Optim Lett (2017) 11:555–569
DOI 10.1007/s11590-016-1018-4

ORIGINAL PAPER

An improved adaptive trust-region algorithm

Ahmad Kamandi[1] · **Keyvan Amini**[1] · **Masoud Ahookhosh**[2]

Received: 3 October 2012 / Accepted: 9 February 2016 / Published online: 23 February 2016

Abstract This paper gives a variant trust-region method, where its radius is automatically adjusted by using the model information gathered at the current and preceding iterations. The primary aim is to decrease the number of function evaluations and solving subproblems, which increases the efficiency of the trust-region method. The next aim is to update the new radius for large-scale problems without imposing too much computational cost to the scheme. Global convergence to first-order stationary points is proved under classical assumptions. Preliminary numerical experiments on a set of test problems from the CUTEst collection show that the presented method is promising for solving unconstrained optimization problems.

Keywords Unconstrained optimization · Trust-region method · Adaptive radius technique · Global convergence

1 Introduction

Trust-region methods are a class of iterative schemes for solving unconstrained optimization problem

✉ Keyvan Amini
keyvanamini1353@yahoo.com; kamini@razi.ac.ir

Ahmad Kamandi
ahmadkamandi@razi.ac.ir

Masoud Ahookhosh
masoud.ahookhosh@univie.ac.at

1 Department of Mathematics, Faculty of Sciences, Razi University, Kermanshah, Iran

2 Faculty of Mathematics, University of Vienna, Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria

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

where $f : \mathbb{R}^n \to \mathbb{R}$ is a twice continuously differentiable function. Trust-region methods have strong global convergence properties and can be applied to ill-conditioned problems [4]. On the kth iteration of a trust-region method, an approximation point x_k and a matrix B_k, which is the exact Hessian $G_k = \nabla^2 f(x_k)$ or an its approximation, are available. Based on these information, the trial step d_k is obtained by solving the quadratic subproblem

$$\begin{aligned} \min\ & m_k(d) := f_k + g_k^T d + \frac{1}{2} d^T B_k d \\ \text{s.t.}\ & \|d\| \leq \delta_k, \end{aligned} \tag{2}$$

where, for the sake of simplicity, we assume that $f_k = f(x_k)$, $g_k = \nabla f(x_k)$, $\|\cdot\|$ denotes the Euclidean norm and δ_k is a radius of the region around x_k guaranteeing an agreement of the model and the objective function. Hence this region is called the trust-region. In traditional trust-region framework, an agreement between the model and the objective function is measured by the ratio

$$r_k = \frac{f_k - f(x_k + d_k)}{m_k(0) - m_k(d_k)}. \tag{3}$$

If the ratio r_k is close to unity, there is a good agreement between the model and the objective function over the current region. Therefore, it is safe to expand the trust-region radius for the next iteration. Conversely, if the ratio r_k is close to zero or even negative, then the model can not appropriately estimate the objective function, so the trust-region radius should be shrunk. As a result, the kth iteration is called successful if $r_k \geq \mu$ for some $\mu \in (0, 1)$ or unsuccessful if $r_k < \mu$.

The strategies of determining and updating the trust-region radius have crucial impacts in attaining the global convergence and computational cost as well. In addition, if the trust-region radius δ_k is very large, then the number of solving subproblems is increased, i.e., the amount of computational cost for solving a problem may be increased. One the other hand, if the trust-region radius δ_k is very small, the total number of iterations will be increased, which effects on the efficiency of the method. However, there are many researches on determining and updating the trust-region radius [1–3,7,10,12,13,15], there is not a general rule to be used. In general view, there are some open questions about determining and updating the trust-region radius that some of them are collected in the following:

- How do you select an efficient initial trust-region radius δ_0?
- How much do you expand the trust-region encountering with the successful iterations?
- How much do you shrink the trust-region encountering with the unsuccessful iterations?
- How do you select and update the trust-region radius for ill-condition and bad-scale problems?

Many authors have investigated the solutions of these questions, but no one have really claimed a general rule to generate and update the trust-region radius. As an example, Sartenaer [10] proposed a new approach to determine an initial radius by monitoring an agreement between the model and the objective function along the steepest descent direction computed at the starting point. But the parameters of this procedure may be dependent on the problem that should be solved. The most practical trust-region updating rule has been proposed by Gould et al. [7], where they examined the sensitivity of the trust-region methods on parameters related to the step acceptance test and trust-region radius updates. Although, they did not discuss an initial trust-region radius.

The pioneer interesting work on the adaptive trust-region radius was proposed by Zhang et al. [15], motivated by a problem in neural network. They introduced the adaptive trust-region radius

$$\delta_k = c^{p_k} \|g_k\| \, \|\widehat{B}_k^{-1}\|, \tag{4}$$

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 from Schnabel and Eskow [11]. The numerical performances of this approach showed that it is promising for small-scale problems. Since it needs to calculate the inverse matrix $\widehat{B}_k^{-1}$, one can not use this radius for large-scale optimization problems. Shi and Guo [12] proposed an adaptive radius given by

$$\delta_k = -c^{p_k} \frac{g_k^T q_k}{q_k^T \widetilde{B}_k q_k} \|q_k\|, \tag{5}$$

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

$$-\frac{g_k^T q_k}{\|g_k\| \cdot \|q_k\|} \geq \tau, \tag{6}$$

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 $q_k^T \widetilde{B}_k q_k > 0$. Theoretical analysis have indicated that the proposed trust-region procedure inherits the global convergence to a first-order critical point, and preliminary numerical results suggest that the approach is efficient for solving medium-scale unconstrained optimization problems.

