A NEW NONMONOTONE ADAPTIVE TRUST REGION ALGORITHM

Ahmad Kamandi, Behshahr, Keyvan Amini, Kermanshah

Received April 29, 2020. Published online April 30, 2021.

Abstract. We propose a new and efficient nonmonotone adaptive trust region algorithm to solve unconstrained optimization problems. This algorithm incorporates two novelties: it benefits from a radius dependent shrinkage parameter for adjusting the trust region radius that avoids undesirable directions and exploits a new strategy to prevent sudden increments of objective function values in nonmonotone trust region techniques. Global convergence of this algorithm is investigated under some mild conditions. Numerical experiments demonstrate the efficiency and robustness of the proposed algorithm in solving a collection of unconstrained optimization problems from the CUTEst package.

Keywords: unconstrained optimization; nonmonotone trust region; adaptive radius; global convergence; CUTEst package

MSC 2020: 90C30

1. Introduction

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 differentiable function. We are interested in the case when the number of variables is large. Despite the fact that the well-known trust region method is a well-documented framework [5], [15] in numerical optimization for solving the problem (1.1), its efficiency needs to be improved. The method itself or its variations are frequently required in tackling emerged problems in extensive recent applications [3], [4], [12], [18].

DOI: 10.21136/AM.2021.0122-20

In order to minimize $f(x)$, the trust region framework uses an approximation x_k of a local minimizer to compute a trial step direction d_k by solving the subproblem

$$\min_{\|d\| \leqslant \delta_k} m_k(d), \quad m_k(d) = f_k + g_k^\top d + \frac{1}{2} d^\top B_k d, \tag{1.2}$$

where $f_k = f(x_k)$, $g_k = \nabla f(x_k)$, δ_k is a positive parameter that is called the trust region radius and B_k is an approximation to the Hessian of the objective function at x_k. In the rest of the paper, $\|\cdot\|$ denotes the Euclidean norm.

Finding a global minimizer of subproblem (1.2) is often too expensive so that, in practice, numerical methods are applied to find an approximation [9], [14], [21]. Global convergence of the classic trust region algorithm is proved provided that the approximate solution d_k of subproblem (1.2) satisfies the following reduction estimation in the model function:

$$m_k(0) - m_k(d_k) \geqslant c \frac{1}{2} \|g_k\| \min \left\{ \delta_k, \frac{\|g_k\|}{\|B_k\|} \right\} \tag{1.3}$$

with $c \in (0, 1)$.

Given a fixed trial direction d_k, define the ratio r_k as

$$r_k := \frac{f_k - f(x_k + d_k)}{m_k(0) - m_k(d_k)}. \tag{1.4}$$

In classical trust region methods, the kth iteration is called a successful iteration if $r_k > \mu$ for some $\mu \in (0, 1)$. In this case, the trial point $x_k + d_k$ is accepted as a new approximation and the trust region radius is enlarged. Otherwise, the iteration k is called an unsuccessful iteration; the trial point is rejected and the trust region is shrunk. The efficiency of trust region methods strongly relies on the generated sequence of radii. A large radius possibly increases the cost of solving the corresponding subproblem and a small radius increases the number of iterations. Hence, choosing an appropriate radius in each iteration is challenging in trust region methods. In the effort to tackle this challenge, many authors have rigorously studied the adaptive trust region methods [1], [8], [13], [19], [23].

Zhang et al. in [23] proposed the adaptive radius

$$\delta_k = c^{p_k} \|g_k\| \, \|\widehat{B}_k^{-1}\|, \tag{1.5}$$

where $c \in (0, 1)$, p_k is a nonnegative integer and $\widehat{B}_k = B_k + E_k$ is a safely positive definite matrix based on a modified Cholesky factorization by Schnabel and Eskow in [17]. Numerical results indicate that embedding this adaptive radius in a pure trust

region increases the efficiency. But the formula (1.5) requires to calculate the inverse matrix $\widehat{B}_k^{-1}$ at each iteration and thus it is not suitable for large-scale problems. Shi and Guo in [19] proposed another adaptive radius

$$\delta_k = -c^{p_k} \frac{g_k^\top q_k}{q_k^\top \widetilde{B}_k q_k} \|q_k\|, \tag{1.6}$$

where $c \in (0,1)$, p_k is a nonnegative integer and q_k is a vector satisfying

$$-\frac{g_k^\top q_k}{\|g_k\| \cdot \|q_k\|} \geqslant \tau \tag{1.7}$$

with $\tau \in (0,1]$. Moreover, $\widetilde{B}_k$ is generated by the procedure: $\widetilde{B}_k = B_k + iI$, where i is the smallest nonnegative integer such that $q_k^\top \widetilde{B}_k q_k > 0$. It is simple to see that the radius (1.6), for $p_k = 0$, estimates norm of the exact minimizer of the quadratic model $f_k + g_k^\top d + \frac{1}{2} d^\top \widetilde{B}_k d$ along the direction q_k.

