AN IMPROVED NONMONOTONE ADAPTIVE TRUST REGION METHOD

Yanqin Xue, Hongwei Liu, Xi'an, Zexian Liu, Hezhou

Received May 2, 2018. Published online April 29, 2019.

Abstract. Trust region methods are a class of effective iterative schemes in numerical optimization. In this paper, a new improved nonmonotone adaptive trust region method for solving unconstrained optimization problems is proposed. We construct an approximate model where the approximation to Hessian matrix is updated by the scaled memoryless BFGS update formula, and incorporate a nonmonotone technique with the new proposed adaptive trust region radius. The new ratio to adjusting the next trust region radius is different from the ratio in the traditional trust region methods. Under some suitable and standard assumptions, it is shown that the proposed algorithm possesses global convergence and superlinear convergence. Numerical results demonstrate that the proposed method is very promising.

Keywords: unconstrained optimization; trust region method; scaled memoryless BFGS update; nonmonotone technique; global convergence

MSC 2010: 90C30

1. Introduction

In this paper, we consider the unconstrained optimization problem

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

where $f\colon \mathbb{R}^n \to \mathbb{R}$ is a twice continuously differentiable function.

Trust region methods and line search methods are two popular iterative approaches for solving problem (1.1). Line search methods refer to a procedure that generates

This research is supported by National Science Foundation of China (No. 11461021), Shaanxi Science Foundation (No. 2017JM1014), and Guangxi Science Foundation (Nos. 2018GXNSFBA281180).

a search direction, and focus their efforts on finding a suitable stepsize along this direction, while trust region methods use a different approach. The trust region methods can be traced back to Marquardt [15] for solving nonlinear least squares problems. The modern versions of trust region methods were first proposed by Powell [17] and Winfield [22]. In the trust region methods, the iteration is in the form of

$$x_{k+1} = x_k + d_k, \quad k = 0, 1, \ldots, \tag{1.2}$$

where the trial step d_k is obtained by solving the subproblem

$$\begin{aligned} &\min \ \ m_k(d) = f_k + g_k^{\mathrm{T}} d + \frac{1}{2} d^{\mathrm{T}} B_k d \\ &\text{s.t.} \quad \|d\| \leqslant \Delta_k, \end{aligned} \tag{1.3}$$

where $f_k = f(x_k)$, $g_k = g(x_k) = \nabla f(x_k)$, B_k is an approximation of the Hessian matrix, $\|\cdot\|$ denotes the Euclidean norm and Δ_k is the TR radius. Trust region methods can not only replace line search to obtain the global convergence, but also handle the difficulty caused by ill-conditioned problems and nonsmooth problems, so they play a crucial role in numerical optimization.

It is well-known that the update strategy of the TR radius affects the number of iterations and convergence of the algorithm. It has attracted many researchers [9], [24] to update the TR radius by using the gradient or the Hessian matrix information. More recently, Shi and Guo [19] proposed an adaptive TR radius. In their method, a vector q_k is chosen so that it satisfies the angle condition [21], i.e.

$$-\frac{g_k^{\mathrm{T}} q_k}{\|g_k\| \cdot \|q_k\|} \geqslant \tau, \tag{1.4}$$

where $\tau \in (0, 1)$. Ahamad Kamandi et al. [12] proposed a modification of q_k

$$q_k = \begin{cases} -g_k, & \text{if } k = 0 \text{ or } \dfrac{-(g_k^{\mathrm{T}} d_{k-1})}{\|g_k\| \|d_{k-1}\|} \leqslant \tau, \\ d_{k-1}, & \text{otherwise,} \end{cases} \tag{1.5}$$

where d_{k-1} is a solution of subproblem (1.3) and $\tau \in (0, 1)$. Clearly, q_k satisfies condition (1.4). The TR radius is seriously reduced when x_k is far from the optimum and the matrix B_k is close to singular, in order to avoid getting a very small TR radius, s_k is determined by

$$s_k = \begin{cases} -\dfrac{g_k^{\mathrm{T}} q_k}{q_k^{\mathrm{T}} B_k q_k} \|q_k\|, & \text{if } k = 0, \\ \max\Big(-\dfrac{g_k^{\mathrm{T}} q_k}{q_k^{\mathrm{T}} B_k q_k} \|q_k\|, \ \gamma \Delta_{k-1}\Big), & \text{otherwise,} \end{cases} \tag{1.6}$$

where $\gamma > 1$ and q_k is computed by (1.5). The TR radius is updated as

$$\Delta_k = t^p \min\{s_k, \bar{\Delta}\}, \tag{1.7}$$

where $\bar{\Delta} > 0$ is a positive constant, $t \in (0, 1)$, and p is a nonnegative integer.

Furthermore, computational experiments confirm that iterative algorithms with suitable nonmonotone technique have better convergence behavior. The earliest nonmonotone technique is the so called watch-dog technique, which was proposed by Chamberlain et al. [5] with the purpose of overcoming the Maratos Effect [14]. Later on, Grippo et al. [11] proposed a nonmonotone technique for Newton's method, in which a line search is performed so that the stepsize α_k satisfies condition $f(x_k + \alpha_k d_k) \leqslant f_{l(k)} + \beta\alpha_k g_k^{\mathrm{T}} d_k$, where $\beta \in (0, 1)$, the nonmonotone term $f_{l(k)}$ is defined by

$$f_{l(k)} = \max_{0 \leqslant j \leqslant m(k)} \{f(x_{k-j})\}, \tag{1.8}$$

in which $m(0) = 0$, $0 \leqslant m(k) \leqslant \min\{m(k-1) + 1, M_1\}$ for $k \geqslant 1$, and M_1 is a given nonnegative integer. Many authors [20], [23], [7], [18] generalized the Grippo's nonmonotone term into the adaptive trust region framework and obtained good numerical results. Peyghami and Tarzanagh [16] provided a new adaptive trust region algorithm which incorporates a variant of nonmonotone technique.