Content This paper gives an improved version of the trust-region radius (5), which can overcome some disadvantages of the formula (5) to handle convex, nonconvex, and ill-conditioned objective functions. The new procedure by generating a suitable trust-region radius decreases the total number of solving subproblems and function evaluations. The global convergence to first-order critical points is established. The preliminary numerical results show the efficiency of the proposed method for some medium- and large-scale unconstrained optimization problems.

The remainder of this paper is organized as follows. Section 2 gives the motivations of our study, describes a variant algorithm and then establishes its global convergence analysis under some suitable conditions. The results of our numerical experiments for the proposed algorithm are reported in Sect. 3. Finally, some conclusions are delivered in Sect. 4.

2 New algorithm and convergence analysis

We here give the motivation of our study and propose a variant trust-region algorithm with an improved automatically adjusted radius. Finally, the global convergence of the proposed algorithm is established under some classical assumptions.

Although the proposed trust-region radius (5) has several practical benefits, it also consists of some disadvantages. We list some of them as follows:

- When x_k is far from the optimum and the matrix $\widetilde{B}_k$ is close to singular, the numerator of formula (5) is bounded away from zero but the denominator tends to zero. This possibly causes a very large trust-region radius, i.e., the number of solving subproblems may be increased. Therefore, it may be encounter with some difficulties for nonconvex or ill-conditioned objective functions.
- When x_k is close to the optimum and the matrix $\widetilde{B}_k$ is a positive definite matrix, the numerator of formula (5) tends to zero; however the denominator is bounded away from zero, i.e., the trust-region radius is seriously reduced. This case may increases the total number of iterations.
- The selection of $q_k = -B_k^{-1} g_k$ by Shi and Guo imposes an additional cost of solving a system of linear equation increasing the total computational cost, specially for large-scale problems.
- The procedure of constructing $\widetilde{B}_k$ is almost unnatural, this may cause some unsuitable radius.

To overcome the first and fourth disadvantages, we briefly review the quasi-Newton updating process of the matrix B_k to find a suitable updating process guaranteeing positive definite of this matrix. It is believed that among various quasi-Newton updates the BFGS formula has the most effective results, but it can not guarantee that the matrix B_k remains positive definite for the nonconvex functions. For constructing a positive definite version of BFGS updating formula and supporting nonconvex functions, Li and Fukushima proposed a modified secant equation preserving a positive definite property of B_k for any given objective function $f(x)$ [8,14]. They reformed the BFGS update formula as follows

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k^* (y_k^*)^T}{s_k^T y_k^*}, \tag{7}$$

where $y_k^* = y_k + t_k s_k$, $s_k = x_{k+1} - x_k$, $y_k = g_{k+1} - g_k$, and t_k is determined by

$$t_k = \bar{C} \|g_k\|^{-\omega} + \max\left\{ \frac{-s_k^T y_k}{\|s_k\|^2}, 0 \right\} > 0,$$

with the constants $\bar{C} > 0$ and $\omega > 0$. The update formula (7) is called MBFGS. It is proved that B_{k+1} generated by (7) inherits the positive definiteness of B_k without the convexity assumption of the objective function. Also the numerical results of MBFGS update is very interesting. Based on these interesting characteristics and due to overcome the first and fourth mentioned disadvantages, we exploit the MBFGS updating formula to construct both the model and the adaptive trust-region radius.

On the other hand, our numerical experiments indicate that q_k has some crucial effects on (5), so a proper selection of q_k is so essential. As we mentioned in the third drawback, the selection of q_k should not impose an additional computational cost to the algorithm, but $q_k = -B_k^{-1} g_k$, proposed by Shi and Guo [12], imposes solving a system of linear equation increasing the total computational cost of the algorithm, especially for large-scale problems. Hence we propose to use

$$q_k := \begin{cases} -g_k & \text{if } k = 0 \text{ or } \dfrac{-(g_k^T d_{k-1})}{\|g_k\| \|d_{k-1}\|} \leq \tau, \\ d_{k-1} & \text{otherwise,} \end{cases} \tag{8}$$

where d_{k-1} is a solution of the subproblem (2) which can be accessed and $\tau \in (0, 1)$. It is straightforward that q_k satisfies the condition (6). To avoid a very small trust-region radius, the second disadvantage, we decided to define the formula

$$s_k := \begin{cases} -\dfrac{g_k^T q_k}{q_k^T B_k q_k} \|q_k\| & \text{if } k = 0, \\ \max\left\{ -\dfrac{g_k^T q_k}{q_k^T B_k q_k} \|q_k\|, \ \gamma \delta_{k-1} \right\} & \text{if } k \geq 1, \end{cases} \tag{9}$$

where B_k is updated by (7), $\gamma > 1$, and q_k is determined by (8). Then, the trust-region is defined by

$$\delta_k := c^{p_k} \min\{s_k, \overline{\delta}\}, \tag{10}$$

where $\overline{\delta} > 0$ is a real-valued constant, $c \in (0, 1)$ and p_k is the smallest nonnegative integer for which $r_k \geq \mu$, where $\mu \in (0, 1)$ is a parameter for successful iterations.