Motivated by this adaptive radius, Kamandi et al. proposed an efficient adaptive trust region method in which the radius at each iteration is determined by using the information gathered from the previous step [13]. Let d_{k-1} be the solution of the subproblem in the previous step, for parameters $\tau \in (0,1)$ and $\gamma > 1$ define

$$q_k := \begin{cases} -g_k & \text{if } k = 0 \text{ or } \dfrac{-(g_k^\top d_{k-1})}{\|g_k\|\|d_{k-1}\|} \leqslant \tau, \\ d_{k-1} & \text{o.w.} \end{cases} \tag{1.8}$$

and

$$s_k := \begin{cases} -\dfrac{g_k^\top q_k}{q_k^\top B_k q_k} \|q_k\| & \text{if } k = 0, \\ \max\Big\{ -\dfrac{g_k^\top q_k}{q_k^\top B_k q_k} \|q_k\|,\ \gamma \delta_{k-1} \Big\} & \text{if } k \geqslant 1. \end{cases} \tag{1.9}$$

Then, the algorithm proposed in [13] for solving (1.1) is as follows:

Algorithm: (IATR) Improved adaptive trust-region algorithm

---

input: $x_0 \in \mathbb{R}^n$, a positive definite matrix $B_0 \in \mathbb{R}^{n \times n}$, $\bar{\delta} > 0$,
$c, \mu \in (0,1)$, $\tau \in (0,1)$, $\gamma \geqslant 1$ and $\varepsilon > 0$.

begin

$k \leftarrow 0$, compute f_0 and g_0.

while $(\|g_k\| > \varepsilon)$

Compute q_k by (1.8) and s_k by (1.9),

set $\delta_{k_0} = \min\{s_k, \bar{\delta}\}$,

$r_k \leftarrow 0$, $p \leftarrow 0$.
while $(r_k < \mu)$
$\delta_{k_p} \leftarrow c^p \delta_{k_0}$,
compute d_{k_p} by solving (1.2) with radius δ_{k_p};
compute r_k by (1.4),
$p \leftarrow p + 1$.
end while
$x_{k+1} \leftarrow x_k + d_{k_p}$,
update B_k by a quasi-Newton formula,
$k \leftarrow k + 1$.
end while
end

Despite it enjoys many advantages [13], this algorithm has several disadvantages. First, setting a fixed value for the shrinkage parameter c in the inner loop of the IATR algorithm is not an intelligent choice. In order to see this, suppose that the step direction d_{k_0}, the solution of the subproblem (1.2) with the radius δ_{k_0}, is rejected by the ratio test. In this case, the algorithm shrinks the radius δ_{k_0} by the factor $c \in (0, 1)$. Hence, we have the new subproblem

$$\min_{\|d\| \leqslant c\delta_{k_0}} m_k(d), \quad m_k(d) = f_k + g_k^\top d + \frac{1}{2} d^\top B_k d. \tag{1.10}$$

Since $c\delta_{k_0} < \delta_{k_0}$, it is clear that the feasible region of the subproblem (1.10) is a subset of the feasible region of the subproblem (1.2). So, in case that $\|d_{k_0}\| \leqslant c\delta_{k_0}$, d_{k_0} is also a solution of (1.10), although we know that it is rejected by the ratio test. This means that the new step direction is rejected by the ratio test again without any improvement; solving the new subproblem has redundant computational costs, though.

Another drawback of a constant shrinkage parameter occurs when the trust region radius is too large and the shrinkage parameter is close to one: the algorithm is forced to solve the trust region subproblem several times until it finds a successful step. So, using a shrinkage parameter close to one may increase the number of function evaluations. On the other hand, using a small shrinkage parameter may cause to shrink the trust radius too fast; in this case, the number of iterations increases.

Furthermore, the sequence of function evaluations generated by this algorithm is decreasing and numerical results show that imposing monotonicity to trust region algorithms may reduce the speed of convergence for some problems, specially in the presence of a narrow valley. In order to overcome similar drawbacks, Grippo et al. proposed a nonmonotone line search technique for Newton's method [11]. By

generalizing the technique to the trust region methods, the nonmonotone version of these methods appeared in the literature [1], [2], [6], [16], [20], [24].

The basic difference between the monotone and nonmonotone trust region approaches is due to the definition of the ratio r_k. In a nonmonotone trust region, the ratio is defined by

$$\hat{r}_k = \frac{C_k - f(x_k + d_k)}{m_k(0) - m_k(d_k)}, \tag{1.11}$$

where C_k is a parameter greater than or equal to f_k. In this paper, we call C_k the nonmonotone parameter. In different versions of nonmonotone algorithms, the nonmonotone parameter computation is based on different methodologies. A common parameter for nonmonotone trust region methods is

$$f_{l_k} := \max_{0 \leqslant j \leqslant N_k} f_{k-j}, \tag{1.12}$$

where $N_0 = 0$ and $N_k = \min\{k, N\}$ for a fixed integer number $N \geqslant 0$.

Note that by taking maximum in the parameter (1.12), a potentially very good function value can be excluded. Trying to tackle this drawback, Ahookhosh et al. in [2] proposed the nonmonotone parameter

$$R_k = \eta_k f_{l_k} + (1 - \eta_k) f_k, \tag{1.13}$$

where $\eta_k \in [\eta_{\min}, \eta_{\max}]$, $\eta_{\min} \in [0, 1)$ and $\eta_{\max} \in [\eta_{\min}, 1]$. When η_k is close to one the effect of nonmonotonicity is amplified. On the other hand, when η_k is close to zero the algorithm ignores the effect of the term f_{l_k} and behaves monotonically.

In 2019, Xue et al. proposed a nonmonotone version of the IATR algorithm based on the nonmonotone parameter (1.13), see [22]. They also used a scaled memoryless BFGS formula to update the approximation of the Hessian matrix. By analyzing the numerical behavior of nonmonotone versions of IATR using the aforementioned nonmonotone parameters, we find out that in some problems, for example OSCIGRAD, the difference between the current objective value f_k and the nonmonotone parameter becomes too large and in this case a large increase is allowed to happen in the next iteration. Another drawback of the above nonmonotone parameters is that they strongly depend on the choice of the memory parameter N_k and the parameter η_k, and there is no specific rule to adjust them.

