large-scale unconstrained optimization

Masoud Ahookhosh [a], Keyvan Amini [a,*], Mohammad Reza Peyghami [b]

[a] Department of Mathematics, Faculty of Science, Razi University, Kermanshah, Iran
[b] Department of Mathematics, K.N. Toosi University of Technology, P.O. Box 16315-1618, Tehran, Iran

ARTICLE INFO

Article history:
Received 12 March 2011
Received in revised form 13 June 2011
Accepted 1 July 2011
Available online 19 July 2011

Keywords:
Unconstrained optimization
Trust-region method
Armijo-type line search
Nonmonotone technique

ABSTRACT

We consider an efficient trust-region framework which employs a new nonmonotone line search technique for unconstrained optimization problems. Unlike the traditional nonmonotone trust-region method, our proposed algorithm avoids resolving the subproblem whenever a trial step is rejected. Instead, it performs a nonmonotone Armijo-type line search in direction of the rejected trial step to construct a new point. Theoretical analysis indicates that the new approach preserves the global convergence to the first-order critical points under classical assumptions. Moreover, superlinear and quadratic convergence are established under suitable conditions. Numerical experiments show the efficiency and effectiveness of the proposed approach for solving unconstrained optimization problems.

1. Introduction

Trust-region and line search methods are two prominent classes of iterative methods to solve the following unconstrained optimization problem

$$\text{minimize } f(x), \quad \text{subject to } x \in \mathbf{R}^n, \tag{1}$$

where $f : \mathbf{R}^n \to \mathbf{R}$ is a twice continuously differentiable function. For a given x_k, line search methods refer to a procedure that computes a steplength α_k in the specific direction d_k and drives a new point as $x_{k+1} = x_k + \alpha_k d_k$. On the other hand, trust-region methods compute a trial step d_k by solving the following quadratic subproblem

$$\text{minimize } m_k(d) = f_k + g_k^T d + \frac{1}{2} d^T B_k d, \text{ subject to } d \in \mathbf{R}^n \text{ and } \|d\| \leqslant \delta_k, \tag{2}$$

where $\|\cdot\|$ denotes the Euclidean norm, $f_k = f(x_k)$, $g_k = \nabla f(x_k)$, B_k is the exact Hessian $G_k = \nabla^2 f(x_k)$, or a symmetric approximation for it, and δ_k is the trust-region radius. Using the ratio

$$\rho_k = \frac{f(x_k) - f(x_k + d_k)}{m_k(0) - m_k(d_k)}, \tag{3}$$

traditional trust-region methods evaluate an agreement between the model and the objective function. The trial step d_k is accepted whenever ρ_k is greater than a positive constant μ_1. This leads us to the new point $x_{k+1} = x_k + d_k$, and the trust-region radius is updated. Otherwise, the trust-region radius must be diminished and the subproblem (2) must be solved again.

* Corresponding author.
E-mail addresses: ahoo.math@gmail.com (M. Ahookhosh), kamini@razi.ac.ir (K. Amini), peyghami@kntu.ac.ir (M.R. Peyghami).

0307-904X/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.apm.2011.07.021

It is well-known that the traditional trust-region frameworks consist of some drawbacks. First of all, solving the trust-region subproblems may lead us to solve one or more linear systems or may impose an iterative process with high computational cost to the method. Considering this fact, traditional trust-region methods are quiet adequate for small or medium scale problems. A way to overcome this drawback is to use a line search strategy with low computational cost through the rejected trial step. Another drawback will be emerged when the elements of the trial step are excessively large. In this case, reducing the trust-region radius gives a new trial step close to the rejected one, and therefore the possibility of accepting the new trial step is reduced significantly. In this situation, line search methods behave in the similar way as traditional trust-region methods, but its computational cost is much lower than the traditional trust-region methods. Inspired by these facts, some modified versions of trust-region approaches have been proposed in [1–3]. The primary aim of these methods is to prevent from resolving the trust-region subproblem when the current trial step is rejected. Indeed, some forms of backtracking line search along directions of the rejected trial steps have been exploited. Theoretical and numerical results of these modified trust-region methods are also interesting.

One of the most interesting techniques for improving the iterative algorithms in optimization is nonmonotone techniques. Perhaps, the first nonmonotone optimization technique is the "watchdog technique", introduced by Chamberlain et al. in [4] in 1982, in order to overcome the Maratos effect in certain constrained optimization methods. Based on this idea, Grippo et al. in [5,6] proposed a nonmonotone line search technique for Newton's method. This algorithm accepts the steplength α_k whether

$$f(x_k + \alpha_k d_k) \leqslant f_{l(k)} + \beta \alpha_k \nabla f(x_k)^T d_k, \tag{4}$$

where $\beta \in (0, 1/2)$,

$$f_{l(k)} = \max_{0 \leqslant j \leqslant m(k)} \{f_{k-j}\} \quad k = 0, 1, 2, \ldots, \tag{5}$$

$m(0) = 0$, $0 \leqslant m(k) \leqslant \min\{m(k-1) + 1, N\}$ for all $k \geqslant 1$ and $N \geqslant 0$. Deng et al. in [7] made some changes in the ratio (3) which assesses the agreement between the quadratic model and the objective function in trust-region methods. This idea was developed further by Zhou and Xiao [8,9], Xiao and Chu [10] and Toint [11,12]. The most common nonmonotone ratio is defined as follows:

$$\tilde{\rho}_k = \frac{f_{l(k)} - f(x_k + d_k)}{m_k(0) - m_k(d_k)}. \tag{6}$$

The nonmonotone methods are distinguished by the fact that they do not enforce strict monotonicity to the objective function values at successive iterations. Some researchers showed that utilizing nonmonotone techniques may improve both the possibility of finding the global optimum and the rate of convergence [5,13]. The numerical results of nonmonotone methods are very favorable, overall, especially when they are applied to very nonlinear problems with a narrow curved valley.

