Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 610612, 8 pages
http://dx.doi.org/10.1155/2014/610612

Research Article

A Novel Self-Adaptive Trust Region Algorithm for Unconstrained Optimization

Yunlong Lu, Wenyu Li, Mingyuan Cao, and Yueting Yang

School of Mathematics and Statistics, Beihua University, Jilin 132013, China

Correspondence should be addressed to Yueting Yang; yyt2858@163.com

Received 28 August 2013; Revised 18 March 2014; Accepted 19 March 2014; Published 15 April 2014

Academic Editor: Kazutake Komori

A new self-adaptive rule of trust region radius is introduced, which is given by a piecewise function on the ratio between the actual and predicted reductions of the objective function. A self-adaptive trust region method for unconstrained optimization problems is presented. The convergence properties of the method are established under reasonable assumptions. Preliminary numerical results show that the new method is significant and robust for solving unconstrained optimization problems.

1. Introduction

Consider the following unconstrained optimization problem:

$$\min_{x\in\mathbb{R}^n} f(x), \tag{1}$$

where $f:\mathbb{R}^n \to \mathbb{R}$ is continuously differentiable.

The trust region methods calculate a trial step d_k by solving the subproblem at each iteration,

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

where $g_k = \nabla f(x_k)$ and B_k is symmetric matric approximating the Hessian of $f(x)$ at x_k, and $\Delta_k > 0$ is the current trust region radius. Throughout this paper, $\|\cdot\|$ denotes the l_2-norm. We define the ratio,

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

and the numerator and the denominator are called the actual reduction and the predicted reduction, respectively. For basic trust region (BTR) method, if the sufficient reduction predicted by the model is realized by the objective function, the trial point $x_k + d_k$ is accepted as the next iterate and the trust region is expanded or kept the same. If the model reduction turns out to be a poor predictor of the actual behavior of the objective function, the trial point is rejected and the trust region is contracted, with the hope that the model provides a better prediction in the smaller region. More formally, basic trust region radius update rule can be usually summarized as follows:

$$\Delta_{k+1} = \begin{cases} [\gamma_1\Delta_k, \gamma_2\Delta_k] & \text{if } r_k < \eta_1, \\ [\gamma_2\Delta_k, \Delta_k] & \text{if } \eta_1 \le r_k < \eta_2, \\ [\Delta_k, \infty) & \text{if } r_k \ge \eta_2, \end{cases} \tag{4}$$

where the constants γ_1, γ_2, η_1, and η_2 satisfy

$$0 \le \eta_1 < \eta_2 < 1, \qquad 0 < \gamma_1 \le \gamma_2 < 1. \tag{5}$$

The iteration is said to be *successful* if $r_k \ge \eta_1$. If not, the iteration is *unsuccessful*, and the trial point is rejected. If $r_k \ge \eta_2$, the iteration is said to be *very successful* iteration [1]. If r_k is significantly larger than one, that is, $r_k \ge \eta_3 > 1 > \eta_2$, the iteration is called *too successful* iteration [2].

Sartenaer [3] developed an elaborate strategy which can automatically determine an initial trust region radius. The basic idea is to determine a maximal initial radius through many repeated trials in the steepest descent direction in order to guarantee a sufficient agreement between the model and the objective function. This strategy requires additional evaluations of the objective function. Zhang et al. [4] presented another strategy of determining the trust region radius. Their basic idea is originated from the following subproblem in an artificial neural network research [5]:

$$\begin{aligned}\min_{d\in R^n}\ & q_k(d) = g_k^T d + \frac{1}{2} d^T B_k d \\ \text{s.t.}\ & -\Delta_k \le d_i \le \Delta_k, \quad i = 1, 2, \ldots, n,\end{aligned} \tag{6}$$

where $\Delta_k = c^p(\|g_k\|/\gamma)$, $0 < c < 1$, $\gamma = \min(\|B_k\|, 1)$, and p is a nonnegative integer. Therefore, instead of adjusting Δ_k, one adjusts p at each iteration. Motivated by this technique, they solved the trust region subproblem with