In this paper, by combining the idea of adaptive trust region and nonmonotone techniques, we propose a new efficient nonmonotone trust region algorithm for solving unconstrained optimization problems. In the new algorithm, a radius dependent shrinkage parameter is used to adjust the trust region radius in rejected steps which

addresses the first disadvantage of IATR. For resolving the second disadvantage, a novel strategy is used to compute the nonmonotone parameter in this algorithm which prevents a sudden increment in the objective values.

The paper is organized as follows: the new algorithm is proposed in the next section. Section 3 is devoted to its convergence properties. The numerical results of testing the new algorithm to solve a collection of the CUTEst test problems are reported in Section 4. The last section includes the conclusion.

2. The new algorithm

In this section, we propose our algorithm for solving unconstrained optimization problems.

As mentioned in the previous section, setting a fixed value for the shrinkage parameter c in the inner loop of the IATR algorithm may impose some useless computational costs to this algorithm. Therefore, for resolving this issue, we propose the radius dependent shrinkage parameter

$$c_{k_p} := c(\delta_{k_p}), \tag{2.1}$$

where $c(\delta) : (0, \bar{\delta}] \to [\alpha_0, \alpha_1]$ is a decreasing function where $0 < \alpha_0 < \alpha_1 < 1$ and $\bar{\delta}$ is the maximum possible radius. Also, in order to exclude the rejected trial step d_{k_p}, in the new algorithm we define the new radius as $\delta_{k_{p+1}} = c_{k_p}\|d_{k_p}\|$.

Note that the radius dependent parameter (2.1) is close to α_0 for a large trust region radius and is close to α_1 for a small one. Hence, this parameter shrinks the trust region harshly for large trust region radii and helps the new algorithm to find a successful step direction fast enough. Further, it shrinks the trust region mildly in the case that the trust region radius is small.

Also numerical tests persuaded us to consider a radius dependent parameter $\gamma_k = \gamma(\delta_{k-1})$ based on the previous trust region radius and use it instead of the constant parameter γ in (1.9). Similar to (2.1), $\gamma(\delta_{k-1})$ is a decreasing function bounded from below by 1.

With the goal of overcoming the second disadvantage of the IATR algorithm and building an efficient nonmonotone version of it, we propose a new nonmonotone parameter C_k. This new parameter benefits from nonmonotonicity in an adaptive way compared to the mentioned parameters. When a very good function value is found at the iteration k, it is better to save that by forcing the algorithm to behave monotonically for the next iteration. To this aim, we define a new nonmonotone parameter using not only the simple parameter f_{l_k} defined by (1.12) but also considering its relative difference from the current function value.

For a positive parameter ν, define sequences $\{M_k\}$ and $\{I_k\}$ as

$$M_k := \begin{cases} 0 & \text{if } k = 0 \text{ or } f_{l_k} - f_k > \nu |f_k|, \\ M_{k-1} + 1 & \text{o.w.} \end{cases}$$

and

$$I_k := \begin{cases} 0 & \text{if } k = 0 \text{ or } f_k < f_{k-1}, \\ I_{k-1} + 1 & \text{o.w.} \end{cases}$$

Having the above sequences for fixed natural numbers $\overline{N}$ and $\overline{I}$, we define the new nonmonotone parameter C_k as

$$C_k := \begin{cases} \max\limits_{0 \leqslant j \leqslant n_k} f_{k-j} & \text{if } I_k \leqslant \overline{I}, \\ f_k & \text{o.w.} \end{cases} \tag{2.2}$$

where $n_k = \min\{M_k, \overline{N}\}$. Note that the sequence $\{I_k\}$ counts the number of consecutive increments in the objective function values. So, the nonmonotone parameter C_k defined by (2.2) prevents large increments in the objective function values and guarantees at least one decrease for each $\overline{I}$th iteration. Also, the definition of the sequence $\{M_k\}$ makes the new algorithm monotone when the relative difference between f_{l_k} and the current function value is large and prevents a sudden increment in the objective function values for the next iteration.

Now, we are ready to propose the new adaptive nonmonotone trust region algorithm.

Algorithm: (NATR) Nonmonotone adaptive trust-region algorithm

---

input: $x_0 \in \mathbb{R}^n$, a positive definite matrix $B_0 \in \mathbb{R}^{n \times n}$, $\bar{\delta} > 0$,
a decreasing function $c(\delta)$, $\mu \in (0, 1)$, $\tau \in (0, 1)$, $\gamma \geqslant 1$, and $\varepsilon > 0$.

begin

$k \leftarrow 0$; compute f_0 and g_0.

while $(\|g_k\| > \varepsilon)$

Compute q_k by (1.8) and s_k by (1.9),
set $\delta_{k_0} = \min\{s_k, \bar{\delta}\}$,
compute C_k by (2.2),
compute d_{k_0} by solving (1.2) with radius δ_{k_0} and $\hat{r}_k$ by (1.11)
and set $p = 0$.

while $(\hat{r}_k < \mu)$

Compute c_{k_p} by (2.1),
$\delta_{k_{p+1}} = c_{k_p} \|d_{k_p}\|$,
compute $d_{k_{p+1}}$ by solving (1.2) with radius $\delta_{k_{p+1}}$ and $\hat{r}_k$ by (1.11),

$p \leftarrow p + 1$.
end while
$x_{k+1} \leftarrow x_k + d_{k_p}$,
update B_k by a quasi-Newton formula,
$k \leftarrow k + 1$.
end while
end

In the next section, we propose the convergence properties of the new algorithm.