In view of our discussion about how to update the Hessian approximation B_k and how to select the radius δ_k, the new trust-region algorithm with adjustable radius is given in the following:

Algorithm 1: IATR (improved adaptive trust-region algorithm)

```
Input: $x_0 \in \mathbb{R}^n$, a positive definite matrix $B_0 \in \mathbb{R}^{n\times n}$, $\bar{\delta} > 0$, $c \in (0,1)$, $\mu > 0$,
       $\gamma > 1$, $\varepsilon > 0$;
Output: $x_b$; $f_b$;
begin
    $k \leftarrow 0$; compute $f_0$ and $g_0$;
    while $\|g_k\| \geq \varepsilon$ do
        compute $q_k$ by (8), $s_k$ by (9), and set $\delta_k = \min\{s_k, \bar{\delta}\}$;
        $r_k \leftarrow 0$, $p \leftarrow 0$;
        while $r_k < \mu$ do
            $\delta_k = c^p \delta_k$;
            compute $d_k$ by solving (2); compute $r_k$ by (3);
            $p \leftarrow p + 1$
        end
        $x_{k+1} \leftarrow x_k + d_k$; update $B_k$ by (7); $k \leftarrow k + 1$;
    end
    $x_b \leftarrow x_k$, $f_b \leftarrow f_k$;
end
```

In Algorithm 1, if $r_k \geq \mu$ (Line 6), it is called a successful iteration. In addition, in the algorithm, the loop started from Line 3 to Line 12 is called the outer cycle, and the loop started from Line 6 to Line 10 is called the inner cycle.

To verify the convergence analysis of the novel algorithm, the following assumptions are required:

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

$$\mathcal{L}(x_0) := \{x \in \mathbb{R}^n \mid f(x) \leq f(x_0),\ x_0 \in \mathbb{R}^n\}.$$

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

$$\|B_k\| \leq M, \quad \text{for all } k \in \mathbb{N}.$$

The subsequent results are essential results to give the global convergence of the sequence $\{x_k\}_{k\geq 0}$ generated by Algorithm 1.

Lemma 1 *Suppose that the sequence $\{x_k\}_{k\geq 0}$ is generated by Algorithm 1, then we have*

$$|f(x_k + d_k) - m_k(d_k))| = o(\|d_k\|).$$

Proof See Nocedal and Wright [9].

Lemma 2 *Suppose that (H2) holds, the sequence $\{x_k\}_{k\geq 0}$ is generated by Algorithm 1, and d_k is a solution of the subproblem (2) with δ_k given by (10). Then*

$$m_k(0) - m_k(d_k) \geq \frac{1}{2} c^{p_k} \min\left\{ \frac{1}{M}\left(\frac{-g_k^T q_k}{\|q_k\|}\right)^2, \bar{\delta}\left(\frac{-g_k^T q_k}{\|q_k\|}\right)\right\}, \tag{11}$$

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

Proof If $s_k \leq \bar{\delta}$, then $\delta_k = c^{p_k} s_k$, leading to

$$\delta_k = c^{p_k} \max\left\{ -\frac{g_k^T q_k}{q_k^T B_k q_k}\ \|q_k\|,\ \gamma \delta_{k-1}\right\} \geq -c^{p_k} \frac{g_k^T q_k}{q_k^T B_k q_k}\ \|q_k\|.$$

This means that

$$\widehat{d_k} = -c^{p_k} \frac{g_k^T q_k}{q_k^T B_k q_k} q_k$$

is a feasible point for the subproblem (2). This fact, along with (H2), implies that

$$\begin{aligned} m_k(0) - m_k(d_k) \geq m_k(0) - m_k(\widehat{d_k}) &= -g_k^T \widehat{d_k} - \frac{1}{2} \widehat{d_k}^T B_k \widehat{d_k} \\ &= c^{p_k} \frac{g_k^T q_k}{q_k^T B_k q_k}\left(g_k^T q_k - \frac{1}{2} c^{p_k} \frac{g_k^T q_k}{q_k^T B_k q_k} q_k^T B_k q_k\right) \\ &= c^{p_k} \frac{(g_k^T q_k)^2}{q_k^T B_k q_k}\left(1 - \frac{1}{2} c^{p_k}\right) \geq c^{p_k} \frac{(g_k^T q_k)^2}{q_k^T B_k q_k}\left(1 - \frac{1}{2}\right) \\ &= \frac{1}{2} c^{p_k}\ \frac{(g_k^T q_k)^2}{q_k^T B_k q_k} \geq \frac{c^{p_k}}{2M}\left(\frac{g_k^T q_k}{\|q_k\|}\right)^2. \end{aligned} \tag{12}$$