In this paper, we use a scaled memoryless BFGS update formula to update B_k in (1.3), also, apply a nonmonotone techniques into trust region method to present an improved nonmonotone adaptive trust region method. Under mild conditions, we analyze the global convergence and superlinear convergence of the proposed method. Numerical results show that for the CUTEr library and the test problem collection given by Andrei [3], the proposed method is superior to the adaptive methods in [16], [18].

The outline of the paper is as follows. In Section 2, an improved nonmonotone adaptive trust region method based on a scaled memoryless BFGS update formula is presented in details. In Section 3, we establish the global and superlinear convergence property of the new algorithm under some suitable assumptions. Some preliminary numerical results are given in Section 4. Finally, we end the paper by some concluding remarks in Section 5.

2. The structure of the new algorithm

In this section, we develop a new strategy to update B_k in (1.3), apply a nonmonotone technique to the frame of the trust region methods and present an improved nonmonotone adaptive trust region method.

We first establish update strategy of the matrix B_k by using the scaled memoryless BFGS formula at each iteration. The scaled memoryless BFGS update formula is defined by

$$B_{k+1} = \theta_k I - \theta_k \frac{d_k d_k^{\mathrm{T}}}{d_k^{\mathrm{T}} d_k} + \frac{y_k y_k^{\mathrm{T}}}{d_k^{\mathrm{T}} y_k}, \tag{2.1}$$

where

$$y_k = g_{k+1} - g_k, \quad \theta_k = \frac{d_k^{\mathrm{T}} y_k}{\|d_k\|^2}.$$

Due to the small memory required and low computational cost, it is widely used to solve unconstrained optimization problems [4]. Numerical and theoretical superiority of the scaled memoryless BFGS update methods motivated us to deal with the matrix B_k update in the trust region methods. Obviously, if $d_k^{\mathrm{T}} y_k > 0$ holds, then B_{k+1} in (2.1) is positive definite. When $d_k^{\mathrm{T}} y_k \leqslant 0$, in the general trust region algorithms, the matrix B_k is not updated, i.e. $B_{k+1} = B_k$. It is observed by numerical experiments that the algorithm is of poor performance. We consider updating the matrix B_k according to a valid formula instead of taking $B_{k+1} = B_k$. For nonconvex unconstrained optimization problems, Li and Fukushima [13] proposed a modified BFGS formula

$$B_{k+1} = B_k - \frac{B_k d_k d_k^{\mathrm{T}} B_k}{d_k^{\mathrm{T}} d_k} + \frac{y_k^* (y_k^*)^{\mathrm{T}}}{d_k^{\mathrm{T}} y_k^*}, \tag{2.2}$$

where

$$y_k^* = y_k + \|g_k\| \Big(1 - \frac{d_k^{\mathrm{T}} y_k}{\|d_k\|^2}\Big) d_k.$$

It is easy to see that B_{k+1} in (2.2) is positive definite when $d_k^{\mathrm{T}} y_k \leqslant 0$. Therefore, we determined B_{k+1} by formula (2.2) if $d_k^{\mathrm{T}} y_k \leqslant 0$. In conclusion, B_{k+1} is updated by

$$B_{k+1} = \begin{cases} \theta_k I - \theta_k \dfrac{d_k d_k^{\mathrm{T}}}{d_k^{\mathrm{T}} d_k} + \dfrac{y_k y_k^{\mathrm{T}}}{d_k^{\mathrm{T}} y_k}, & \text{if } d_k^{\mathrm{T}} y_k > 0, \\ B_k - \dfrac{B_k d_k d_k^{\mathrm{T}} B_k}{d_k^{\mathrm{T}} d_k} + \dfrac{y_k^* (y_k^*)^{\mathrm{T}}}{d_k^{\mathrm{T}} y_k^*}, & \text{otherwise.} \end{cases} \tag{2.3}$$

In order to enhance the numerical performance of the algorithm, we introduce the nonmonotone technique to the trust region algorithm. In [11], Grippo et al.

proposed nonmonotone technique that contains some drawbacks. For example, a good function value generated at any iteration may be abandoned; the numerical performances are seriously dependent on the choice of parameter M_1. To cope with these defects, Ahookhosh and Amini [2] proposed a new nonmonotone scheme which is a convex combination of the maximum of function value of some prior successful iterates and the current function value, it is also observed that this nonmonotone technique was superior to the nonmonotone technique (1.8). The nonmonotone term in [2] is defined by

$$R_k = \eta_k f_{l(k)} + (1-\eta_k) f_k, \tag{2.4}$$

where $\eta_k \in [\eta_{\min}, \eta_{\max}]$; $\eta_{\min} \in [0,1)$ and $\eta_{\max} \in [\eta_{\min}, 1]$ are two prefixed constants, and $f_{l(k)}$ is defined by (1.8).