3. Convergence properties

In this section, we analyze the global convergence of the new algorithm. To this end, we need the following assumptions:

(H1) The objective function $f(x)$ is continuously differentiable and has a lower bound on the level set

$$\mathcal{L}(x_0) = \{x \in \mathbb{R}^n;\ f(x) \leqslant f(x_0),\ x_0 \in \mathbb{R}^n\}.$$

(H2) The approximation matrix B_k is uniformly bounded, i.e., there exists a constant $M > 0$ such that

$$\|B_k\| \leqslant M \quad \forall\, k \in \mathbb{N}.$$

The following lemma is similar for both the IATR and NATR algorithms, so its proof is omitted.

Lemma 3.1. *Suppose that the sequence $\{x_k\}$ is generated by the NATR algorithm. Then*

$$|f(x_k + d_{k_p}) - m_k(d_{k_p}))| = o(\|d_{k_p}\|).$$

Proof. See [15]. □

The next two lemmas guarantee the existence of some lower bounds for the trust region radius δ_{k_0} and the norm of the trial step d_{k_0} at the iteration k generated by the NATR algorithm.

Lemma 3.2. *Suppose that $\delta_{k_0} = \min\{s_k, \bar{\delta}\}$ is the trust region radius at the iteration k of the NATR algorithm such that s_k is defined by* (1.9). *Then*

$$\delta_{k_0} \geqslant \min\Big\{\tau \frac{\|g_k\|}{\|B_k\|}, \bar{\delta}\Big\}. \tag{3.1}$$

P r o o f. In case $\bar{\delta} \leqslant s_k$, we have $\delta_{k_0} = \bar{\delta}$ and the inequality (3.1) is valid. Thus, consider the case that $s_k < \bar{\delta}$ and $\delta_{k_0} = s_k$. The definition of s_k in (1.9) and the Cauchy-Schwarz inequality yield that

$$\delta_{k_0} \geqslant \frac{-g_k^\top q_k}{\|B_k\|\|q_k\|}. \tag{3.2}$$

By the definition of q_k in (1.8) if $q_k = -g_k$, inequality (3.2) results in

$$\delta_{k_0} \geqslant \frac{\|g_k\|}{\|B_k\|}.$$

When $q_k = d_{k-1}$, we have $-g_k^\top q_k \geqslant \tau\|g_k\|\|q_k\|$ so that inequality (3.2) implies

$$\delta_{k_0} \geqslant \tau\frac{\|g_k\|}{\|B_k\|}.$$

By the above explanation along with the fact that $\tau \in (0, 1)$ we can conclude that (3.1) is valid. □

Lemma 3.3. *Suppose that d_{k_0} is the solution of the subproblem* (1.2) *with radius δ_{k_0}. Then*

$$\|d_{k_0}\| \geqslant \min\Big\{\frac{\|g_k\|}{\|B_k\|}, \delta_{k_0}\Big\}. \tag{3.3}$$

P r o o f. By Theorem 4.1 of [15], when d_{k_0} lies strictly inside the feasible region of subproblem (1.2), we must have $B_k d_{k_0} = -g_k$ such that the Cauchy-Schwarz inequality yields

$$\|d_{k_0}\| \geqslant \frac{\|g_k\|}{\|B_k\|}.$$

In the other case d_{k_0} lies on the boundary of the feasible region of the subproblem (1.2) which implies $\|d_{k_0}\| = \delta_{k_0}$. So, from the above discussion we can conclude that (3.3) is valid. □

By (3.1) and (3.3), we can also obtain a lower bound for δ_{k_p}. Note that, at the iteration k of the NATR algorithm, for any $p \geqslant 1$ the solution d_{k_p} lies on the boundary of the region defined by δ_{k_p}. Since the objective is fixed for each iteration, when the trial step d_{k_p} is rejected by the ratio test, the new radius $\delta_{k_{p+1}}$ is set to

exclude d_{k_p} from the new region. Thus, by the contraction of the inner loop of the NATR algorithm, we have

$$\begin{aligned}\delta_{k_p} &= c_{k_{p-1}}\|d_{k_{p-1}}\| = c_{k_{p-1}}\delta_{k_{p-1}}\\ &= c_{k_{p-1}}c_{k_{p-2}}\|d_{k_{p-2}}\| = c_{k_{p-1}}c_{k_{p-2}}\delta_{k_{p-2}}\\ &\vdots\\ &= \prod_{i=0}^{p-1} c_{k_i}\|d_{k_0}\|.\end{aligned}$$

This equation along with (3.1), (3.3) and the fact that α_0 is a lower bound and α_1 an upper bound for c_{k_i} for any $i \geqslant 0$ yield that

$$\alpha_0^p \min\Big\{\tau\frac{\|g_k\|}{\|B_k\|}, \bar{\delta}\Big\} \leqslant \delta_{k_p} \leqslant \alpha_1^p\bar{\delta}. \tag{3.4}$$

In Lemma 3.4, we propose a lower bound for the denominator of the ratio $\hat{r}_k$ defined by (1.11) which is used in Lemma 3.5 to prove that the inner loop of the NATR algorithm terminates in a finite number of inner iterations.