$$\Delta_k = c^p \|g_k\| \left\|\widehat{B}_k^{-1}\right\|, \tag{7}$$

where $c \in (0, 1)$, p is a nonnegative integer, and $\widehat{B}_k = B_k + iI$ is a positive definite matrix for some i. However, their method needs to estimate $\|B_k\|$ or $\|\widehat{B}^{-1}\|$ at each iteration, which leads to some additional cost of computation. As a result, Shi and Guo [6] proposed a simple adaptive trust region method. The new trust region model is more consistent with the objective function at the current iteration. Fu et al. [7] developed an adaptive trust region method based on the conic model by using the above adaptive strategy [4]. Sang and Sun [8] gave a new self-adaptive adjustment strategy to update the trust region radius, which makes full use of the information at the current point. Yu and Li [9] proposed a self-adaptive trust region algorithm for solving this nonsmooth equation. Some authors [1, 10] adopted different values for parameters (5) but seldom questioned the radius update formula (4). In addition, many adaptive nonmonotonic trust region methods have been proposed in [11–18]. Hei [19] proposed a self-adaptive update method, in which trust region radius is a product of a so-called R-function and Δ_k; that is, $\Delta_{k+1} = R(r_k)\Delta_k$, where $R(\cdot)$ is the R-function. As the ratio r_k is larger than one, $R(\cdot)$ is nondecreasing and bounded. However, as the iteration is very successful, the ratio r_k is larger than one; it implies that the local approximation of the objective function by the model function is not good. Walmag and Delhez [2] suggested that it is not overconfident in the model $q_k(d)$ at too successful iterations. They presented a self-adaptive update method, in which trust region radius is $\Delta_{k+1} = \Lambda(r_k)\Delta_k$, where $\Lambda(\cdot)$ is the Λ-function. If r_k is significantly larger than one, Λ-function is nonincreasing. However, they took $\Lambda(r_k) > 1$ but close to one to match the convergence criteria presented by Conn et al. [1].

In our opinion, the agreement between the model and the objective function is not good enough at too successful iterations. We take that the updated trust region radius Δ_{k+1} is less than Δ_k and Δ_{k+1} is bounded lower away from zero. It implies that $\Lambda(r_k) > 1$ is not always necessary. This strategy can also match the convergence criteria presented by Conn et al. [1].

In the paper, the L-function $L(\cdot)$ is introduced, which is a variant of Λ-function. A new self-adaptive trust region method is proposed, in which the trust region radius is $\Delta_{k+1} = L(r_k)\Delta_k$. The new method is more efficient at too successful iterations.

The rest of the paper is organized as follows. In Section 2, we define L-function to introduce new update rules and a new self-adaptive trust region algorithm is presented. In Section 3, the convergence properties of proposed algorithm are investigated. In Section 4, numerical results are given. In Section 5 conclusions are summarized.

2. L-Function and the New Self-Adaptive Trust Region Algorithm

To obtain the new trust region radius update rules, we define L-function $L(t)$, $t \in \mathbb{R}$.

Definition 1. A function $L(t)$ is called an L-function if it satisfies the following:

(1) $L(t)$ is nondecreasing in $(-\infty, \eta_2]$ and nonincreasing in $(2 - \eta_2, +\infty)$, $L(t) = \beta_2$, for $t \in [\eta_2, 2 - \eta_2]$,

(2) $\lim_{t\to-\infty} L(t) = c_1$,

(3) $L(0) = c_2$,

(4) $\lim_{t\to\eta_2^-} L(t) = 1$,

(5) $\lim_{t\to 0^+} L(t) = \beta_1$,

(6) $L(t) < 1$, for $t > \eta_3$,

(7) and $\lim_{t\to+\infty} L(t) = \beta_3$,

where the constants β_1, β_2, β_3, η_2, η_3, c_1, c_2 are positive constants such that

$$0 < c_1 < c_2 < \beta_1 \le \beta_3 < 1 < \beta_2, \qquad \eta_3 > 2 - \eta_2. \tag{8}$$

It is easy to prove that the L-function is a bounded function in $\mathbb{R}$. In the following, we show the differences between the usual empirical rule, the R-function rule, the Λ-function rule, and the L-function rule.