In our method, the actual reduction of the objective function value is

$$Ared_k = R_k - f(x_k + d_k), \tag{2.5}$$

and the predicted reduction of the objective function value is

$$Pred_k = m_k(0) - m_k(d_k). \tag{2.6}$$

Now, the modified ratio is given by

$$r_k = \frac{Ared_k}{Pred_k} = \frac{R_k - f(x_k + d_k)}{m_k(0) - m_k(d_k)}, \tag{2.7}$$

where R_k is computed by (2.4).

We describe the new algorithm as below:

Algorithm 2.1 AINATR (An improved nonmonotone adaptive trust region method)

Step 0. Let $x_0 \in \mathbb{R}^n$, a positive definite matrix $B_0 \in \mathbb{R}^{n\times n}$, $\tau \in (0,1)$, $\bar{\Delta} > 0$, $t \in (0,1)$, $u > 0$, $\gamma > 1$, $\eta_{\min} \in [0,1)$, $\eta_{\max} \in [\eta_{\min}, 1]$, $R_0 = f(x_0)$, a positive integer M_1 and $\varepsilon > 0$ be given. Set $k := 0$.

Step 1. If $\|g_k\| \leqslant \varepsilon$, then stop.

Step 2. Compute q_k according to expression (1.5), s_k by (1.6) and set $p = 0$.

Step 3. Compute Δ_k by (1.7), solve subproblem (1.3) to find the trial step d_k and compute r_k by (2.7).

Step 4. If $r_k < u$, then $p = p + 1$. Goto Step 3.

Step 5. Set $x_{k+1} = x_k + d_k$.

Step 6. Choose $\eta_k \in [\eta_{\min}, \eta_{\max}]$ and update Hessian approximation B_k by (2.3). Set $k := k + 1$ and goto Step 1.

In Algorithm 2.1, the loop between Step 3 and Step 4 is called the inner cycle. If $r_k < u$, it is called an unsuccessful iteration. Note that the parameter u in Step 4 plays an important role in deciding whether the trial step d_k would be accepted or not.

3. Convergence analysis

In this section, we intend to discuss the global convergence property and the superlinear convergence rate of Algorithm 2.1. In order to verify these properties, we need to make the following assumptions:

(A1) The level set $L_0 = \{x \in \mathbb{R}^n;\ f(x) \leqslant f(x_0)\}$ is bounded and f is twice continuously differentiable over L_0;

(A2) The matrix B_k is uniformly bounded, i.e. there exists a positive constant M such that $\|B_k\| \leqslant M$ for all $k \in \mathbb{N} \cup \{0\}$.

To establish the global convergence of Algorithm 2.1, we first prove some useful lemmas.

Lemma 3.1. *Suppose that the sequence $\{x_k\}$ is generated by Algorithm* 2.1. *Then we get*

$$|f_k - f(x_k + d_k) - Pred_k| \leqslant O(\|d_k\|^2). \tag{3.1}$$

P r o o f. The inequality is obtained by Taylor's expansion and (A2), the proof can be found in [5]. □

Lemma 3.2. *If* (A2) *holds and d_k is a solution of* (1.3), *then*

$$m_k(0) - m_k(d_k) \geqslant \frac{1}{2} t^{p_k} \min\Big\{ \frac{1}{M}\Big(\frac{-g_k^{\mathrm{T}} q_k}{\|q_k\|}\Big)^2,\ \bar{\Delta}\Big(\frac{-g_k^{\mathrm{T}} q_k}{\|q_k\|}\Big)\Big\}, \tag{3.2}$$

where $t \in (0,1)$, p_k is the smallest nonnegative integer for which $r_k \geqslant u$, $u \in (0,1)$.

P r o o f. The proof is similar to the proof of Lemma 3.2 in [12]. □

Lemma 3.3. *Let $\{x_k\}$ be the sequence generated by Algorithm* 2.1. *Then we have*

$$f_k \leqslant R_k, \quad \forall\, k \in \mathbb{N}. \tag{3.3}$$

P r o o f. The proof is similar to the proof of Lemma 3.5 in [2] and the details are omitted. □

Lemma 3.4. *Suppose that the sequence $\{x_k\}$ is generated by Algorithm 2.1. Then the sequence $\{f_{l(k)}\}$ is a decreasing sequence.*

P r o o f. The proof can be found in Lemma 4 of [2]. □

Lemma 3.5. *Step 3 and Step 4 of Algorithm 2.1 are well-defined in the sense that at each iteration they terminate finitely.*