Lemma 3.4. *Suppose that* (H2) *holds, the sequence* $\{x_k\}$ *is generated by the NATR algorithm, and* d_{k_p} *is an approximate solution of subproblem* (1.2) *with radius* δ_{k_p}, *that satisfies* (1.3). *Then,*

$$m_k(0) - m_k(d_{k_p}) \geqslant \frac{1}{2}c\alpha_0^p\|g_k\| \min\Big\{\tau\frac{\|g_k\|}{M}, \bar{\delta}\Big\} \quad \forall\, k \in \mathbb{N}. \tag{3.5}$$

P r o o f. By (1.3), for d_{k_p} we have

$$m_k(0) - m_k(d_{k_p}) \geqslant c\frac{1}{2}\|g_k\| \min\Big\{\delta_{k_p}, \frac{\|g_k\|}{\|B_k\|}\Big\}.$$

This inequality along with the assumption (H2) and inequality (3.4) result in

$$m_k(0) - m_k(d_{k_p}) \geqslant \frac{1}{2}c\alpha_0^p\|g_k\| \min\Big\{\tau\frac{\|g_k\|}{M}, \bar{\delta}\Big\}.$$

So, the proof is completed. □

Lemma 3.5. *Suppose that the assumption* (H2) *holds. Then the inner loop of the NATR algorithm is well-defined.*

P r o o f. By contradiction, assume that the inner loop of the NATR algorithm at the iteration k is not well-defined. Since x_k is not the optimum, $\|g_k\| \geqslant \varepsilon$.

Now, let d_{k_p} be the solution of subproblem (1.2) corresponding to $p \in \mathbb{N} \cup \{0\}$ at x_k. It follows from Lemma 3.1 and (3.5) that

$$\begin{aligned}\Big|\frac{f(x_k) - f(x_k + d_{k_p})}{m_k(0) - m_k(d_{k_p})} - 1\Big| &= \Big|\frac{f(x_k) - f(x_k + d_{k_p}) - (m_k(0) - m_k(d_{k_p}))}{m_k(0) - m_k(d_{k_p})}\Big| \\ &\leqslant \frac{o(\|d_{k_p}\|)}{\frac{1}{2}c\alpha_0^p\|g_k\| \min\{\tau\|g_k\|/M, \bar{\delta}\}} \\ &\leqslant \frac{o(\|d_{k_p}\|)}{\frac{1}{2}c\alpha_0^p\varepsilon \min\{\tau\varepsilon/M, \bar{\delta}\}}.\end{aligned}$$

By (3.4), we have $\delta_{k_p} \leqslant \alpha_1^p\bar{\delta}$. So, if the inner loop of the NATR algorithm cycles infinitely many times (or $p \to \infty$), then δ_{k_p} tends to zero. Thus, the feasibility of d_{k_p}, $\|d_{k_p}\| \leqslant \delta_{k_p}$, implies that the right-hand side of the above equation tends to zero. This means that for a sufficiently large p, we get

$$\frac{f(x_k) - f(x_k + d_{k_p})}{m_k(0) - m_k(d_{k_p})} \geqslant \mu.$$

This inequality along with the fact that $C_k \geqslant f_k$ yield that

$$\hat{r}_k = \frac{C_k - f(x_k + d_{k_p})}{m_k(0) - m_k(d_{k_p})} \geqslant \mu,$$

which means that the inner cycle of the NATR algorithm is terminated in the finite number of internal iterations. □

The following lemma illustrates some properties of the sequences $\{x_k\}$ and $\{C_k\}$, generated by the NATR algorithm. The statement of this lemma is used to prove the global convergence of the NATR algorithm.

Lemma 3.6. *Suppose that the assumption* (H1) *holds and the sequence* $\{x_k\}$ *is generated by the NATR algorithm. Then* $f_{k+1} \leqslant C_{k+1} \leqslant C_k$. *Therefore, the sequence* $\{x_k\}$ *is contained in the level set* $\mathcal{L}(x_0)$ *and the sequence* $\{C_k\}$ *is convergent.*

P r o o f. By the NATR algorithm, at the successful iteration k, we have

$$C_k - f_{k+1} \geqslant \mu(m_k(0) - m_k(d_{k_p})) \geqslant 0.$$

This inequality along with (2.2) and the definition of the sequence $\{M_k\}$ imply that

$$C_{k+1} = \max_{0 \leqslant j \leqslant n_{k+1}} \{f_{k+1-j}\} \leqslant \max\{f_{k+1}, C_k\} = C_k.$$

Thus

$$f_{k+1} \leqslant C_{k+1} \leqslant C_k \leqslant C_0 = f_0. \tag{3.6}$$

The last equation means that $\{x_k\}$ is contained in the level set $\mathcal{L}(x_0)$. Accordingly, the assumption (H1) and (3.6) yield that $\{C_k\}$ is decreasing and bounded from below. Therefore, the sequence $\{C_k\}$ is convergent. □

Now, we are ready to present the global convergence theorem.

Theorem 3.1. *Suppose that the assumptions* (H1) *and* (H2) *hold. Then the NATR algorithm either terminates in a finite number of steps, or generates an infinite sequence* $\{x_k\}$ *such that*

$$\liminf_{k\to\infty} \|g_k\| = 0. \tag{3.7}$$

P r o o f. If the NATR algorithm terminates in a finite number of steps, then the proof is trivial. Hence, assuming that the sequence $\{x_k\}$ generated by this algorithm is infinite, we show that (3.7) holds. To this end, suppose that there exists a constant $\varepsilon_0 > 0$ such that

$$\|g_k\| \geqslant \varepsilon_0 \tag{3.8}$$

for all k. Let $C_k = f_{i_k}$, where $f_{i_k} = \arg\max\{\max_{0\leqslant j\leqslant n_k}\{f_{k-j}\}\}$. Then, by Lemma 3.6, the sequence $\{f_{i_k}\}$ is a convergent subsequence of $\{f_k\}$. By the fact that $\hat{r}_k \geqslant \mu$, we have

$$f_{i_k} - f_{k+1} \geqslant \mu(m_k(0) - m_k(d_{k_p})).$$

Next, by replacing k with $i_k - 1$, we conclude that

$$f_{i_{i_k-1}} - f_{i_k} \geqslant \mu(m_{i_k-1}(0) - m_{i_k-1}(d_{(i_k-1)_p})).$$