If $s_k > \bar{\delta}$, then $\delta_k = \bar{\delta}$. We consider two cases: (i) $-(g_k^T q_k)/(q_k^T B_k q_k)\ \|q_k\| \leq \bar{\delta}$; (ii) $-(g_k^T q_k)/(q_k^T B_k q_k)\ \|q_k\| > \bar{\delta}$. In Case (i), by $\delta_k = c^{p_k}\bar{\delta}$, the point

$$\hat{d}_k = -c^{p_k} \frac{g_k^T q_k}{q_k^T B_k q_k} q_k$$

is feasible for the subproblem (2). Then, by (12), we get

$$m_k(0) - m_k(d_k) \geq \frac{c^p}{2M}\left(\frac{g_k^T q_k}{\|q_k\|}\right)^2.$$

In Case (ii), we have

$$-\frac{\bar{\delta}\, q_k^T B_k q_k}{g_k^T q_k\, \|q_k\|} < 1. \tag{13}$$

Since $\delta_k \le \bar{\delta}$, the point $\widetilde{d}_k = c^{p_k}(\bar{\delta})/(\|q_k\|)q_k$ is feasible for the subproblem (2). This and (13) leads to

$$\begin{aligned} m_k(0) - m_k(d_k) \ge m_k(0) - m_k(\widetilde{d}_k) &= -g_k^T \widetilde{d}_k - \frac{1}{2}\widetilde{d}_k^T B_k \widetilde{d}_k \\ &= c^{p_k}\bar{\delta}\left(\frac{-g_k^T q_k}{\|q_k\|}\right)\left(1 - \frac{1}{2}c^{p_k}\frac{\bar{\delta}}{-\|q_k\| g_k^T q_k} q_k^T B_k q_k\right) \\ &\ge c^{p_k}\bar{\delta}\left(\frac{-g_k^T q_k}{\|q_k\|}\right)\left(1 - \frac{1}{2}c^{p_k}\right) \\ &\ge \frac{1}{2}c^{p_k}\bar{\delta}\left(\frac{-g_k^T q_k}{\|q_k\|}\right), \end{aligned}$$

giving the result. □

The next result shows that the inner cycle of Algorithm 1 is terminated in the finite number of internal iterations.

Lemma 3 *Suppose that (H2) holds, and the sequence* $\{x_k\}_{k\ge 0}$ *is generated by Algorithm 1. Then the inner cycle of Algorithm 1 is well-defined.*

Proof By contradiction, we assume that the inner cycle of Algorithm 1 cycles infinity. From this fact that x_k is not the optimum, we know that there is a constant $\varepsilon > 0$ such that $\|g_k\| \ge \varepsilon$. This fact, together with (6), yields

$$-\frac{g_k^T q_k}{\|q_k\|} \ge \tau\varepsilon. \tag{14}$$

Now, let d_k^p be the solution of subproblem (2) corresponding to $p \in \mathbb{N} \cup \{0\}$ at x_k. It follows from Lemma 1, (11), and (14) that

$$\begin{aligned} \left|\frac{f(x_k) - f(x_k + d_k^p)}{m_k(0) - m_k(d_k^p)} - 1\right| &= \left|\frac{f(x_k) - f(x_k + d_k^p) - (m_k(0) - m_k(d_k^p))}{m_k(0) - m_k(d_k^p)}\right| \\ &\le \frac{o(\|d_k^p\|)}{\frac{1}{2}c^p \min\left\{\frac{1}{M}\left(\frac{-g_k^T q_k}{\|q_k\|}\right)^2, \bar{\delta}\left(\frac{-g_k^T q_k}{\|q_k\|}\right)\right\}} \\ &\le \frac{o(\|d_k^p\|)}{\frac{1}{2}c^p \min\left\{\frac{1}{M}(\tau\varepsilon)^2, \bar{\delta}(\tau\varepsilon)\right\}}. \end{aligned}$$

By the assumption that the inner cycle cycles infinity and (10), we know $\delta_k^p \to 0$ as $p \to \infty$. Hence $\|d_k^p\| \leq \delta_k \leq c^p s_k$ implying that the right hand side the above equation tends to zero. Therefore, for sufficiently large p_k, we get

$$r_k = \frac{f(x_k) - f(x_k + d_k^{p_k})}{m_k(0) - m_k(d_k^{p_k})} \geq \mu,$$

implying that the inner cycle of the Algorithm 1 is terminated in the finite number of internal iterations giving the result. □

To prove the global convergence of the sequence generated by Algorithm 1, the next two statements are required to be established.

Lemma 4 *Suppose that (H1) holds and the sequence $\{x_k\}_{k\geq 0}$ is generated by Algorithm* 1. Then the sequence $\{f_k\}_{k\geq 0}$ is convergent.