P r o o f. We prove this lemma by contradiction. Suppose that the inner cycle between Step 3 and Step 4 in Algorithm 2.1 is infinite. We define the cycling index at iteration k by $k(i)$. Then we have

$$r_{k(i)} < u, \quad i = 1, 2, \dots \tag{3.4}$$

Since x_k is not the optimum, there is a constant $\varepsilon > 0$ such that $\|g_k\| > \varepsilon$, which yields together with (1.4)

$$-\frac{g_k^{\mathrm{T}} q_k}{\|q_k\|} > \tau\varepsilon. \tag{3.5}$$

Let $d_{k(i)}$ be the solution of subproblem (1.3) corresponding to $p_{k(i)} \in \{0\} \cup \mathbb{N}$. Then it follows from Lemma 3.1, (3.2), and (3.5) that

$$\begin{aligned} \Big|\frac{f_k - f(x_k + d_{k(i)})}{m_k(0) - m_k(d_{k(i)})} - 1\Big| &= \Big|\frac{f_k - f(x_k + d_{k(i)}) - Pred_{k(i)}}{m_k(0) - m_k(d_{k(i)})}\Big| \\ &\leqslant \frac{O(\|d_{k(i)}\|^2)}{m_k(0) - m_k(d_{k(i)})} \\ &\leqslant \frac{O(\|d_{k(i)}\|^2)}{\frac{1}{2}t^{p_{k(i)}} \min\{(-g_k^{\mathrm{T}} q_k/\|q_k\|)^2/M, \bar{\Delta}(-g_k^{\mathrm{T}} q_k/\|q_k\|)\}} \\ &< \frac{O(\|d_{k(i)}\|^2)}{\frac{1}{2}t^{p_{k(i)}} \min\{(\tau\varepsilon)^2/M, \bar{\Delta}(\tau\varepsilon)\}}. \end{aligned} \tag{3.6}$$

From the assumption that the inner cycle is infinite and from (1.7) we obtain $\Delta_{k(i)} \to 0$ with $i \to \infty$. Hence, $\|d_{k(i)}\| \leqslant \Delta_{k(i)} \leqslant t_{k(i)}^p s_k$ implies that the right-hand side of equation (3.6) tends to zero. Therefore, it is obvious that for sufficiently large i

$$\lim_{i\to\infty} \frac{f_k - f(x_k + d_{k(i)})}{m_k(0) - m_k(d_{k(i)})} = 1. \tag{3.7}$$

Combining (2.7) and Lemma 3.3, we obtain

$$r_{k(i)} = \frac{R_k - f(x_k + d_{k(i)})}{m_k(0) - m_k(d_{k(i)})} \geqslant \frac{f_k - f(x_k + d_{k(i)})}{m_k(0) - m_k(d_{k(i)})}. \tag{3.8}$$

This inequality implies that for $i \to \infty$, $r_{k(i)} \geqslant u \in (0, 1)$, which is contradictory to (3.4). This completes the proof of Lemma 3.5. □

Based on the above lemmas, we prove the global convergence of Algorithm 2.1.

Theorem 3.1. *Suppose that* (A1) *holds and the sequence* $\{x_k\}$ *is generated by Algorithm* 2.1. *Then*

$$\liminf_{k\to\infty} \|g_k\| = 0. \tag{3.9}$$

P r o o f. By contradiction, suppose there exists a constant $\delta > 0$ such that

$$\|g_k\| \geqslant \delta, \quad k \in \{0\} \cup \mathbb{N}. \tag{3.10}$$

Using (2.7) and $r_k \geqslant u$, we conclude that

$$f(x_k + d_k) \leqslant R_k - uPred_k. \tag{3.11}$$

From the definitions of R_k and $f_{l(k)}$ we can get

$$R_k = \eta_k f_{l(k)} + (1 - \eta_k) f_k \leqslant \eta_k f_{l(k)} + (1 - \eta_k) f_{l(k)} = f_{l(k)}. \tag{3.12}$$

Using (3.11) and (3.12), we have

$$uPred_k \leqslant f_{l(k)} - f(x_k + d_k). \tag{3.13}$$

Replacing k with $l(k) - 1$ and using Lemma 3.2 yield

(3.14)

$$\begin{aligned} f_{l(l(k)-1)} - f_{l(k)} &\geqslant uPred_{l(k)-1} \\ &\geqslant \frac{1}{2} u t^{p_{l(k)-1}} \min\Big\{ \frac{1}{M} \Big(\frac{-g_{l(k)-1}^{\mathrm{T}} q_{l(k)-1}}{\|q_{l(k)-1}\|} \Big)^2,\ \bar{\Delta} \Big(\frac{-g_{l(k)-1}^{\mathrm{T}} q_{l(k)-1}}{\|q_{l(k)-1}\|} \Big) \Big\}. \end{aligned}$$

From Lemma 3.4, we know that the sequence $\{f_{l(k)}\}$ is monotonically nonincreasing. According to Assumption (A1) that f has a lower bound, we can deduce that $\{f_{l(k)}\}$ is convergent. So we have from (3.14)

$$\sum_{k=0}^{\infty} t^{p_{l(k)-1}} \min\Big\{ \frac{1}{M} \Big(\frac{-g_{l(k)-1}^{\mathrm{T}} q_{l(k)-1}}{\|q_{l(k)-1}\|} \Big)^2,\ \bar{\Delta} \Big(\frac{-g_{l(k)-1}^{\mathrm{T}} q_{l(k)-1}}{\|q_{l(k)-1}\|} \Big) \Big\} < \infty. \tag{3.15}$$