This inequality along with Lemma 3.4 yield that

$$f_{i_{i_k-1}} - f_{i_k} \geqslant \mu\Big[\frac{1}{2}c\alpha_0^p\|g_{i_k-1}\| \min\Big\{\tau\frac{\|g_{i_k-1}\|}{M}, \bar{\delta}\Big\}\Big] \geqslant \mu\Big[\frac{1}{2}c\alpha_0^p\varepsilon_0 \min\Big\{\tau\frac{\varepsilon_0}{M}, \bar{\delta}\Big\}\Big].$$

Taking limit in this inequality when $k \to \infty$ implies that

$$0 \geqslant \mu\Big[\frac{1}{2}c\alpha_0^p\varepsilon_0 \min\Big\{\tau\frac{\varepsilon_0}{M}, \bar{\delta}\Big\}\Big],$$

which is a contradiction. Thus, equation (3.7) is valid. □

4. Numerical results

In this part of the paper, we report some numerical experiments that indicate the efficiency of the proposed algorithm. The results have been obtained by implementing two versions of the NATR algorithm and the adaptive nonmonotone algorithm proposed by Xue et al. [22] in MATLAB environment on a laptop (CPU Corei7-2.5 GHz, RAM 12 GB) and comparing the results of solving a collection of 228 unconstrained optimization test problems from the CUTEst collection [10]. The test problems and their dimensions are listed in Table 1.

In this section, we use the following notations:

- ▷ AINTR: The adaptive nonmonotone algorithm proposed by Xue et al. [22].
- ▷ NATR1: Nonmnotone adaptive trust region method (the NATR algorithm) based on the modified BFGS update formula used in [13].
- ▷ NATR2: Nonmnotone adaptive trust region method (the NATR algorithm) based on the scaled memoryless BFGS update formula used in [22].

For the NATR algorithm we used the following radius dependent parameters:

$$\gamma(\delta)=\begin{cases}1.5 & \text{if } \dfrac{\bar\delta}{2}<\delta\leqslant\bar\delta,\\ 1.9 & \text{if } \dfrac{\bar\delta}{5}<\delta\leqslant\dfrac{\bar\delta}{2},\\ 2 & \text{if } \dfrac{\bar\delta}{10}<\delta\leqslant\dfrac{\bar\delta}{5},\\ 3 & \text{if } 10^{-6}<\delta\leqslant\dfrac{\bar\delta}{10},\\ 3.5 & \text{o.w.}\end{cases}\qquad\text{and}\quad c(\delta)=\begin{cases}0.3 & \text{if } \dfrac{\bar\delta}{10}<\delta\leqslant\bar\delta,\\ 0.45 & \text{if } 10^{-6}<\delta\leqslant\dfrac{\bar\delta}{10},\\ 0.6 & \text{o.w.}\end{cases}$$

similar to [22], the other parameters are chosen as $\tau=0.01$, $N=15$, $\mu=0.07$, $\bar\delta=100$, $\varepsilon=10^{-6}\|g_0\|$ and for the NATR algorithm the remaining parameters are selected as $\overline{N}=10$, $\overline{I}=6$ and $\nu=10$. The trust region subproblems are solved by the Steihaug-Toint scheme [21].

To visualize the whole behaviour of the algorithms, we use the performance profiles proposed by Dolan and More [7]. The results of 14 test problems (the red ones in the table) are excluded from comparison because all the tested algorithms failed to solve them. So, the comparison of the algorithms is based on the remaining 214 test problems. Among these 214 test problems, NATR1, NATR2 and AINTR faced with 9, 42 and 49 failures, respectively.

The total number of function evaluations, the total number of iterations and the running time of each algorithm are considered as performance indexes. Note that at

Problem name	Dim	Problem name	Dim
ARGLINA	100, 200	ARGLINB	100, 200
ARGLINC	100, 200	BDQRTIC	100, 500, 1000, 5000
BROWNAL	100, 200, 1000	BRYBND	100, 500
CHAINWOO	100	CURLY10	100
CURLY20	100	CURLY30	100
EIGENALS	110, **2550**	EIGENBLS	110, **2550**
EIGENCLS	462, **2652**	EXTROSNB	100,1000
FREUROTH	100, 500, 1000, 5000	GENROSE	100, 500
LIARWHD	100, 500, 1000, 5000	MANCINO	100
MODBEALE	200, 2000	MSQRTALS	100, 529
MSQRTBLS	100, 529	NONDIA	100, 500, 1000, 5000
NONSCOMP	100, 500, 1000, 5000	OSCIGRAD	100, 1000
OSCIPATH	100, 500	PENALTY1	100, 500, 1000
PENALTY2	100, 200	SENSORS	100, 1000
SPMSRTLS	100, 499, 1000, 4999	SROSENBR	100, 500, 1000, 5000
SSBRYBND	100	TQUARTIC	100, 500, 1000, 5000
VAREIGVL	100, 500, 1000, 5000	WOODS	100, 1000, 4000
ARWHEAD	100, 500, 1000, 5000	BOX	100, 1000
BOXPOWER	100, 1000	COSINE	100, 1000
CRAGGLVY	100, 500, 1000, 5000	TESTQUAD	**1000**, **5000**
DIXMAANA	300, 1500, 3000	DIXMAANC	300, 1500, 3000
DIXMAAND	300, 1500, 3000	DIXMAANE	300, 1500, 3000
DIXMAANF	300, 1500, 3000	DIXMAANG	300, 1500, 3000
DIXMAANH	300, 1500, 3000	DIXMAANI	300, 1500, 3000
DIXMAANJ	300, 1500, 3000	DIXMAANK	300, 1500, 3000
DIXMAANL	300, 1500, 3000	DIXMAANM	300, 1500, 3000
DIXMAANN	300, 1500, 3000	DIXMAANO	300, 1500, 3000
DIXMAANP	300, 1500, 3000	DQRTIC	100, 500, 1000, 5000
EDENSCH	2000	ENGVAL1	100,1000, 5000
FLETCBV2	100	FLETCHCR	100, **1000**
FMINSRF2	121, 961, 1024	FMINSURF	121, 961, 1024
INDEFM	100, 1000, 5000	NCB20	110, 1010
NONCVXU2	100, 1000, **5000**	NONCVXUN	100, 1000, **5000**
NONDQUAR	100, 500, 1000, 5000	PENALTY3	100