Proof We first, by induction, show that $\{x_k\}_{k\geq 0} \subseteq \mathcal{L}(x_0)$. It is clear that $x_0 \in \mathcal{L}(x_0)$. Hence we assume $x_k \in \mathcal{L}(x_0)$ and show $x_{k+1} \in \mathcal{L}(x_0)$. Using $r_k \geq \mu$ we obtain

$$f_k \geq f(x_k + d_k) + \mu(m_k(0) - m_k(d_k)) \geq f(x_k + d_k). \tag{15}$$

This inequality and the assumption of induction yield that $x_{k+1} \in \mathcal{L}(x_0)$. On the other hand, the inequality (15) suggests that the sequence $\{f_k\}_{k\geq 0}$ is a decreasing sequence. Therefore, this fact and (H1) the sequence $\{f_k\}_{k\geq 0}$ is convergent. □

Lemma 5 *Suppose that (H1) and (H2) hold, then Algorithm* 1 *either stops at a stationary point of* (1) *or generates an infinite sequence $\{x_k\}_{k\geq 0}$ such that*

$$\lim_{k\to\infty} -\frac{g_k^T q_k}{\|q_k\|} = 0. \tag{16}$$

Proof Let us assume that Algorithm 1 does not stop finitely. Then we will show that (16) holds. To obtain a contradiction, we suppose that there exist a constant $\varepsilon_0 > 0$ and an infinite subset $\mathcal{K} \subseteq \mathbb{N} \cup \{0\}$ such that

$$-\frac{g_k^T q_k}{\|q_k\|} \geq \varepsilon_0 \tag{17}$$

for all $k \in \mathcal{K}$. Using Lemma 2, (17), and $r_k \geqslant \mu$ we get

$$\begin{aligned} f_k - f(x_k + d_k) &\geq \mu(m_k(0) - m_k(d_k)) \\ &\geq \frac{1}{2}\mu c^{p_k} \min\left\{\frac{1}{M}\left(\frac{-g_k^T q_k}{\|q_k\|}\right)^2, \bar{\delta}\left(\frac{-g_k^T q_k}{\|q_k\|}\right)\right\} \\ &\geq \frac{1}{2}\mu c^{p_k} \min\left\{\frac{1}{M}\varepsilon_0^2, \bar{\delta}\varepsilon_0\right\}. \end{aligned}$$

This inequality together with Lemma 4, as $k \to \infty$, imply

$$0 \geq \mu \frac{1}{2} c^{p_k} \min \left\{ \frac{1}{M} \varepsilon_0^2, \bar{\delta}\varepsilon_0 \right\}.$$

This means that $c^{p_k} \to 0$ as $k \to \infty$ which is a contradiction with Lemma 3. Therefore there exists no infinite subset of $\mathcal{K}$ such that (17) holds and the proof is completed. □

Here, we are in the position to establish that all limit points of the sequence $\{x_k\}_{k\geq 0}$ generated by Algorithm 1 satisfy the first-order optimality condition.

Theorem 1 *Suppose that all conditions of Lemma 5 hold and q_k satisfies (6). Then Algorithm 1 either terminates finitely or generates an infinite sequence $\{x_k\}_{k\geq 0}$ such that*

$$\lim_{k\to\infty} \|g_k\| = 0. \tag{18}$$

Proof If Algorithm 1 terminates finitely, then the proof is obvious. Assume that Algorithm 1 generates an infinite sequence $\{x_k\}_{k\geq 0}$, Lemma 5 indicates that this sequence satisfies (16). Since q_k satisfies (6), we obtain

$$0 \leq \tau \|g_k\| \leq -\frac{g_k^T q_k}{\|g_k\|.\|q_k\|} \|g_k\| = -\frac{g_k^T q_k}{\|q_k\|} \to 0, \quad as\ k \to \infty.$$

Therefore, (18) holds, and the sequence $\{x_k\}_{k\geq 0}$ converges to first-order optimizer points. □

Let us conclude this section by this statement that the local superlinear and the local quadratic convergence rates of the sequence $\{x_k\}_{k\geq 0}$ generated by Algorithm 1 can be shown the same as Theorems 4.1 and 4.2 in [1], respectively.

3 Preliminary numerical experiments

In this section we report some computational experiments of the proposed algorithm (IATR) compared with a traditional trust-region algorithm in [4] (TTR), a version of the adaptive algorithm of Shi and Guo [12] using the BFGS updating formula (SATR1) and a version of this algorithm employing the MBFGS updating formula (ATRS2). Since the algorithm of Zhang et al. [15] needs to compute an approximation of the inverse Hessian B_k^{-1} in each iteration, one can not apply it for large-scale problems. Hence our comparison does not involve this algorithm. In our experiment, we use 188 test problems of the CUTEst test collections [6] from the dimension 10 to 5000 (see Table 3).

All codes are written in MATLAB 7.4 programming environment with double precision format. In running the algorithms, we check whether different codes converge to the same local minimizer and only provide results for problems in which all algorithms

Table 1 A comparison among 4 versions of IATR by setting $\tau = 1,\ 10^{-2},\ 10^{-4},\ 10^{-6}$

Parameter τ	1	1e−2	1e−4	1e−6
Average gradient evaluations	1.2442e+3	1.2208e+3	1.2292e+3	1.2292e+3
Average function evaluations	1.3759e+3	1.3394e+3	1.3550e+3	1.3550e+3
Average running time	1.0217e+3	1.0204e+3	1.0210e+3	1.0210e+3

converge to the same minimizer. For IATR, we use the parameters $\mu = 0.01, c = 0.35$, $\overline{\delta} = 100$, and $\gamma = 1.7$, and the others use the original parameters discussed in the related literature. To solve the quadratic subproblem (2), we use the Steihaug-Toint scheme [4] (Chapter 7, Page 205) where the scheme is terminated if

$$\|g(x_k + d)\| \le \min\left\{1/10,\ \|g_k\|^{1/2}\right\} \|g_k\| \quad \text{or} \quad \|d\| = \delta_k.$$

In our experiments, the algorithms stop whenever

$$\|g_k\| \le 10^{-6}\|g_0\|$$

or when the total number of iterations exceeds 4000.