The usual empirical rules ([1, 20, 21]) (Figure 1(a)) can be usually summarized as follows:

$$\Delta_{k+1} = \begin{cases} \beta_1 \Delta_k, & \text{if } r_k < \eta_1, \\ \Delta_k, & \text{if } \eta_1 \le r_k < \eta_2, \\ \beta_2 \Delta_k, & \text{if } \eta_2 \le r_k, \end{cases} \tag{9}$$

where β_1, β_2, η_1, and η_2 are predefined constants such that

$$0 \le \eta_1 < \eta_2 < 1, \qquad 0 < \beta_1 < 1 < \beta_2. \tag{10}$$

The R-function rule and the Λ-function rule can be described as follows:

$$\Delta_{k+1} = R(r_k)\Delta_k, \qquad \Delta_{k+1} = \Lambda(r_k)\Delta_k, \tag{11}$$

where the R-function $R(r_k)$ (Figure 1(b)) proposed by Hei [19] is chosen as

$$R(r_k) = \begin{cases} \dfrac{2}{\pi}(M - 1 - \alpha_1)\arctan(r_k - \eta_1) + (1 + \alpha_1), & \text{if } r_k \ge \eta_1; \\ (1 - \alpha_2 - \beta)\exp(r_k - \eta_1) + \dfrac{\beta}{1 - \alpha_2 - \beta}, & \text{otherwise}, \end{cases} \tag{12}$$