Inequalities (3.15) and (3.10) imply that there exists an infinite index set T such that

$$\lim_{k\to\infty,\ k\in T} \frac{-g_{l(k)-1}^{\mathrm{T}} q_{l(k)-1}}{\|q_{l(k)-1}\|} \neq 0, \tag{3.16}$$

which implies that

$$\lim_{k\to\infty,\ k\in T} t^{p_{l(k)-1}} = 0. \tag{3.17}$$

From (1.7), $\Delta_{l(k)-1} \to 0$ as $k \to \infty$ and $k \in T$. Without loss of generality, we assume that for all $k \in T$ there are more than one inner cycles performing in the loop between Steps 3 and 4 at the kth iterate. So, the solution $\bar{d}_k$ of the subproblem

$$\begin{aligned} &\min \ \ m_k(d) = f_k + g_k^{\mathrm{T}} d + \frac{1}{2} d^{\mathrm{T}} B_k d \\ &\text{s.t.} \quad \|d\| \leqslant \Delta_k / t, \ k \in T, \end{aligned} \tag{3.18}$$

is not accepted at the kth iteration for all $k \in T$, which means

$$r_k = \frac{R_k - f(x_k + \bar{d}_k)}{m_k(0) - m_k(\bar{d}_k)} < u, \quad k \in T. \tag{3.19}$$

On the other hand, by Lemma 3.5, we have $r_k \geqslant u$ for sufficiently large $k \in T$, which contradicts (3.19). Consequently, (3.9) holds, which completes the proof. □

Under suitable conditions, we analyze the superlinear convergence of Algorithm 2.1. We first make an assumption.

(A3) The matrix B_k is invertible, $\|B_k^{-1} g_k\| \leqslant \Delta_k$ and Algorithm 2.1 chooses the step $d_k = -B_k^{-1} g_k$ for all k.

Theorem 3.2. *Suppose that* (A1), (A2) *and* (A3) *hold, the sequence* $\{x_k\}$ *is generated by Algorithm* 2.1 *and converges to* x^*. *Also suppose* $\nabla^2 f(x)$ *is a Lipschitz continuous matrix in a neighborhood* $N(x^*, \varepsilon)$. *Moreover, assume that* $\nabla^2 f(x^*)$ *is positive definite such that*

$$\lim_{k\to\infty} \frac{\|(B_k - \nabla^2 f(x^*)) d_k\|}{\|d_k\|} = 0 \tag{3.20}$$

holds. Then the sequence $\{x_k\}$ *converges to* x^* *superlinearly.*

P r o o f. The proof is similar to Theorem 4.1 of [1] and here is omitted. □

4. Numerical results

In this section, the numerical experiments are divided into two groups to show the effectiveness of the proposed algorithm. In the first group of experiments, we utilize a set of 80 test functions mainly from [3], the dimension of each problem is set to 100, they were run on 3.60 GHz CPU processor (Intel(R) Xeon(R) CPU E5-1650), 64 GB RAM memory and Windows 7 operation system. The second group of experiments was performed on a set of 109 test problems from the CUTEr library [10] with dimensions 2 to 1000, the codes were run in Ubuntu 10.04 LTS which is fixed in VMware Workstation 10.0 installed in Windows 7.

We compare the AINATR method with the ANMTR method [18] and the RNATR method [16]. In the numerical experiments, the following parameters are used in the AINATR method:

$$\bar{\Delta} = 100,\ t = 0.3,\ u = 0.07,\ \gamma = 1.9,\ \tau = 10^{-2},\ B_0 = I,\ M_1 = 15,\ \eta_0 = 0.5,$$

we update the parameter η_k by

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

In order to maintain consistency, we solve the quadratic subproblem (1.3) by using the Steihaug-Toint scheme [6] (Page 205) in the considered algorithms. For all methods, the iteration is terminated if the gradient satisfies $\|g_k\|_\infty \leqslant 10^{-6}$ or the number of iterations exceeds 50000.

We adopt the performance profiles proposed by Dolan and Moré [8] to display the performance of the methods. Let P denote the set of n_p test problems and S be the set of all algorithms. For each problem p and solver s we define $t_{p,s}$ as computational time required to solve problem p by solver s, that is, the performance ratio is defined as

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s}\colon\ s \in S\}}. \tag{4.1}$$

It is obvious that $r_{p,s} \geqslant 1$ for all p and s. For each solver s, the performance profile is defined as the cumulative distribution function for performance ratio

$$P(\tau) = \frac{\text{size}\{p \in P\colon\ r_{p,s} \leqslant \tau\}}{n_p}, \tag{4.2}$$

that is, for each method, in the following figures, we plot the fraction $P(\tau)$ of problems for which the method is within a factor τ of the best time. Obviously, $P(1)$ represents

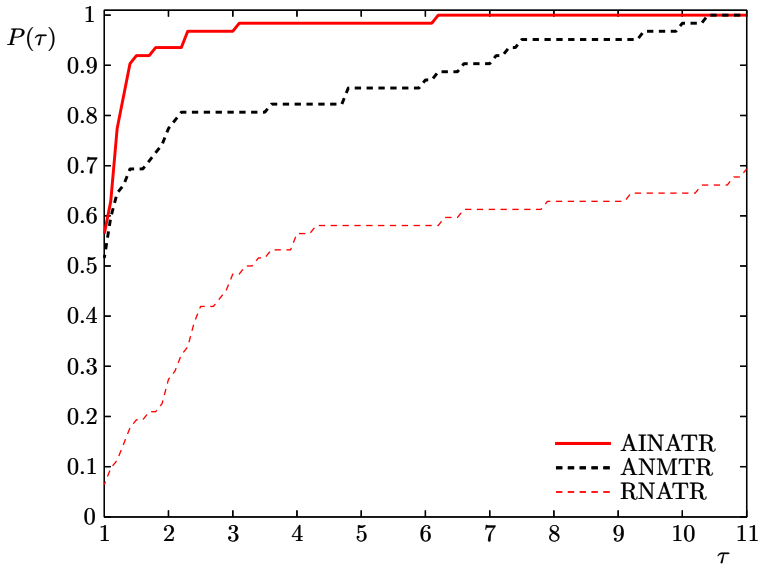

Figure 1. Performance profile based on N_{iter}(Pro_Andrei).