Table 1. List of test problems.

POWELLSG	100, 500, 1000, 5000	POWER	100, 500, 1000, 5000
QUARTC	100, 500, 1000, 5000	SCHMVETT	100, 500, 1000, 5000
NCB20B	100,180,500,1000,2000	SPARSINE	100, 1000, 5000
SPARSQUR	100, 1000, 5000	TOINTGSS	100, 500, **1000**, **5000**
VARDIM	100, 200	DIXON3DQ	100, 1000
DQDRTIC	100, 500, 1000, 5000	TRIDIA	100, 500, 1000, 5000
BROYDN7D	100,500,1000,5000	SINQUAD	**100**, **500**, **1000**, **5000**

Table 1. (continued).

each iteration of the considered algorithms, the gradient of the objective function is computed just once, so the total number of iterations and the total number of the gradient evaluations are the same. Figure 1 illustrates the performance profile of these algorithms, where the performance index is the total number of function evaluations. It can be seen that the NATR1 is the best solver with the probability around 55%, while the probability of solving a problem as the best solver is around 42% and 30% for NATR2 and AINTR, respectively.

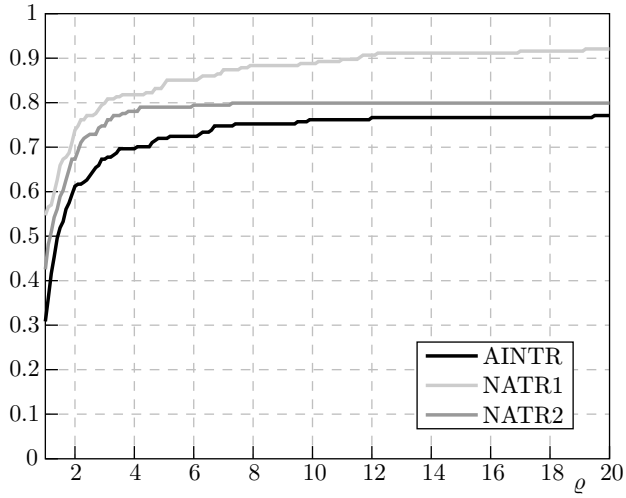

Figure 1. Performance profiles for the number of function evaluations.

The performance index in Figure 2 is the total number of iterations. From this figure, we observe that NATR1 obtains the most wins on approximately 58% of all test problems and the probability of being the best solver is 41% and 29% for NATR2 and AINTR, respectively.

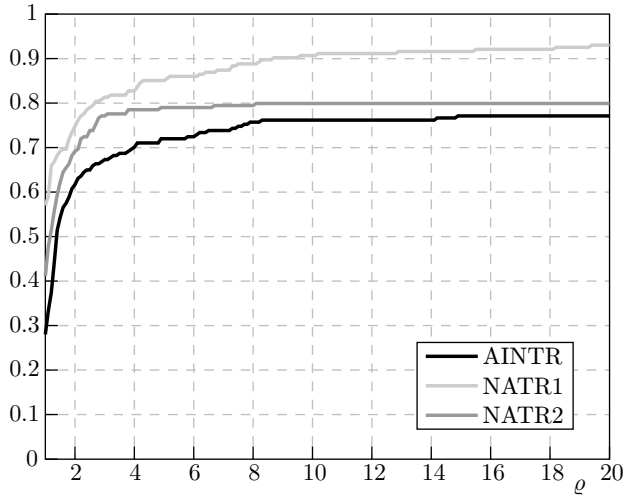

Figure 2. Performance profiles for the number of iterations.

The performance profiles for the running times are illustrated in Figure 3. From this figure, it can be observed that NATR1 is the best algorithm. Another important factor of these three figures is that the graph of the NATR1 algorithm grows up faster than the others.

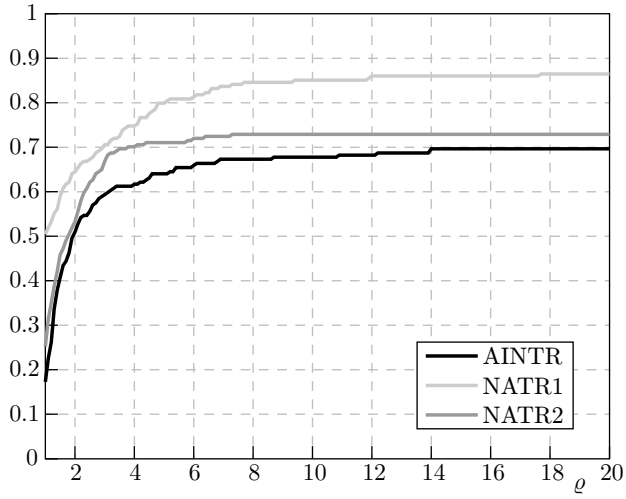

Figure 3. Performance profiles for the running times.