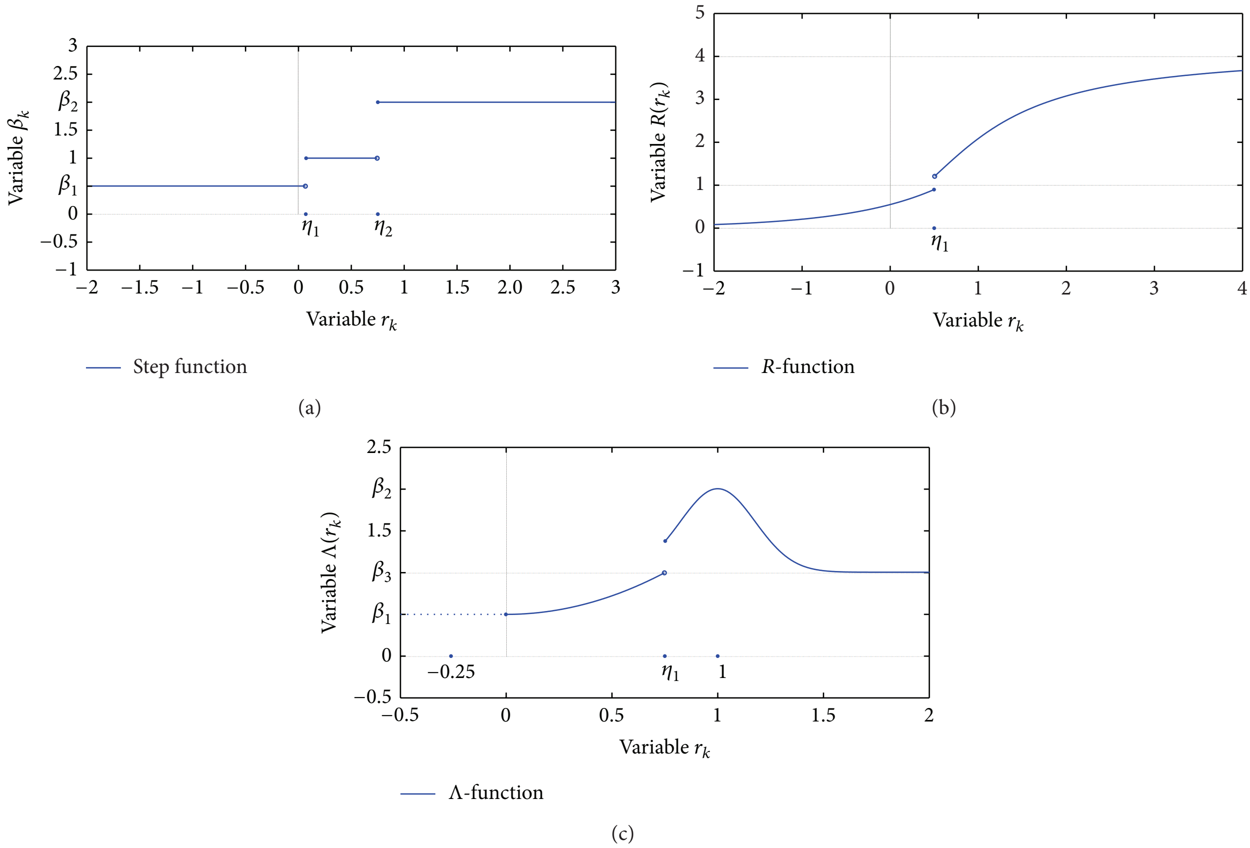

FIGURE 1: (a) Step function. (b) R-function $R(r_k)$. (c) Λ-function $\Lambda(r_k)$.

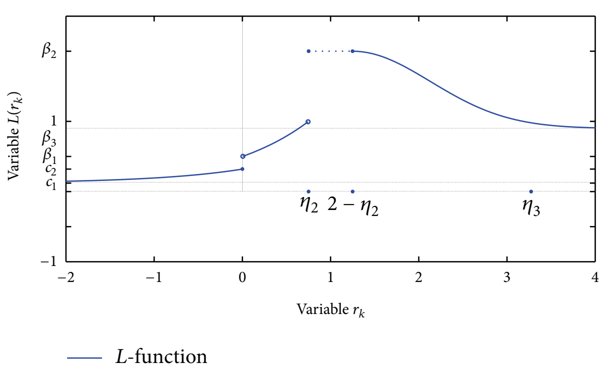

FIGURE 2: L-function $L(t)$.

where α_1, α_2, β, M, and η are constants, and the Λ-function $\Lambda(r_k)$ (Figure 1(c)) proposed by Walmag and Delhez [2] is chosen as

$$\Lambda(r_k) = \begin{cases} \beta_1, & \text{if } r_k \le 0, \\ \beta_1 + (1-\beta_1)\left(\dfrac{r_k}{\eta_1}\right)^2, & \text{if } 0 < r_k < \eta_1, \\ \beta_3 + (\beta_2 - \beta_3)\exp\left(-\left(\dfrac{r_k - 1}{\eta_1 - 1}\right)^2\right), & \text{if } r_k \ge \eta_1, \end{cases} \tag{13}$$

where β_1, β_2, β_3, and η_1 are constants.

The L-function rule can be described as follows:

$$\Delta_{k+1} = L(r_k)\,\Delta_k, \tag{14}$$

where the L-function $L(r_k)$ (Figure 2) is chosen as

$$L(r_k) = \begin{cases} c_1 + (c_2 - c_1)\exp(r_k), \\ \qquad \text{if } r_k \le 0, \\ \dfrac{1-\beta_1\exp(\eta_2)}{1-\exp(\eta_2)} - \dfrac{(1-\beta_1)\exp(\eta_2)}{1-\exp(\eta_2)}\exp((r_k - \eta_2)), \\ \qquad \text{if } 0 < r_k < \eta_2, \\ \beta_2, \\ \text{if } \eta_2 \le r_k \le 2 - \eta_2, \\ \beta_3 + (\beta_2 - \beta_3)\exp\left(-\left(\dfrac{r_k + \eta_2 - 2}{\eta_2 - 2}\right)^2\right), \\ \qquad \text{if } r_k > 2 - \eta_2, \end{cases} \tag{15}$$

where β_1, β_2, β_3, c_1, c_2, and η_2 are constants.

The L-function generalizes the R-function and the Λ-function. It contains some favorable features of the R-function [19] and the Λ-function [2].

Now describe the new self-adaptive trust region algorithm with improved update rules.

Algorithm 2. One has the following.

Step 1. Given $x_0 \in \mathbb{R}^n$, $B_0 \in \mathbb{R}^{n\times n}$, $0 < \eta < \eta_2 < 1$, $0 < c_1 < c_2 < 1$, $0 < \beta_1 \le \beta_3 < 1 \le \beta_2$, and $\Delta_0 > 0$; $\varepsilon \ge 0$; set $k := 0$.