Since the parameter τ in (8) plays a crucial role in our scheme, we give a comparison among 4 versions of IATR by setting $\tau = 1,\ 10^{-2},\ 10^{-4},\ 10^{-6}$. The results of our comparison in Table 1 for the number of gradient evaluations (N_g), the number of function evaluations (N_f), and the running time (T) are given in Table 1. The results for all considered measures suggest that IATR attain the best results with $\tau = 10^{-2}$ that we will use for further use of IATR. Note that $\tau = 1$ is equivalent to setting $q_k = -gk$, which produces the worst results in our setting while the algorithm performs almost the same for $\tau = 10^{-4}$ and $\tau = 10^{-6}$.

To compare the performance of considered algorithms, we exploit the performance profile of Dolan and Moré [5]. In this procedure, the performance of each code is measured by the ratio of its computational outcome versus the best numerical outcome of all algorithms. Then the statistical structure is offered for comparing the performance of iterative processes. In particular, let $\mathcal{S}$ is a set of all algorithms and $\mathcal{P}$ denotes a set of test problems, with n_s solvers and n_p problems. For each problem p and solver s, $t_{p,s}$ is the computed result regarding to the performance index. Now, the following performance ratio is defined

$$r_{p,s} = (t_{p,s})/\min\{t_{p,s} : s \in \mathcal{S}\}. \tag{19}$$

In the case that an algorithm s is not convergent for a problem p, the procedure sets $r_{p,s} = r_{\text{fail}}$, where r_{fail} should be strictly larger than any performance ratio (19). For any factor τ, the overall performance of algorithm s is given by

$$\rho_s(\tau) = 1/n_p \,\text{size}\{p \in \mathcal{P} : r_{p,s} \le \tau\}. \tag{20}$$

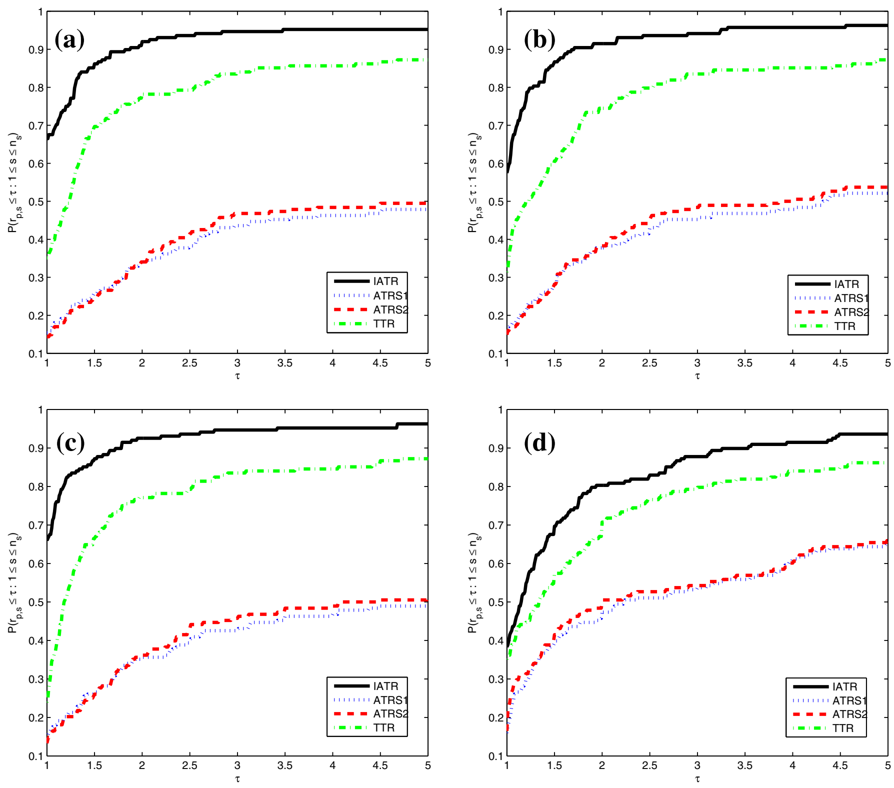

Fig. 1 A comparison among IATR, ATRS1, ATRS2, and TTR with the performance profile for unconstrained optimization problem of the form (1). Subfigure **a** displays the number of gradient evaluations (N_g); subfigure **b** shows the number of function evaluations (N_f); subfigure **c** illustrates the mixed measure $N_f + 3N_g$; subfigure **d** displays the running time T

The function $\rho_s(\tau)$ is the distribution function showing the performance ratio $r_{p,s}$ is within a factor $\tau \in \mathbb{R}^n$ of the best possible ratio for the algorithm s. In particular, $\rho_s(1)$ gives the probability that algorithm s wins to the other algorithms, and $\lim_{\tau \to r_{\text{fail}}} \rho_s(\tau)$ gives the probability of that algorithm s solve a problem. Therefore, this performance profile can be considered as a measure of the efficiency of iterative processes. In Fig. 1, the x-axis shows the number τ while the y-axis displays $P(r_{p,s} \le \tau : 1 \le s \le n_s)$.