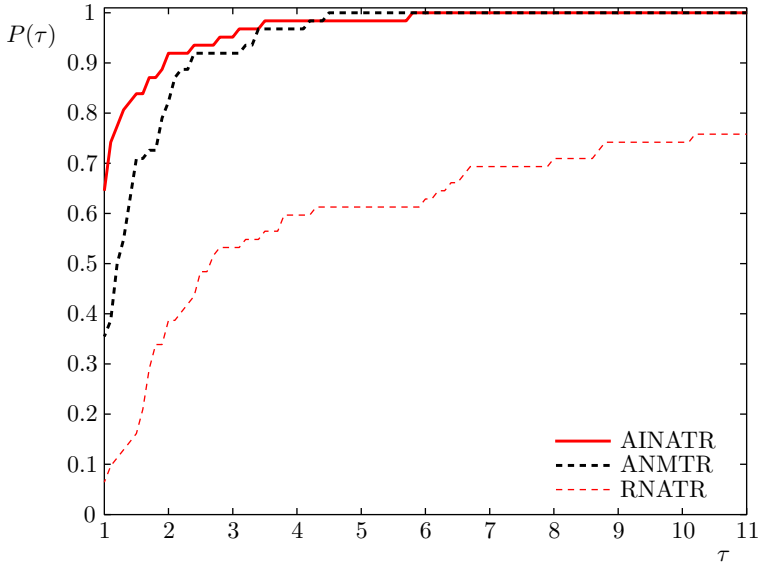

Figure 2. Performance profile based on N_f(Pro_Andrei).

the percentage of the test problems for which the method is the fastest. The top curve is the method that solved most problems in a time that was within the factor τ of the best time. See [8] for more details about the performance profile.

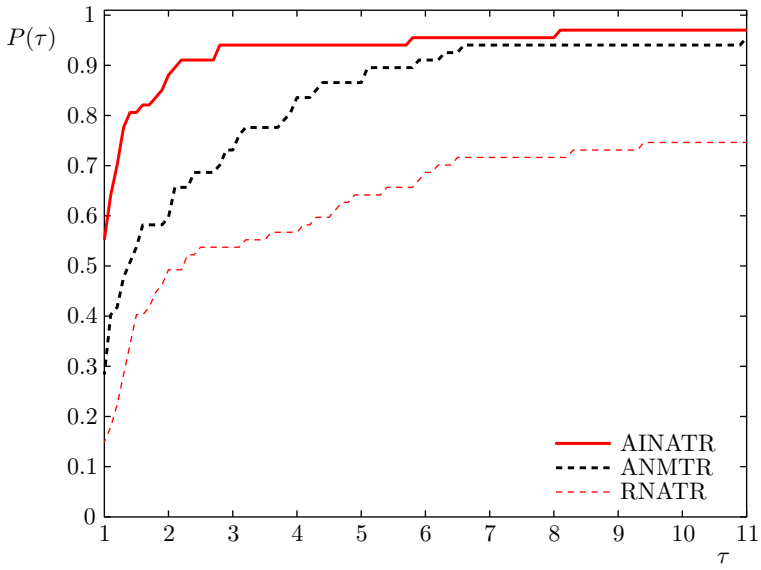

Figure 3. Performance profile based on T_{cpu}(Pro_Andrei).

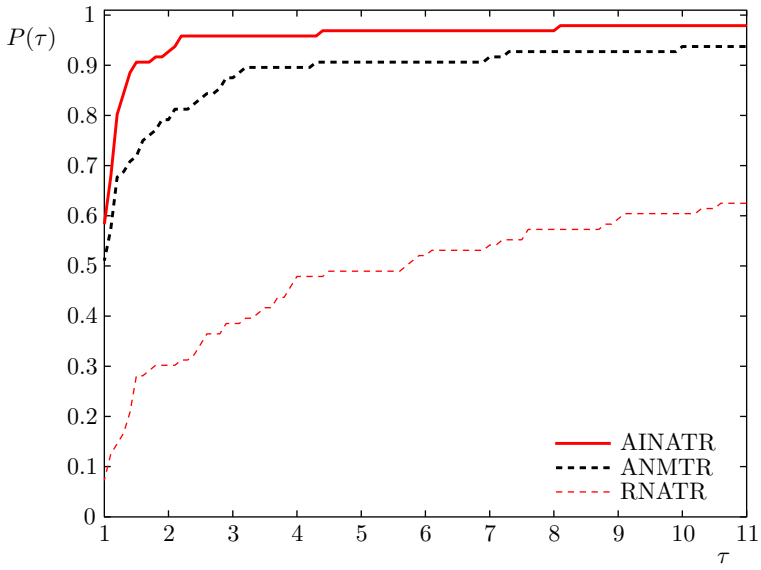

Figure 4. Performance profile based on N_{iter}(CUTEr).