Step 2. If $\|g_k\| \le \varepsilon$ or $f(x_k) - f(x_{k+1}) \le \varepsilon \max\{1, |f(x_k)|\}$, stop. Otherwise solve subproblem (2) to get d_k.

Step 3. Compute

$$r_k = \frac{f(x_k) - f(x_k + d_k)}{q_k(0) - q_k(d_k)}. \tag{16}$$

Compute x_{k+1} as follows:

$$x_{k+1} = \begin{cases} x_k + d_k, & \text{if } r_k > \eta, \\ x_k, & \text{otherwise.} \end{cases} \tag{17}$$

Update the trust region radius

$$\Delta_{k+1} = L(r_k)\Delta_k, \tag{18}$$

where $L(r_k)$ is defined by (15).

Step 4. Compute g_{k+1} and B_{k+1}; set $k := k + 1$; go to step 2.

3. Convergence of Algorithm 2

In the section, we investigate the convergence properties of Algorithm 2. Since it can be considered as a variant of the basic trust region method of Conn et al. [1], we expect similar results and significant similarities in their proofs under the following assumptions.

Assumption 3. Consider the following.

(i) The sequence $\{B_k\}$ is uniformly bounded in norm; that is $\|B_k\| \le M$, for some constant M.

(ii) The function f is bounded on the level set $S = \{x \mid f(x) \le f(x_0)\}$.

(iii) The solution d_k of the subproblem (2) satisfies

$$q_k(0) - q_k(d_k) \ge \sigma \|g_k\| \min\left\{\Delta_k, \frac{\|g_k\|}{\|B_k\|}\right\}, \tag{19}$$

where $\sigma \in (0, 1]$.

(iv) The solution d_k of the subproblem (2) satisfies

$$\|d_k\| \le \bar{\eta}\Delta_k, \tag{20}$$

for positive constant $\bar{\eta} \ge 1$.

Lemma 4. *Suppose that Assumption 3 holds. Then*

$$|f(x_k + d_k) - q(d_k)| \le \frac{1}{2}M\|d_k\|^2 + C(\|d_k\|)\|d_k\|, \tag{21}$$

where $C(\|d_k\|)$ arbitrarily decreases with d_k decreasing.

Proof. Since from Taylor theorem, we have that

$$f(x_k + d_k) = f(x_k) + g_k^T d_k + \int_0^1 [\nabla f(x_k + td_k) - \nabla f(x_k)]^T d_k dt, \tag{22}$$

it follows from the definition of $q_k(d)$ in (2) that

$$\begin{aligned} &|f(x_k + d_k) - q_k(d_k)| \\ &= \left|\frac{1}{2}d_k^T B_k d_k - \int_0^1 [\nabla f(x_k + td_k) - \nabla f(x_k)]^T d_k dt\right| \\ &\le \frac{1}{2}M\|d_k\|^2 + C(\|d_k\|)\|d_k\|. \end{aligned} \tag{23}$$

□

By Algorithm 2, we are capable of showing that the iteration must be very successful but not too successful if the trust region radius is sufficiently small enough and also that the trust region radius has to increase in this case. The following lemma's proof is a bit different from the proof of Theorem 6.4.2 in [1].

Lemma 5. *Suppose that Assumption 3 holds. If $g_k \ne 0$ and there exists a constant $\bar{\Delta} > 0$ such that*

$$\Delta_k \le \bar{\Delta}, \tag{24}$$

then

$$\Delta_{k+1} \ge \Delta_k. \tag{25}$$

Proof. Using Assumption 3 and $g_k \ne 0$ and assuming that there is $\varepsilon > 0$ such that $\|g_k\| \ge \varepsilon$, we obtain that

$$q_k(0) - q_k(d_k) \ge \sigma\|g_k\| \min\left\{\Delta_k, \frac{g_k}{\|B_k\|}\right\} \ge \sigma\varepsilon \min\left(\Delta_k, \frac{\varepsilon}{M}\right). \tag{26}$$