From the presented results, we can conclude that the radius dependent shrinkage parameter and the new nonmonotone procedure are effective to improve the efficiency of the IATR algorithm [13] compared with the nonmonotone algorithm proposed by Xue [22].

5. Conclusion

In this paper, we have proposed a new nonmonotone adaptive trust region algorithm to solve unconstrained optimization problems. The new algorithm incorporates a recently proposed adaptive trust region algorithm with nonmonotone techniques. We show that setting a constant shrinkage parameter for the adaptive trust region may impose unnecessary additional computational costs to the algorithm that affect its efficiency. Therefore, we consider a radius dependent shrinkage parameter in the new algorithm. Further, we propose a new nonmonotone parameter that prevents sudden increments in the objective function values.

The global convergence of the new algorithm is investigated under some mild conditions. Numerical experiments show the efficiency and robustness of the new algorithm in solving a collection of unconstrained optimization problems from the CUTEst package. It is concluded that exploiting the new ideas is a practical means to increase the efficiency of the nonmonotone adaptive trust region algorithms and these ideas also can be used in other nonmonotone and adaptive trust region algorithms which suffer from similar drawbacks mentioned in this paper.

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, M. R. Peyghami*: A nonmonotone trust-region line search method for large-scale unconstrained optimization. Appl. Math. Modelling *36* (2012), 478–487. zbl MR doi

[3] *R. Ayanzadeh, S. Mousavi, M. Halem, T. Finin*: Quantum annealing based binary compressive sensing with matrix uncertainty. Available at `https://arxiv.org/abs/1901.00088` (2019), 15 pages.

[4] *R. Chen, M. Menickelly, K. Scheinberg*: Stochastic optimization using a trust-region method and random models. Math. Program. *169* (2018), 447–487. zbl MR doi

[5] *A. R. Conn, N. I. M. Gould, P. L. Toint*: Trust Region Methods. MPS/SIAM Series on Optimization 1. SIAM, Philadelphia, 2000. zbl MR doi

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

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

[8] *H. Esmaeili, M. Kimiaei*: A trust-region method with improved adaptive radius for systems of nonlinear equations. Math. Methods Oper. Res. *83* (2016), 109–125. zbl MR doi

[9] *N. I. M. Gould, S. Lucidi, M. Roma, P. L. Toint*: Solving the trust-region subproblem using the Lanczos method. SIAM J. Optim. *9* (1999), 504–525. zbl MR doi

[10] *N. I. M. Gould, D. Orban, P. L. Toint*: CUTEst: A constrained and unconstrained testing environment with safe threads for mathematical optimization. Comput. Optim. Appl. *60* (2015), 545–557. zbl MR 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] *M. Hong, M. Razaviyayn, Z. Q. Luo, J.-S. Pang*: A unified algorithmic framework for block-structured optimization involving big data: With applications in machine learning and signal processing. IEEE Signal Processing Magazine *33* (2016), 57–77. doi
[13] *A. Kamandi, K. Amini, M. Ahookhosh*: An improved adaptive trust-region algorithm. Optim. Lett. *11* (2017), 555–569. zbl MR doi
[14] *J. J. Moré, D. C. Sorensen*: Computing a trust region step. SIAM J. Sci. Stat. Comput. *4* (1983), 553–572. zbl MR doi
[15] *J. Nocedal, S. J. Wright*: Numerical Optimization. Springer Series in Operations Research and Financial Engineering. Springer, New York, 2006. zbl MR
[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] *R. B. Schnabel, E. Eskow*: A new modified Cholesky factorization. SIAM J. Sci. Stat. Comput. *11* (1990), 1136–1158. zbl MR doi
[18] *J. Shen, S. Mousavi*: Least sparsity of p-norm based optimization problems with $p > 1$. SIAM J. Optim. *28* (2018), 2721–2751. 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] *Z. Shi, S. Wang*: Nonmonotone adaptive trust region method. Eur. J. Oper. Res. *208* (2011), 28–36. zbl MR doi
[21] *T. Steihaug*: The conjugate gradient method and trust regions in large scale optimization. SIAM J. Numer. Anal. *20* (1983), 626–637. zbl MR doi
[22] *Y. Xue, H. Liu, Z. Liu*: An improved nonmonotone adaptive trust region method. Appl. Math., Praha *64* (2019), 335–350. zbl MR doi
[23] *X. Zhang, J. Zhang, L. Liao*: An adaptive trust region method and its convergence. Sci. China, Ser. A *45* (2002), 620–631. zbl MR
[24] *Q. Zhou, D. Hang*: Nonmonotone adaptive trust region method with line search based on new diagonal updating. Appl. Numer. Math. *91* (2015), 75–88. zbl MR doi

Authors' addresses: *Ahmad Kamandi* (corresponding author), Department of Mathematics, University of Science and Technology of Mazandaran, P.O. Box 48518-78195, Behshahr, Iran, e-mail: `ahmadkamandi@mazust.ac.ir`; *Keyvan Amini*, Department of Mathematics, Faculty of Science, Razi University, P.O. Box 67141-15111, Kermanshah, Iran, e-mail: `kamini@razi.ac.ir`.