Since the number of iterations and gradient evaluations are the same, the number of gradient evaluations will be discarded from the discussion below. In Figs. 1–6,

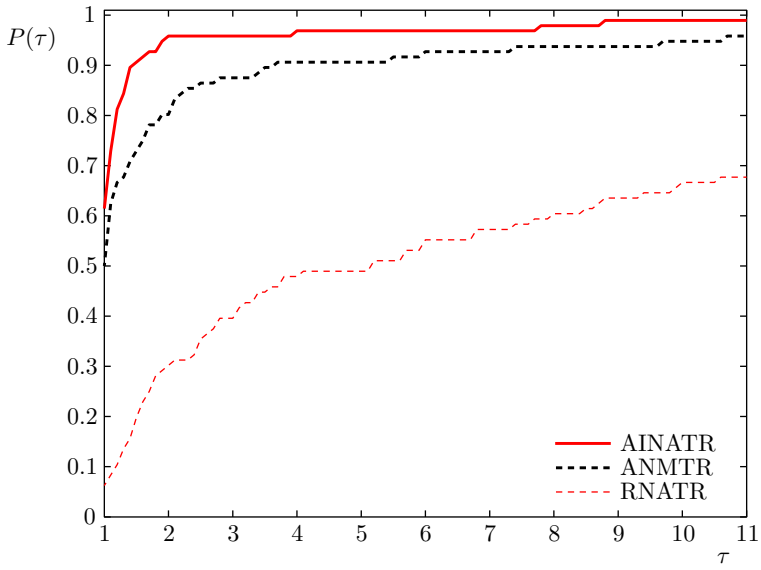

Figure 5. Performance profile based on N_f(CUTEr).

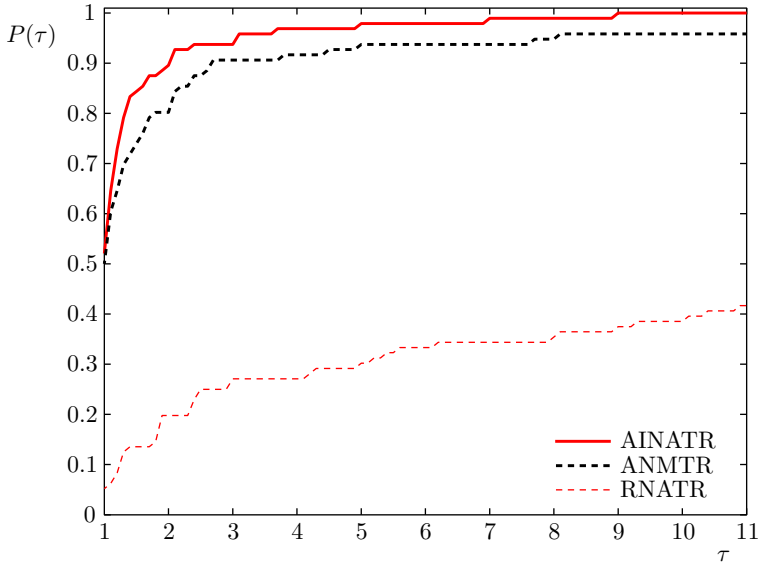

Figure 6. Performance profile based on T_{cpu}(CUTEr).

N_{iter}, N_f and T_{cpu} represent the number of iterations, the number of function evaluations and the CPU time, respectively.

In the first group of the numerical experiments, we compare the AINATR method with the ANMTR method and the RNATR method for 80pro_ Andrei. The AINATR method successfully solves 79 problems, while the ANMTR method and the RNATR method successfully solves 75 and 73 problems, respectively. From Fig. 1, we can easily see that the AINATR method is the best performing relative to the number of iterations among the three algorithms considered. In Fig. 2, we observe that the AINATR method is more effective than the ANMTR method and the RANTR method relative to the number of function evaluations for the case of $\tau \leqslant 4$. In Fig. 3, one can see that the AINATR method grows faster than the ANMTR method and the RNATR method. From Figs. 1, 2 and 3 we could say that the AINATR method is competitive with the other two related methods in terms of the test questions given.

In the second group of the numerical experiments, we discuss the performance of the AINATR method, the ANMTR method and the RNATR method for 109 test problems from the CUTEr library [10]. In the numerical experiments, the AINATR method successfully solves 108 test problems, while the ANMTR method successfully solves 106 problems and the RNATR method successfully solves 100 problems. As shown in Fig. 4, the AINATR method requires less iterations than the ANMTR method and the RNATR method. In Fig. 5, we observe that the AINATR method is more efficient than the ANMTR method and the RNATR method, and it successfully solves about 62% of test problems with the least number of function evaluations, while the percentages of solved problems of the ANMTR method and the RNATR method are 50% and 9%, respectively. Fig. 6 indicates that the AINATR method is faster than the ANMTR method and the RNATR method. From Figs. 4, 5, and 6 we can see that the AINATR method outperforms ANMTR and RNATR for the given test set.

5. Conclusions

In this paper, we propose an improved nonmonotone adaptive trust region method for solving unconstrained optimization problems. Approximating Hessian matrix by the scaled memoryless BFGS formula, an approximate model is constructed. Furthermore, the nonmonotone technique is employed in the adaptive trust region method in order to enhance the effectiveness of the algorithm. From the perspective of theoretical analysis, the proposed algorithm inherits the global convergence and the superlinear convergence rate of traditional trust region algorithms under classical assumptions. Finally, the effectiveness of the new proposed algorithm has been verified by experiments on two groups of standard test problems set.