Combining (21) and (26), we have

$$\begin{aligned} |r_k - 1| &= \left|\frac{f(x_k + d_k) - q(d_k)}{q_k(0) - q_k(d_k)}\right| \\ &\le \frac{(1/2)M\|d_k\|^2 + C(\|d_k\|)\|d_k\|}{\sigma\varepsilon \min\{\Delta_k, \varepsilon/M\}} \\ &\le \frac{\bar{\eta}\Delta_k (M\bar{\eta}\Delta_k + 2C(\|d_k\|))}{\sigma\varepsilon \min\{\Delta_k, \varepsilon/M\}}. \end{aligned} \tag{27}$$

By (24), we can choose sufficient small $\bar{\Delta}$, such that

$$\Delta_k \le \bar{\Delta} \le \frac{\varepsilon}{M}, \tag{28}$$

$$M\bar{\eta}\Delta_k + 2C(\|d_k\|) \le (1 - \eta_2)\sigma\frac{\varepsilon}{\bar{\eta}}. \tag{29}$$

Using (27) and (28), we have $|r_k - 1| \leq 1 - \eta_2$; that is, $\eta_2 \leq r_k \leq 2 - \eta_2$. Since $2 - \eta_2 \leq \eta_3$, then $\eta_2 \leq r_k \leq \eta_3$. And so, by Algorithm 2, we have $\Delta_{k+1} \geq \Delta_k$, where Δ_k falls below the threshold $\overline{\Delta}$. □

The proof of Lemma 5 efficiently uses the conditions $r_k \leq 2 - \eta_2 \leq \eta_3$ to explain the case of too successful iteration, as distinguished from the proof of Theorem 6.4.2 in [1].

Theorem 6. *Suppose that Assumption 3 holds. Let the sequence* $\{x_k\}$ *be generated by Algorithm 2. Then*

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

Proof. Assume, for the purpose of deriving a contradiction, that, for all k,

$$\|g_k\| \geq \varepsilon. \tag{31}$$

Suppose that there is an infinite iteration subsequence such that $r_k \geq \eta_2$. Using Algorithm 2 and (21), we have

$$\begin{aligned} f(x_k) - f(x_{k+1}) &\geq \eta_2 [q_k(0) - q_k(d_k)] \\ &\geq \sigma\eta_2 \|g_k\| \min\left\{\Delta_k, \frac{\|g_k\|}{\|B_k\|}\right\} \\ &\geq \sigma\eta_2\varepsilon \min\left\{\Delta_k, \frac{\varepsilon}{\beta}\right\}, \end{aligned} \tag{32}$$

where $\beta = \max\{1 + \|B_k\|\}$. Since $\{f(x_k)\}$ is bounded below, then

$$\lim_{k\to\infty} \Delta_k = 0, \tag{33}$$

which contradicts (25). Hence (30) holds. □

In our strategy of the trust region radius' adjustment, most of the iterations are indeed very successful but not too successful; the trust region constraint becomes irrelevant in the local subproblem. Therefore, superlinear convergence of trust region algorithm is preserved by the proposed self-adaptive radius update.

4. Numerical Experiments

In this section, we present preliminary numerical results to illustrate that the performance of Algorithm 2, denoted by LATR, the basic trust region method in [1], denoted by BTR, and the parameters needed in (9) are chosen that $\beta_1 = 0.5$, $\beta_2 = 2.0$, $\eta_1 = 0.25$, and $\eta_2 = 0.75$; the adaptive trust region method in [19], denoted by RATR, and the parameters needed in (12) are chosen that $\alpha_1 = \alpha_2 = 0.01$, $\beta = 0.1$, $M = 5$, and $\eta_1 = 0.25$; and the adaptive trust region method in [2], denoted by ΛATR, and the parameters needed in (12) are chosen that $\beta_1 = 0.5$, $\beta_2 = 2$, $\beta_3 = 1.01$, and $\eta_1 = 0.95$. In Algorithm 2, $\eta = 0.01$, $\beta_1 = 0.5$, $\beta_2 = 2$, $\beta_3 = 0.7$, $c_1 = 0.12$, $c_2 = 0.14$, and $\eta_2 = 0.75$. All tests are implemented by using MATLAB R2012a on a PC with CPU 2.67 GHz and 8.00 GB RAM. The first eleven problems are taken from [22]; others are from the CUTEr collection [15, 23]. The discrete Newton method is used to update the approximate Hessian matrix B_k. To stabilize the algorithms, the approximate Hessian matrix B_k can be chosen as follows:

$$B_k = \overline{B}_k + \min\left\{1, 0.5\|\nabla f(x_k)\|^2\right\} I, \tag{34}$$

where $\overline{B}_k$ is obtained by forward difference formula at x_k (see [20]) and I is an identity matrix.