Although the nonmonotone technique (5) has many advantages, it suffers from some drawbacks, see [2,14,13]. Firstly, although an iterative method may generates R-linearly convergent iterations for strongly convex function, the iterations may not satisfy the condition (4) for k sufficiently large, for any fixed bound N on the memory. Secondly, a good function value generated in any iteration is essentially discarded due to the maximum term in (4). Finally, in some cases, the numerical performance is highly depended on the choice of N. There are some proposals to overcome these disadvantages [2,13] that lead us to introduce a new formula instead of $f_{l(k)}$ in (6). As a result of above discussion, some modified versions of nonmonotone trust-region ratio have been proposed [2,14,12]. Theoretical and computational results show that these modifications can improve the efficiency of the nonmonotone trust-region methods [2,14].

In this paper, we introduce a new nonmonotone strategy and exploit it into a modified trust-region framework. The proposed method first solve the subproblem (2) in order to compute the trial step d_k, and then set $x_{k+1} = x_k + d_k$ whenever the trial step d_k is accepted by the new nonmonotone trust-region technique. Otherwise, the method takes the advantage of line search in the direction of rejected trial step d_k, instead of resolving the trust-region subproblem. The analysis of the new approach shows that it inherits both stability of trust-region methods and low computational cost of line search methods. We also investigate the global convergence property of the proposed algorithm and provide the superlinear and quadratic convergence rate. To show the efficiency of the proposed method in practice, some numerical results are reported.

This paper is organized as follows. In Section 2, we describe our new algorithm and give its properties. Section 3 is devoted to investigate the global, superlinear and quadratic convergence properties of the proposed algorithm. Numerical results are provided in Section 4 to show that the new proposed method is well promising for unconstrained optimization problems. Finally, some concluding remarks are given in Section 5.

2. New trust-region line search algorithm

In this section, we first introduce a new nonmonotone strategy and then describe our trust-region line search algorithm. We also establish some properties of our proposed algorithm.

It has been proved that the best convergence results are obtained by stronger nonmonotone strategy when iterations are far from the optimum, and by weaker nonmonotone strategy when iterations are close to it [13]. We believe that the nonmonotone strategy (5) does not sufficiently employ the current value of the objective function f. Hence, it seems that this

nonmonotone strategy has well performance far from the optimum; in contrast, it does not have an adequate behavior near the optimum. On the other hand, taking maximum is one of the most prominent information among the recent successful iterations, and we do not want to ignore this striking fact. Therefore, in order to avoid this disadvantage and decrease effects of the traditional nonmonotone strategy (5), as mentioned in Section 1, we define the new nonmonotone term by

$$R_k = \eta_k f_{l(k)} + (1-\eta_k) f_k, \tag{7}$$

where $\eta_k \in [\eta_{\min}, \eta_{\max}]$ for $\eta_{\min} \in [0,1)$ and $\eta_{\max} \in [\eta_{\min}, 1]$. It is obvious that when η_k is close to 1, one can obtain a strong nonmonotone strategy and can get a weaker nonmonotone strategy when η_k is close to 0. Hence, by choosing η_k adaptively, one can increase the affect of $f_{l(k)}$ far from the optimum and decrease it near to the optimum.

Let us describe one step of our algorithm. We first solve the subproblem (2) in order to compute the trial step d_k and then define the following ratio

$$\hat{\rho}_k = \frac{R_k - f(x_k + d_k)}{m_k(0) - m_k(d_k)}. \tag{8}$$

If $\hat{\rho}_k \geqslant \mu_1$, then we accept the trial step and set $x_{k+1} = x_k + d_k$. Otherwise, we determine the steplength $\alpha_k \in \{s, \rho s, \rho^2 s, \ldots\}$ by subsequent Armijo-type line search

$$f(x_k + \alpha_k d_k) \leqslant R_k + \beta \alpha_k g_k^T d_k, \tag{9}$$

where s is a positive constant, $\rho \in (0,1)$ and $\beta \in (0, 1/2)$. In this case, we set $x_{k+1} = x_k + \alpha_k d_k$. Now, we can outline our new nonmonotone trust-region line search algorithm as follows:

Algorithm 1. New Nonmonotone Trust-Region Line Search Agorithm (NMTLN)

1. *Initialization*. An initial point $x_0 \in \mathbf{R}^n$, a symmetric matrix $B_0 \in \mathbf{R}^{n\times n}$ and initial trust-region radius $\delta_0 > 0$ are given. The constants $0 < \mu_1 \leqslant \mu_2 < 1$, $0 < \rho < 1$, $0 < \beta < 1/2$, $0 \leqslant \eta_{\min} < 1$ and $\eta_{\min} \leqslant \eta_{\max} < 1, N \geqslant 0, \bar{\delta} > 0, c > 0$ and $\epsilon > 0$ are also given. Compute $f(x_0)$ and set $k = 0$.
2. *Termination criteria*. Compute $g(x_k)$. If $\|g(x_k)\| \leqslant \epsilon$, stop.
3. *Trial step calculation*. Solve the subproblem (2) to determine a trial step d_k that satisfies $\|d_k\| \leqslant \delta_k$.
4. *Acceptance of the trial step*. Compute $n(k)$, $f_{l(k)}$, R_k and $\hat{\rho}_k$. If $\hat{\rho}_k \geqslant \mu_1$, set $x_{k+1} = x_k + d_k$ and go to Step 5. Otherwise, find the steplength α_k satisfying in (9), and set $x_{k+1} = x_k + \alpha_k d_k$. Update the trust-region radius by $\delta_{k+1} = \min\{c\|x_{k+1} - x_k\|, \delta_k\}$ and go to Step 6.
5. *Trust-region radius update*. Set

$$\delta_{k+1} \in \begin{cases} \delta_k, & \text{if } \mu_1 \leqslant \hat{\rho}_k < \mu_2, \\ [\delta_k, \bar{\delta}), & \text{if } \hat{\rho}_k \geqslant \mu_2. \end{cases} \tag{10}$$

6. *Parameters update*. Update the matrix B_{k+1} by a quasi-Newton formula, set $k = k + 1$ and go to Step 2.