In Fig. 1, subfigure (a–c) display the performance profiles for N_g, N_f, and the mixed measure $N_f + 3N_g$, respectively. In these three subfigures, IATR attains the most wins with about 67, 62, and 66 % score, respectively, and outperforms the others significantly; TTR stands in the second place, while ATRS1 and ATRS2 are comparable. Subfigure (d) of Fig. 1 shows the performance profile for the running time T, where IATR are slightly better than TTR and both of them outperforms ATRS1 and ATRS2 considerably. In the later case, one can see that IATR grows up faster than TTR which means in the cases that this algorithm does not get the best result, it performs close to the performance index of the best algorithm.

Table 2 The percentage of success for solving the 188 considered problems for IATR, ATRS1, ATRS2, and TTR by considering several fixed number of function evaluations as a stopping criterion

Max. func.	100	200	300	400	500	1000	2000	3000	5000	10,000	15,000	20,000
IATR	51.06	65.95	68.61	70.74	73.93	85.10	90.42	93.08	94.14	95.21	95.21	100.00
ATRS1	26.06	29.25	32.97	34.57	36.17	44.14	46.80	53.19	59.04	59.04	59.04	98.40
ATRS2	26.06	30.85	34.57	37.23	38.29	48.40	48.93	56.38	61.17	61.17	61.17	98.40
TTR	42.55	56.38	60.10	62.23	65.95	75.53	80.85	84.57	86.17	86.17	86.17	98.40

We now consider a different measure to compare the results of IATR, ATRS1, ATRS2, and TTR by fixing the total number of function evaluations and considering the percentage of success for solving the 188 considered problems. The results of our comparison for several maximum number of function evaluations are given in Table 2. The results show that IATR are able to solve much more problems than ATRS1, ATRS2, and TTR for the fixed number of function evaluations.

Summarizing the results of our implementation in Fig. 1 and Table 2 shows that IATR performs promising for solving unconstrained optimization problems of the form (1) and outperforms the algorithms considered in our comparison significantly; however further numerical experiments are required to figure out a clear picture about the performance of IATR.

4 Concluding remarks

This study describes a novel trust-region method for unconstrained optimization, which exploits the automatically adjustable radius and attempts to improve it by overcoming some disadvantages of the original framework. It is known that the traditional trust-region procedure has some disadvantages about how to select a suitable initial radius and how to update it in each iteration. To overcome these disadvantages and to decrease the total number of solving subproblems and function evaluations, several adaptive trust-region radiuses have been proposed and developed. Here, the proposed adaptive radius of Shi and Guo [12] is analyzed carefully to give an improved version of it, which can handle the convex, nonconvex, and ill-conditioned objective functions for large number of variables. It is proved that every limit point of the sequence of iterations generated by the algorithm is satisfied in the first-order optimality condition. The promising behavior of the proposed method is encouraging for the unconstrained optimization problems.

Acknowledgements We are very thankful to two anonymous referees for careful reading and many useful suggestions, which improve the paper.

Appendix

See Table 3 for the list of test problems from the CUTEst collection.

Table 3 List of test problems

Problem name	Dim	Problem name	Dim	Problem name	Dim	Problem name	Dim
ARGLINA	200	OSCIPATH	500	DIXMAANG	1500	NONCVXUN	100
ARGLINB	200	PENALTY1	100	DIXMAANG	3000	NONCVXUN	1000
ARGLINC	200	PENALTY1	500	DIXMAANH	1500	NONDQUAR	100
BDQRTIC	500	PENALTY1	1000	DIXMAANH	3000	NONDQUAR	500
BDQRTIC	1000	PENALTY2	100	DIXMAANI	1500	PENALTY3	100
BROWNAL	100	PENALTY2	200	DIXMAANI	3000	POWELLSG	60
BROWNAL	200	SENSORS	100	DIXMAANJ	1500	POWELLSG	80
BROWNAL	1000	SPMSRTLS	100	DIXMAANJ	3000	POWELLSG	100
BRYBND	500	SPMSRTLS	499	DIXMAANK	1500	POWELLSG	500
CHAINWOO	100	SPMSRTLS	1000	DIXMAANK	3000	POWELLSG	1000
CHAINWOO	1000	SROSENBR	100	DIXMAANL	1500	POWER	50
CHNROSNB	50	SROSENBR	500	DIXMAANL	3000	POWER	100
CHNRSNBM	50	SROSENBR	1000	DIXMAANM	1500	POWER	500
CURLY10	100	SROSENBR	5000	DIXMAANM	3000	POWER	1000
CURLY20	100	TQUARTIC	100	DIXMAANN	1500	QUARTC	100
CURLY30	100	TQUARTIC	500	DIXMAANN	3000	QUARTC	500
EIGENALS	110	TQUARTIC	1000	DIXMAANO	1500	SCHMVET	100
ERRINROS	50	VAREIGVL	99	DIXMAANO	3000	SCHMVET	500
ERRINRSM	50	VAREIGVL	499	DIXMAANP	1500	SCHMVET	1000
EXTROSNB	100	VAREIGVL	999	DIXMAANP	3000	SINQUAD	50
FREUROTH	100	WATSON	31	DQRTIC	50	SINQUAD	100
FREUROTH	500	WOODS	50	DQRTIC	100	SINQUAD	500
GENROSE	100	WOODS	1000	DQRTIC	500	SINQUAD	1000
LIARWHD	100	ARWHEAD	100	EDENSCH	2000	SPARSINE	50
LIARWHD	500	ARWHEAD	500	ENGVAL1	1000	SPARSINE	100
LIARWHD	1000	ARWHEAD	1000	ENGVAL1	5000	SPARSINE	1000
MANCINO	50	ARWHEAD	5000	FLETCBV2	100	SPARSQUR	100
MANCINO	100	BOX	100	FLETCBV2	1000	SPARSQUR	1000
MODBEALE	200	BOX	1000	FLETCBV3	100	SPARSQUR	5000
MODBEALE	2000	BOXPOWER	100	FLETCBV3	1000	TOINTGSS	100
MSQRTALS	100	BOXPOWER	1000	FLETCHCR	100	TOINTGSS	500
MSQRTALS	529	BROYDN7D	100	FMINSRF2	64	TOINTGSS	1000
MSQRTBLS	100	BROYDN7D	500	FMINSRF2	121	TOINTGSS	5000
MSQRTBLS	529	COSINE	100	FMINSRF2	1024	VARDIM	50
NONDIA	50	COSINE	1000	FMINSURF	64	VARDIM	100
NONDIA	90	CRAGGLVY	100	FMINSURF	121	VARDIM	200
NONDIA	100	CRAGGLVY	500	FMINSURF	1024	DIXON3	100
NONDIA	500	DIXMAANA	1500	INDEFM	50	DQDRTI	50
NONDIA	1000	DIXMAANA	3000	INDEFM	100	DQDRTI	100
NONDIA	5000	DIXMAANA	1500	INDEFM	1000	DQDRTI	500