References

[1] *M. Ahookhosh, K. Amini*: A nonmonotone trust region method with adaptive radius for unconstrained optimization problems. Comput. Math. Appl. *60* (2010), 411–422. zbl MR doi

[2] *M. Ahookhosh, K. Amini*: An efficient nonmonotone trust-region method for unconstrained optimization. Numer. Algorithms *59* (2012), 523–540. zbl MR doi

[3] *N. Andrei*: An unconstrained optimization test functions collection. Adv. Model. Optim. *10* (2008), 147–161. zbl MR

[4] *N. Andrei*: Accelerated scaled memoryless BFGS preconditioned conjugate gradient algorithm for unconstrained optimization. Eur. J. Oper. Res. *204* (2010), 410–420. zbl MR doi

[5] *R. M. Chamberlain, M. J. D. Powell, C. Lemarechal, H. C. Pedersen*: The watchdog technique for forcing convergence in algorithms for constrained optimization. Math. Program. Study *16* (1982), 1–17. zbl MR doi

[6] *A. R. Conn, N. I. M. Gould, P. L. Toint*: Trust-Region Methods. MPS/SIAM Series on Optimization, Society for Industrial and Applied Mathematics; Mathematical Programming Society, Philadelphia, 2000. zbl MR doi

[7] *N. Y. Deng, Y. Xiao, F. J. Zhou*: Nonmonotonic trust region algorithm. J. Optimization Theory Appl. *76* (1993), 259–285. zbl MR doi

[8] *E. D. Dolan, J. J. Moré*: Benchmarking optimization software with performance profiles. Math. Program. *91* (2002), 201–213. zbl MR doi

[9] *J.-Y. Fan, Y.-X. Yuan*: A new trust region algorithm with trust region radius converging to zero. Proceedings of the 5th International Conference on Optimization: Techniques and Applications. Hong Kong, 2001, pp. 786–794.

[10] *N. I. M. Gould, D. Orban, P. L. Toint*: CUTEr and SifDec: a constrained and unconstrained testing environment, revisited. ACM Trans. Math. Softw. *29* (2003), 373–394. zbl doi

[11] *L. Grippo, F. Lampariello, S. Lucidi*: A nonmonotone line search technique for Newton's method. SIAM J. Numer. Anal. *23* (1986), 707–716. zbl MR doi

[12] *A. Kamandi, K. Amini, M. Ahookhosh*: An improved adaptive trust-region algorithm. Optim. Lett. *11* (2017), 555–569. zbl MR doi

[13] *D. Li, M. Fukushima*: A modified BFGS method and its global convergence in nonconvex minimization. J. Comput. Appl. Math. *129* (2001), 15–35. zbl MR doi

[14] *N. Maratos*: Exact penalty function algorithms for finite dimensional and control optimization problems. Ph.D. Thesis, University of London, London, 1978.

[15] *D. W. Marquardt*: An algorithm for least-squares estimation of nonlinear parameters. J. Soc. Ind. Appl. Math. *11* (1963), 431–441. zbl MR doi

[16] *M. R. Peyghami, D. A. Tarzanagh*: A relaxed nonmonotone adaptive trust region method for solving unconstrained optimization problems. Comput. Optim. Appl. *61* (2015), 321–341. zbl MR doi

[17] *M. J. D. Powell*: A new algorithm for unconstrained optimization. Nonlinear Programming, Proceedings of a Symposium Conducted by the Mathematics Research Center. Academic Press, New York, 1970, pp. 31–65. zbl MR doi

[18] *S. Rezaee, S. Babaie-Kafaki*: An adaptive nonmonotone trust region algorithm. Optim. Methods Softw. *34* (2019), 264–277. zbl MR doi

[19] *Z.-J. Shi, J. Guo*: A new trust region method with adaptive radius. Comput. Optim. Appl. *41* (2008), 225–242. zbl MR doi

[20] *W. Sun*: Nonmonotone trust region method for solving optimization problems. Appl. Math. Comput. *156* (2004), 159–174. zbl MR doi

[21] *W. Sun, Y. Yuan*: Optimization Theory and Methods. Nonlinear Programming. Springer Optimization and Its Applications 1, Springer, New York, 2006. zbl MR doi

[22] *D. Winfield*: Function minimization by interpolation in a data table. J. Inst. Math. Appl. *12* (1973), 339–347. zbl MR doi

[23] *J.-L. Zhang, X.-S. Zhang*: A nonmonotone adaptive trust region method and its convergence. Comput. Math. Appl. *45* (2003), 1469–1477. zbl MR doi
[24] *X. Zhang, J. Zhang, L. Liao*: An adaptive trust region method and its convergence. Sci. China, Ser. A *45* (2002), 620–631. zbl MR doi

Authors' addresses: *Yanqin Xue*, *Hongwei Liu*, School of Mathematics and Statistics, Xidian University, Xi'an, Shaanxi, 710126, People's Republic of China, e-mail: `yanqin_xue@163.com`, `hwliu@mail.xidian.edu.cn`; *Zexian Liu*, School of Mathematics and Computer Science, Hezhou University, Hezhou, 542899, People's Republic of China, e-mail: `liuzexian2008@163.com`.