In order to ease of reference, we define two index sets as below

$$I = \{k : \hat{\rho}_k \geqslant \mu_1\} \quad \text{and } J = \{k : \hat{\rho}_k < \mu_1\}.$$

At each iteration, Algorithm 1 generates a trial step d_k by solving the trust-region subproblem (2). Strong theoretical and implementation results for the proposed algorithm can be obtained if the trial step d_k satisfies

$$m_k(0) - m_k(d_k) \geqslant \theta \|g_k\| \min\left\{\delta_k, \frac{\|g_k\|}{\|B_k\|}\right\} \tag{11}$$

and

$$g_k^T d_k \leqslant -\theta \|g_k\| \min\left\{\delta_k, \frac{\|g_k\|}{\|B_k\|}\right\}, \tag{12}$$

where $0 < \theta < 1$ is a constant. Similar to [3], we can solve the trust-region subproblem inaccurately so that (11) and (12) hold. The following assumptions are used to analyze the convergence properties of Algorithm 1:

(H1) The level set $L(x_0) = \{x \in R^n | f(x) \leqslant f(x_0)\}$ satisfies $L(x_0) \subseteq \Omega$, where $\Omega \subseteq \mathbf{R}^n$ is a closed and bounded set.
(H2) There exists a positive constant m such that, for all $d \in \mathbf{R}^n$ and $k \in \mathbf{N}$, we have

$$m\|d\|^2 \leqslant d^T B_k d.$$

(H3) The matrix B_k is uniformly bounded, i.e., there exists a constant $M_1 > 0$ such that, for all $k \in \mathbf{N}$, $\|B_k\| \leqslant M_1$.

Remark 2.1. Let $f(x)$ be a twice continuously differentiable function. Then, (H1) implies that there exists a constant $M_2 > 0$ such that

$$\|G_k\| \leqslant M_2 \quad \forall\, x \in \Omega.$$

Lemma 2.1. *Suppose that the sequence $\{x_k\}$ is generated by Algorithm 1. Then, for all $k \in \mathbf{N} \cup \{0\}$, we have $x_k \in L(x_0)$ and $\{f_{l(k)}\}$ is a decreasing sequence.*

Proof. Using the definition of R_k and $f_{l(k)}$, we have

$$R_k = \eta_k f_{l(k)} + (1-\eta_k) f_k \leqslant \eta_k f_{l(k)} + (1-\eta_k) f_{l(k)} = f_{l(k)}. \tag{13}$$

It is clear that $R_0 = f_0$. By induction, we show that $x_k \in L(x_0)$, for all $k \in \mathbf{N}$. We assume that $x_i \in L(x_0)$, for $i = 1, 2, \ldots, k$. We then prove that $x_{k+1} \in L(x_0)$. To do so, we consider two cases:

Case 1. $k \in I$: We have

$$R_k - f(x_k + d_k) \geqslant \mu_1 (m_k(0) - m_k(d_k)) \geqslant 0.$$

Case 2. $k \in J$: using (9) and (12), we have

$$f(x_k + \alpha_k d_k) \leqslant R_k + \delta \alpha_k g_k^T d_k \leqslant R_k.$$

These two inequalities along with (13) show that

$$f_{k+1} \leqslant R_k \leqslant f_{l(k)} \leqslant f_0 \quad \forall k \in \mathbf{N} \cup \{0\}. \tag{14}$$

Thus, the sequence $\{x_k\}$ is contained in $L(x_0)$.

Now, we prove that the sequence $\{f_{l(k)}\}$ is a decreasing sequence. To this end, we consider two cases based on $k \in I$ or $k+1 \in J$.

For $k < N$, it is obvious that $m(k) = k$. Since, for any k, $f_k \leqslant f_0$, then we have $f_{l(k)} = f_0$.

For $k \geqslant N$, we have $m(k+1) \leqslant m(k) + 1$. Thus, from the definition of $f_{l(k+1)}$ and (14), we can write

$$f_{l(k+1)} = \max_{0 \leqslant j \leqslant m(k+1)} \{f_{k-j+1}\} \leqslant \max_{0 \leqslant j \leqslant m(k)+1} \{f_{k-j+1}\} = \max\{f_{l(k)}, f_{k+1}\} \leqslant f_{l(k)}. \tag{15}$$

Both cases show that the sequence $\{f_{l(k)}\}$ is a decreasing sequence. □

Corollary 2.1. *Suppose that (H1) holds and the sequence $\{x_k\}$ is generated by Algorithm 1. Then the sequence $\{f_{l(k)}\}$ is convergent.*

Proof. Lemma 2.1 along with (H1) imply that

$$\exists \lambda \quad \text{s.t. } \forall n \in \mathbf{N} \cup \{0\} : \lambda \leqslant f_{k+n} \leqslant f_{l(k+n)} \leqslant \cdots \leqslant f_{l(k+1)} \leqslant f_{l(k)}.$$

This shows that the sequence $\{f_{l(k)}\}$ is convergent. □

Lemma 2.2. *Suppose that the sequence $\{x_k\}$ is generated by Algorithm 1. Then we have*

$$f_{k+1} \leqslant R_{k+1} \quad \forall k \in \mathbf{N} \cup \{0\}. \tag{16}$$