Table 3 continued

Problem name	Dim	Problem name	Dim	Problem name	Dim	Problem name	Dim
NONSCOMP	50	DIXMAANA	3000	NCB20	100	DQDRTI	1000
NONSCOMP	100	DIXMAAND	1500	NCB20B	50	HILBERTA	10
NONSCOMP	500	DIXMAAND	3000	NCB20B	100	HILBERTB	50
NONSCOMP	1000	DIXMAANE	1500	NCB20B	180	TESTQUAD	100
OSCIGRAD	100	DIXMAANE	3000	NCB20B	500	TOINTQOR	50
OSCIGRAD	1000	DIXMAANF	1500	NONCVXU2	100	TRIDIA	50
OSCIPATH	100	DIXMAANF	3000	NONCVXU2	1000	TRIDIA	100

References

1. Ahookhosh, M., Amini, K.: A nonmonotone trust region method with adaptive radius for unconstrained optimization. Comput. Math. Appl. **60**, 411–422 (2010)
2. Ahookhosh, M., Esmaeili, H., Kimiaei, M.: An effective trust-region-based approach for symmetric nonlinear systems. Int. J. Comput. Math. **90**(3), 671–690 (2013)
3. Amini, K., Ahookhosh, M.: A hybrid of adjustable trust-region and nonmonotone algorithms for unconstrained optimization. Appl. Math. Modell. **38**, 2601–2612 (2014)
4. Conn, A.R., Gould, N.I.M., Toint, PhL: Trust Region Methods. Society for Industrial and Applied Mathematics. SIAM, Philadelphia (2000)
5. Dolan, E., Moré, J.J.: Benchmarking optimization software with performance profiles. Math. Program. **91**, 201–213 (2002)
6. Gould, N.I.M., Orban, D., Toint, Ph.L.: CUTEst: a constrained and unconstrained testing environment with safe threads for mathematical optimization. Comput. Optim. Appl. **60**, 545–557 (2015)
7. Gould, N.I.M., Orban, D., Sartenaer, A., Toint, PhL: Sensitivity of trust region algorithms to their parameters. Q. J. Oper. Res. **3**, 227–241 (2005)
8. Li, D.H., Fukushima, M.: A modified BFGS method and its global convergence in nonconvex minimization. J. Comput. Appl. Math. **129**, 15–35 (2001)
9. Nocedal, J., Wright, S.J.: Numerical Optimization. Springer, NewYork (2006)
10. Sartenaer, A.: Automatic determination of an initial trust region in nonlinear programming. SIAM J. Sci. Comput. **18**(6), 1788–1803 (1997)
11. Schnabel, R.B., Eskow, E.: A new modified Cholesky factorization. SIAM J. Sci. Comput. **11**(6), 1136–1158 (1990)
12. Shi, Z.J., Guo, J.H.: A new trust region method with adaptive radius. Comput. Optim. Appl. **41**, 225–242 (2008)
13. Walmag, J.M.B., Delhez, E.J.M.: A note on trust region radius update. SIAM J. Optim. **16**(2), 548–562 (2005)
14. Xiao, Y., Sun, H., Wang, Z.: A globally convergent BFGS method with nonmonotone line search for non-convex minimization. J. Comput. Appl. Math. **230**, 95–106 (2009)
15. Zhang, X.S., Zhang, J.L., Liao, L.Z.: An adaptive trust region method and its convergence. Sci. China **45**, 620–631 (2002)