Proof. From the definition of $f_{l(k+1)}$, we have $f_{k+1} \leqslant f_{l(k+1)}$, for all $k \in \mathbf{N}$. Thus

$$f_{k+1} = \eta_{k+1} f_{k+1} + (1-\eta_{k+1}) f_{k+1} \leqslant \eta_{k+1} f_{l(k+1)} + (1-\eta_{k+1}) f_{k+1} = R_{k+1} \quad \forall k \in \mathbf{N} \cup \{0\}.$$

This completes the proof of the lemma. □

Lemma 2.3. *Suppose that the sequence $\{x_k\}$ is generated by Algorithm 1. Then, Step 4 of the algorithm is well-defined.*

Proof. The proof is straight for $\hat{\rho}_k \geqslant \mu_1$. Now, let $\hat{\rho}_k < \mu_1$. We prove that the line search terminates in the finite number of steps. For establishing a contradiction, assume that there exists $k \in J$ such that

$$f(x_k + \rho^i s_k d_k) > R_k + \beta \rho^i s_k g_k^T d_k \quad \forall i \in \mathbf{N} \cup \{0\}. \tag{17}$$

From Lemma 2.2, we have $f_k \leqslant R_k$. This fact,along with (17), implies that

$$\frac{f(x_k + \rho^i s_k d_k) - f_k}{\rho^i s_k} > \beta g_k^T d_k \quad \forall i \in \mathbf{N} \cup \{0\}.$$

Since f is a differentiable function, by taking a limit, as $i \to \infty$, we obtain

$$g_k^T d_k \geqslant \beta g_k^T d_k.$$

Using the fact that $\beta \in (0, 1/2)$, this inequality leads us to $g_k^T d_k \geqslant 0$ which contradicts (12). Therefore, Step 4 in Algorithm 1 is well-defined. □

3. Convergence analysis

It is well-known that trust-region methods have strong global convergence [15,16,3,13]. In this section, we show that our proposed method inherits these properties from trust-region methods.

Lemma 3.1. *Suppose that the sequence $\{x_k\}$ is generated by Algorithm 1 and there exists a positive constant c_1 such that $\|d_k\| \leqslant c_1\|g_k\|$. Then, foe all $k \in J$, the steplength α_k satisfies*

$$\alpha_k > \frac{2\rho\theta(1-\beta)\min\left\{1, \frac{1}{c_1 M_1}\right\}}{c_1 M_2}. \tag{18}$$

Proof. Let $\alpha = \alpha_k/\rho$. Then, from Step 4 of Algorithm 1, we have

$$R_k + \beta\alpha g_k^T d_k < f(x_k + \alpha d_k).$$

Using Taylor expansion, (16) and Remark 2.1, we obtain

$$f_k + \beta\alpha g_k^T d_k \leqslant R_k + \beta\alpha g_k^T d_k < f_k + \alpha g_k^T d_k + \frac{1}{2}\alpha^2 d_k^T \nabla^2 f(\xi) d_k \leqslant f_k + \alpha g_k^T d_k + \frac{1}{2}\alpha^2 M_2 \|d_k\|^2,$$

where $\xi \in (x_k, x_k + \alpha d_k)$. Thus, we have

$$-(1-\beta) g_k^T d_k < \frac{1}{2}\alpha M_2 \|d_k\|^2. \tag{19}$$

On the other hand, from $\|d_k\| \leqslant c_1\|g_k\|$ and (12), we can write

$$g_k^T d_k \leqslant -\theta\|g_k\| \min\left\{\delta_k, \frac{\|g_k\|}{\|B_k\|}\right\} \leqslant -\theta\frac{\|d_k\|}{c_1} \min\left\{\|d_k\|, \frac{\|d_k\|}{c_1 M_1}\right\} = -\frac{\theta}{c_1}\min\left\{1, \frac{1}{c_1 M_1}\right\}\|d_k\|^2. \tag{20}$$

Considering (19) and (20), we obtain

$$(1-\beta)\frac{\theta}{c_1}\min\left\{1, \frac{1}{c_1 M_1}\right\}\|d_k\|^2 < \frac{M_2}{2\rho}\alpha_k\|d_k\|^2,$$

which completes the proof of the lemma. □

Lemma 3.2. *Suppose that all conditions of Lemma 3.1 hold. Then, we have*

$$\lim_{k\to\infty} f(x_{l(k)}) = \lim_{k\to\infty} f(x_k). \tag{21}$$

Proof. It is sufficient to prove the lemma in the following two cases:

Case 1. $k \in I$. It follows from the definition of x_{k+1} and (13) that

$$\frac{f_{l(k)} - f(x_k + d_k)}{m_k(0) - m_k(d_k)} \geqslant \frac{R_k - f(x_k + d_k)}{m_k(0) - m_k(d_k)} \geqslant \mu_1.$$

Now, similar to the same proof of Theorem 3.2 in [17], we can deduce that (21) holds.

Case 2. $k \in J$. For $k > N$, using (9) and (13), we obtain

$$f(x_{l(k)}) = f(x_{l(k)-1} + \alpha_{l(k)-1} d_{l(k)-1}) \leqslant R_{l(k)-1} + \beta\alpha_{l(k)-1} g_{l(k)-1}^T d_{l(k)-1} \leqslant f(x_{l(l(k))-1}) + \beta\alpha_{l(k)-1} g_{l(k)-1}^T d_{l(k)-1}.$$

Preceding inequality, along with $\beta > 0$ and Corollary 2.1, implies that

$$\lim_{k\to\infty} \alpha_{l(k)-1} g_{l(k)-1}^T d_{l(k)-1} = 0. \tag{22}$$

Thus, using (20), we can conclude that

$$\lim_{k\to\infty} \alpha_{l(k)-1}\|d_{l(k)-1}\| = 0. \tag{23}$$

The rest of the proof is similar to a theorem in [5]. □

Corollary 3.1. *Suppose that sequence $\{x_k\}$ is generated by Algorithm 1. Then we have*

$$\lim_{k\to\infty} R_k = \lim_{k\to\infty} f(x_k). \tag{24}$$

Proof. From (13) and (16), we have

$$f_k \leqslant R_k \leqslant f_{l(k)}.$$

This completes the proof by using Lemma 3.2. □

Lemma 3.3. *Suppose that all conditions of Lemma 3.1 hold. Assume that the sequence $\{x_k\}$ does not converge to a stationary point, i.e., there exists a constant $0 < \epsilon < 1$ such that for all $k \in \mathbf{N}$, we have $\|g_k\| > \epsilon$. Then, we have*

$$\lim_{k\to\infty} \min\left\{\delta_k, \frac{\epsilon}{L_k}\right\} = 0, \tag{25}$$

where $L_k = 1 + max_{1\leqslant i\leqslant k}\|B_k\|$.

Proof. We first show that there exists a constant φ such that

$$f(x_{k+1}) \leqslant R_k - \varphi \min\left\{\delta_k, \frac{\epsilon}{L_k}\right\} \quad \forall k \in \mathbf{N}. \tag{26}$$

To do so, we consider the following two cases:

Case 1. $k \in I$. It follows from the definition of x_{k+1}, (11) and the definition of L_k that

$$R_k - f(x_k + d_k) \geqslant \mu_1[m_k(0) - m_k(d_k)] \geqslant \theta\mu_1\|g_k\| \min\left\{\delta_k, \frac{\|g_k\|}{\|B_k\|}\right\} \geqslant \theta\mu_1\epsilon \min\left\{\delta_k, \frac{\epsilon}{L_k}\right\} = \varphi_1 \min\left\{\delta_k, \frac{\epsilon}{L_k}\right\}, \tag{27}$$

where $\varphi_1 = \theta\mu_1\epsilon$.

Case 2. $k \in J$. Using (9), (12) and (18) we obtain

$$\begin{aligned} f(x_k + \alpha_k d_k) &\leqslant R_k + \beta\alpha_k g_k^T d_k \leqslant R_k - \beta\alpha_k\|g_k\| \min\left\{\delta_k, \frac{\|g_k\|}{\|B_k\|}\right\} \leqslant R_k - \frac{2\rho\theta\beta(1-\beta)\min\left\{1, \frac{1}{c_1M_1}\right\}}{c_1M_2} \min\left\{\delta_k, \frac{\epsilon}{L_k}\right\} \\ &= R_k - \varphi_2 \min\left\{\delta_k, \frac{\epsilon}{L_k}\right\}, \end{aligned} \tag{28}$$

where $\varphi_2 = \frac{2\rho\theta\beta(1-\beta)\min\left\{1, \frac{1}{c_1M_1}\right\}}{c_1M_2}$.

Now, setting $\varphi = \min\{\varphi_1, \varphi_2\}$, we can conclude that (26) holds, for all $k \in \mathbf{N}$. This fact, along with Corollary 3.1, completes the proof. □

Lemma 3.4. *Suppose that all conditions of Lemma 3.3 hold. Then, for sufficiently large k, we have*

$$\delta_k \geqslant \epsilon/L_k, \tag{29}$$

Proof. The proof is similar to the proof of Lemma 3.6 and 3.7 in [2] and therefore it is omitted here. □

Theorem 3.5. *Suppose that (H1)–(H3) and all conditions of Lemma 3.3 hold. Then the sequence $\{x_k\}$ generated by Algorithm 1 satisfies*

$$\liminf_{k\to\infty} \|g_k\| = 0. \tag{30}$$

Proof. By contrary, assume that there exists a constant $\epsilon > 0$ such that $\|g_k\| \geqslant \epsilon$, for all $k \in \mathbf{N} \cup \{0\}$. Thus, there exists an integer $\bar{k}$ such that, for $k \geqslant \bar{k}$, (29) holds. This fact together with (13) and (26) imply that

$$f_{l(k)} \geqslant R_k \geqslant f_{k+1} + \varphi \min\left\{\delta_k, \frac{\epsilon}{L_k}\right\} \geqslant f_{k+1} + \epsilon\varphi\frac{1}{L_k}.$$

Thus, using Lemma 2.1, for $s = 0, 1, \ldots, N$, we can get

$$f_{l(k)} \geqslant \cdots \geqslant f_{l(k+s)} \geqslant f_{k+s+1} + \epsilon\varphi\frac{1}{L_{k+s}}.$$

This inequality, monotonicity of the sequence $\{L_k\}$ and Lemma 3.3 lead us to have

$$f_{l(k)} \geqslant \max\{f_{k+s+1} : 0 \leqslant s \leqslant N\} + \epsilon\varphi\frac{1}{L_{k+s}} = f_{l(k+N+1)} + \epsilon\varphi\frac{1}{L_{k+s}} \geqslant f_{l(k+N+1)} + \epsilon\varphi\frac{1}{L_{k+N+1}}. \tag{31}$$

Now, using (31), (H1) and Corollary 2.1, we obtain

$$\sum_{k=\bar{k}}^{\infty}\frac{1}{L_{k+N+1}} \leqslant \frac{1}{\epsilon\varphi}\sum_{k=\bar{k}}^{\infty}(f_{l(k)} - f_{l(k+N+1)}) = \frac{1}{\epsilon\varphi}\sum_{k=\bar{k}}^{\infty}\sum_{s=0}^{N}(f_{l(k+s)} - f_{l(k+s+1)}) \leqslant \frac{1}{\epsilon\varphi}\sum_{s=0}^{N}\sum_{k=1}^{\infty}\left(f_{l(k+s)} - f_{l(k+s+1)}\right) < \infty. \tag{32}$$

On the other hand, from (H3) and definition of L_k, we have $L_k \leqslant M_1 + 1$. Hence, we can conclude that

$$\sum_{k=\bar{k}}^{\infty}\frac{1}{M_1+1} \leqslant \sum_{k=\bar{k}}^{\infty}\frac{1}{L_k}.$$

This implies that $\sum_{k=\bar{k}}^{\infty}\frac{1}{L_k} = \infty$, which contradicts (32). This completes the proof of the theorem. □

Theorem 3.6. *Suppose that the sequence $\{x_k\}$ generated by Algorithm 1 converges to x^*. Moreover, assume that the matrix $\nabla^2 f(x^*)$ is a positive definite matrix, $\|B_k^{-1}g_k\| \leqslant \delta_k, d_k = -B_k^{-1}g_k$ and*

$$\lim_{k\to\infty}\frac{\|[B_k - \nabla^2 f(x^*)]d_k\|}{\|d_k\|} = 0. \tag{33}$$

Then, the sequence $\{x_k\}$ converges x^* superlinearly. Furthermore, the convergence is quadratically whenever $B_k = \nabla^2 f(x_k)$.

Proof. For $k \in I$, we have $x_{k+1} = x_k + d_k$, otherwise, we set $x_{k+1} = x_k + \alpha_k d_k$. Therefore, it is sufficient to show that $\alpha_k = 1$, for sufficiently large $k \in J$. Using Lemma 3.1 and the fact that the sequence $\{x_k\}$ converges to x^*, we have $d_k \to 0$ as $k \to \infty$. From (33), we have

$$\lim_{k\to\infty}\frac{\|g_k + \nabla^2 f(x^*)d_k\|}{\|d_k\|} = \lim_{k\to\infty}\frac{\|[B_k - \nabla^2 f(x^*)]d_k\|}{\|d_k\|} = 0,$$

which implies that

$$g_k^T d_k = -d_k^T\nabla^2 f(x^*)d_k + o(\|d_k\|^2). \tag{34}$$

Using the Taylor expansion, Lemma 2.2 and (34), there exists $\xi \in (x_k, x_k + d_k)$ such that

$$\begin{aligned} f(x_k + d_k) - R_k - \beta g_k^T d_k &\leqslant f(x_k + d_k) - f_k - \beta g_k^T d_k \leqslant (1-\beta)g_k^T d_k + \frac{1}{2}d_k^T\nabla^2 f(\xi)d_k \\ &= -(1-\beta)d_k^T\nabla^2 f(x^*)d_k + \frac{1}{2}d_k^T\nabla^2 f(\xi)d_k + o(\|d_k\|^2). \end{aligned}$$

From the fact that $\nabla^2 f(\xi) \to \nabla^2 f(x^*)$, as $k \to \infty$, and $\beta \in (0, \frac{1}{2})$, for sufficiently large k, we obtain

$$f(x_k + d_k) \leqslant R_k + \beta g_k^T d_k, \tag{35}$$

Therefore, for sufficiently large k, we have $\alpha_k = 1$ which shows that Algorithm 1 has been reduced to the superlinealy convergent standard quasi-Newton methods [16]. The same argument can be done to drive $\alpha_k = 1$ whenever $B_k = \nabla^2 f(x_k)$. Thus, in this case, the method is reduced to the quadratically convergent Newton method [16]. □

4. Preliminary numerical experiments

Here,we provide some numerical experiments to show the performance of our proposed algorithm. The new algorithm, denoted by NMTLN from now on, have been compared with the MTL algorithm of Nocedal and Yuan [3], the NMTLG nonmonotone algorithm of Gu and Mo [2] and a version of Algorithm 1 using $f_{l(k)}$ instead of R_k, denoted by NMTLG, on 60 large-scale test problems with dimension 500 to 5000 from [18]. Due to references [15,19], we exploit the following value for the parameters of Algorithm 1:

$$\mu_1 = 0.05, \quad \mu_2 = 0.9, \quad \rho = 0.5, \quad N = 10, \quad \delta_0 = 1, \quad \bar{\delta} = 100.$$

In addition, we choose $\eta_0 = 0.15$ and update η_k by the following recursive formula

$$\eta_k = \begin{cases} \eta_0/2, & \text{if } k = 1, \\ (\eta_{k-1} + \eta_{k-2})/2, & \text{if } k \geqslant 2. \end{cases}$$

Table 1
Numerical results.

Problem name	Dim	MTL n_i/n_f	NMTLM n_i/n_f	NMTLG n_i/n_f	NMTLN n_i/n_f
Extended Wood	500	1613/2023	1411/3003	2162/3738	1872/3304
Extended Rosenbrock	500	881/1093	627/1296	1020/1724	294/556
Generalized Rosenbrock	500	5040/6482	3723/7211	3869/5628	3579/5826
Extended White and Holst	500	2329/2892	1509/3126	1605/2718	1488/2790
Generalized White and Holst	500	6986/8730	5072/9073	5961/8168	2977/5353
Perturbed Tridiagonal Quadratic	500	844/1087	789/1710	1126/2159	890/1733
Almost Perturbed Quadratic	500	835/1091	799/1733	1123/2169	859/1670
FLETCHCR	500	4333/5534	2845/6079	2184/4200	2727/4705
Extended Powell	1000	3062/3809	1560/3198	1374/2249	1382/2697
Quadratic QF1	1000	1640/2139	1575/3385	1762/3307	1772/3383
Quadratic QF2	1000	1906/2452	1848/3980	2141/4009	2107/2892
Perturbed Quadratic	1000	1654/2148	1572/3386	1733/3334	1837/3413
Generalized Quartic	1000	19/23	22/22	19/19	19/19
Extended Rosenbrock	1000	52/55	52/55	53/53	53/53
Partial Perturbed Quadratic	1000	440/562	370/810	1171/2254	471/943
DIXON3DQ	1000	1472/1925	1215/1930	1349/1424	1195/1639
TRIDIA	1000	3867/4947	2827/6070	2640/5212	2652/4981
NONDQUAR	1000	1186/1188	1266/1273	1232/1240	1218/1219
POWER	1000	17583/22372	11218/23966	9343/18742	9415/18339
Diagonal 2	1000	220/220	220/220	218/218	218/218
CUBE	1000	9048/11065	5787/10368	2526/3332	3216/4572
BIGGSB1	1000	1470/1923	1151/1822	1336/1415	1198/1614
DIXMAANE	1500	173/179	208/215	343/369	220/222
DIXMAANF	1500	146/146	146/146	144/144	144/144
DIXMAANG	1500	146/146	146/146	380/405	228/255
DIXMAANH	1500	153/154	144/144	193/193	193/193
DIXMAANI	1500	2807/2807	2807/2807	2809/2809	2809/2809
DIXMAANJ	1500	1630/1630	1630/1630	14620/14681	628/642
Extended Beale	2000	22/25	18/20	15/17	15/17
Extended PSC1	2000	20/21	19/20	17/17	17/17
Generalized PSC1	2000	101/104	146/164	490/550	95/104
VARDIM	2000	59/59	59/59	58/58	58/58
HIMMELBG	2000	24/24	24/24	22/22	22/22
QUARTC	2000	23/23	23/23	22/22	22/22
LIARWHD	2000	31/32	27/27	26/26	31/32
Extended Tridiagonal 1	2000	23/24	23/24	23/23	23/23
Extended Freudenstein and Roth	3000	58/58	20/20	15/15	15/15
Extended Penalty	3000	63/67	61/66	80/80	74/75
DQDRTIC	3000	36/45	20/25	25/26	22/24
EG2	3000	178/190	135/154	119/127	143/163
BDEXP	3000	24/24	24/24	23/23	23/23
Raydan 2	3000	10/10	10/10	9/9	9/9
Extended TET	3000	9/9	9/9	10/10	9/10
Diagonal 4	3000	9/9	9/9	6/6	6/6
Extended QP1	3000	22/23	24/24	24/24	24/24
DIXMAANA	3000	15/15	15/15	14/14	14/14
DIXMAANB	3000	15/15	15/15	40/46	41/52
DIXMAANC	3000	37/47	37/47	37/47	37/47
DIXMAAND	3000	56/70	55/70	58/71	61/76
DENSCHNB	5000	14/14	14/14	12/12	12/12
ARWHEAD	5000	6/7	6/6	6/6	6/6
HIMMELH	5000	9/9	9/9	9/9	9/9
SINCOS	5000	18/19	18/20	17/17	17/17
Diagonal 5	5000	10/10	10/10	8/8	8/8
Diagonal 7	5000	15/15	11/11	9/9	9/9
Diagonal 8	5000	11/14	9/9	8/8	8/8
Extended BD1	5000	15/16	14/15	16/17	14/16
Extended EP1	5000	4/4	4/4	4/4	4/4
Full Hessian FH3	5000	9/9	9/9	7/7	7/7
Extended Himmelblau	5000	13/13	13/13	15/16	14/16

We have implemented above mentioned algorithms in MATLAB 7.4 environment with double precision format. In our experiments, algorithms are being stopped when

$$\|\nabla f(x_k)\| \leqslant 10^{-5}$$

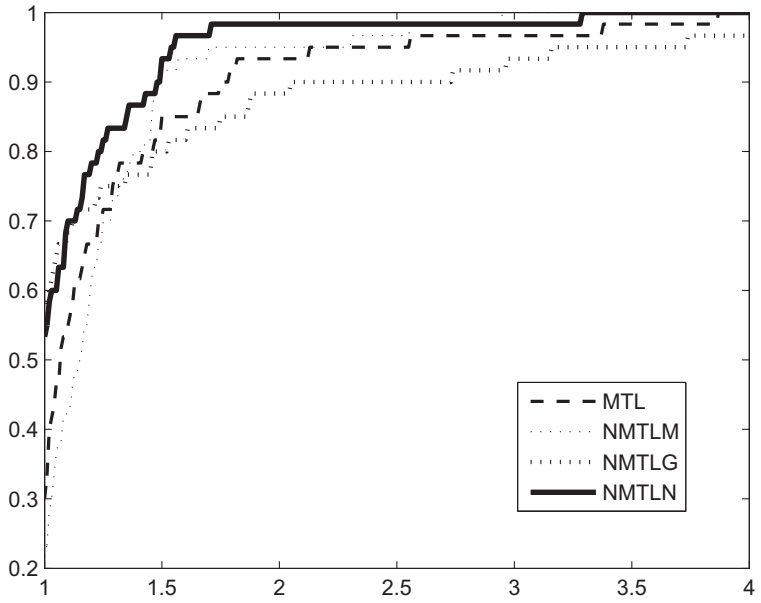

Fig. 1. Performance profile for $n_f + 3n_i$.

or the total number of iterations exceeds 20000. During running of the algorithms, we have checked the codes to make sure that the convergence happens to the same local minimizer, and therefore we have just provided those results in which all algorithms converge to the same minimizer. In all algorithms, the matrix B_k is updated by the following BFGS formula

$$B_{k+1} = B_k + \frac{y_k y_k^T}{s_k^T y_k} - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k},$$

where $s_k = x_{k+1} - x_k$ and $y_k = g_{k+1} - g_k$. However, we do not update B_k whenever the curvature condition, i.e. $s_k^T y_k > 0$, does not hold.

In Table 1, n_i and n_f indicate the total number of iterations and function evaluations, respectively. Considering Algorithm 1, we know that the total number of iterations and gradient evaluations are equal. Thus, we just exploit the total number of iterations and function evaluations as a measure of performance for the algorithms. Table 1 shows that in the most cases both the total number of iterations and function evaluations of the new algorithm is partly less than the other presented algorithms. However, in some cases, the new algorithm is not the best algorithm, but it mostly has a better numerical performance compared with the other presented algorithms. In order to have a better comparison, the performance of these four algorithms, based on $n_c = n_f + 3n_i$, has been assessed in Fig. 1 using the performance profile of Dolan and Moré [20].

From Fig. 1, firstly, we can see that NMTLG and NMTLN are so competitive in the sense of how many times they are the best algorithms. Secondly, we observe that the proposed algorithm is the best algorithm on more than 55% of the test problems. Thirdly, the proposed algorithm solves all of test functions. Finally, we can see that the proposed algorithm grows up faster than the other considered algorithms. This means that whenever the proposed algorithm is not the best algorithm, its performance index is close to the performance index of the best algorithm. Therefore, we can deduce that the new algorithm is more efficient and robust than the other considered trust-region algorithms for solving large-scale unconstrained optimization problems.

5. Conclusions

In this paper, a variant nonmonotone trust-region algorithm for solving large-scale unconstrained optimization problems is proposed. Unlike the standard trust-region method, the proposed algorithm does not reject a trial step, but performs a new nonmonotone Armijio-type line search in direction of the rejected trial step in order to avoid resolving the trust-region subproblem instead. This procedure highly affects on the total number of subproblem solving. Theoretical analysis shows that the proposed algorithm inherits the global convergence of standard trust-region algorithms to the first-order stationary points under classical assumptions. Under some suitable conditions, the superlinear and the quadratic convergence results are established. Preliminary numerical experiments indicate the efficiency and the robustness of the proposed algorithm for large-scale unconstrained optimization problems.

Acknowledgement

The first two authors would like to thank the research council of Razi University for supporting this research. The last author would like to thank the research council of K.N. Toosi University of Technology for supporting this research.

References

[1] E.M. Gertz, A quasi-Newton trust region method, Math. Program. 100 (2004) 447–470.
[2] N. Gu, J. Mo, Incorporating nonmonotone strategies into the trust region for unconstrained optimization, Comput. Math. Appl. 209 (2007) 97–108.
[3] J. Nocedal, Y. Yuan, Combining trust region and line search techniques, in: Y. Yuan (Ed.), Advanced in Nonlinear Programming, Kluwer Academic Publishers, Dordrecht, 1996, pp. 153–175.
[4] R.M. Chamberlain, M.J.D. Powell, C. Lemarechal, H.C. Pedersen, The watchdog technique for forcing convergence in algorithm for constrained optimization, Math. Program. Stud. 16 (1982) 1–17.
[5] L. Grippo, F. Lamparillo, S. Lucidi, A nonmonotone line search technique for Newton's method, SIAM J. Numer. Anal. 23 (1986) 707–716.
[6] L. Grippo, F. Lamparillo, S. Lucidi, A truncated Newton method with nonmonotone linesearch for unconstrained optimization, J. Optim. Theory Appl. 60 (3) (1989) 401–419.
[7] N.Y. Deng, Y. Xiao, F.J. Zhou, Nonmonotonic trust region algorithm, J. Optim. Theory Appl. 76 (1993) 259–285.
[8] Y. Xiao, F.J. Zhou, Nonmonotone trust region methods with curvilinear path in unconstrained optimization, Computing 48 (1992) 303–317.
[9] F. Zhou, Y. Xiao, A class of nonmonotone stabilization trust region methods, Computing 53 (2) (1994) 119–136.
[10] Y. Xiao, E.K.W. Chu, Nonmonotone trust region methods, Technical Report 95/17, Monash University, Clayton, Australia, 1995.
[11] Ph.L. Toint, An assessment of nonmonotone line search technique for unconstrained optimization, SIAM J. Sci. Comput. 17 (1996) 725–739.
[12] Ph.L. Toint, Non-monotone trust region algorithm for nonlinear optimization subject to convex constraints, Math. Program. 77 (1997) 69–94.
[13] H.C. Zhang, W.W. Hager, A nonmonotone line search technique and its application to unconstrained optimization, SIAM J. Optim. 14 (4) (2004) 1043–1056.
[14] J. Mo, C. Liu, S. Yan, A nonmonotone trust region method based on nonincreasing technique of weighted average of the successive function value, J. Comput. Appl. Math. 209 (2007) 97–108.
[15] A.R. Conn, N.I.M. Gould, Ph.L. Toint, Trust-Region Methods, Society for Industrial and Applied Mathematics, SIAM, Philadelphia, 2000.
[16] J. Nocedal, S.J. Wright, Numerical Optimization, Springer, NewYork, 2006.
[17] M. Ahookhosh, K. Amini, A nonmonotone trust region method with adaptive radius for unconstrained optimization, Comput. Math. Appl. 60 (2010) 411–422.
[18] N. Andrei, An unconstrained optimization test functions collection, Adv. Model. Optim. 10 (1) (2008) 147–161.
[19] N.I. M Gould, D. Orban, A. Sartenaer, Ph.L. Toint, Sensitivity of trust region algorithms to their parameters, Quart. J. Oper. Res. 3 (2005) 227–241.
[20] E. Dolan, J.J. Moré, Benchmarking optimization software with performance profiles, Math. Program. 91 (2002) 201–213.