In all trust region algorithms in this paper, the trial step d_k is computed by CG-Steihaug algorithm in [20]. The iteration is terminated by the following condition:

$$\|g_k\|_\infty \leq 10^{-5}, \tag{35}$$

except for problem Waston, which will exceed 500 iterations. For the problem, the stopping criterion is

$$|f(x_{k+1}) - f(x_k)| \leq 10^{-5} \max\{1.0, |f(x_k)|\}. \tag{36}$$

In Table 1, we give the dimension (Dim) of each test problem, the number of function evaluations (nf), and the number of gradient evaluations (ng). In many cases, algorithm LATR is superior than algorithms BTR, RATR, and ΛATR, especially for solving problems (1), (7), (8), (13), (19), and (28), although the numbers of gradient evaluations are a bit more than others sometimes. Further result is shown by Figure 3, which is characterized by means of performance profile proposed in [24]. Consider the following performance profile function:

$$\psi_s(\tau) = \frac{1}{n_p}\text{size}\left\{p : 1 \leq p \leq n_p, \log_2\left(r_{p,s}\right) \leq \tau\right\}, \tag{37}$$

where $\tau \geq 0$ and

$$r_{p,s} = \frac{N_{p,s}}{\min\left\{N_{p,i} : 1 \leq i \leq n_s\right\}} \tag{38}$$

is the performance ratio of a solver s on a problem; n_p denotes the number of the tested problems, n_s the number of the solvers, and $N_{p,i}$ the number of the function evaluations (or the CPU time, the number of gradient evaluations, number of iterations, etc.) required to solve the problem p by the solver i.

From Table 1, we know that $n_s = 4$ and $n_p = 91$; then performance profile is given on the sum of the number of function and gradient evaluations to solve the problem. As we can see on Figure 3, the new self-adaptive algorithm is superior to the other three algorithms.

5. Conclusion

This paper presents a new self-adaptive trust region algorithm according to the new self-adaptive radius update rule. As the iteration istoo successful, we suggest reducing the trust region radius with the new rules. The convergence properties of the method are established under reasonable assumptions. Numerical experiments show that the new algorithm for solving unconstrained optimization problems is significant and robust.

For future research, we should investigate how to select an appropriate L-function to conduct numerical experiments.

Table 1: Numerical comparisons of BTR, RATR, ΛATR, and LATR.

Problem	Dim	BTR		RATR		ΛATR		LATR	
		nf	ng	nf	ng	nf	ng	nf	ng
(1) Trigonometric	200	214	17	205	10	—	—	78	9
(2) Extended Powell singular	200	28	20	26	20	32	21	30	24
(3) Schittkowski function 302	200	239	160	363	275	117	24	83	65
(4) Linear function full rank	200	23	4	21	4	15	3	11	7
(5) Watson	200	30	14	22	10	16	10	18	11
	300	39	15	115	78	73	49	67	48
	400	84	39	27	12	38	25	107	78
	500	95	43	29	14	68	48	39	26
(6) Nearly separable	200	41	21	27	15	23	16	24	17
	300	46	25	34	18	24	16	23	16
	400	52	25	39	23	33	22	26	19
	500	—	—	—	—	—	—	37	26
(7) Yang tridiagonal	200	67	40	61	44	99	32	51	41
	300	151	113	72	57	105	38	74	61
	400	133	93	141	106	151	68	109	90
	500	148	97	120	95	257	149	109	90
(8) Allgower	200	40	23	53	38	57	2	83	69
	300	18	1	17	1	24	1	7	1
	400	25	15	17	3	56	2	21	17
	500	20	3	17	3	52	3	9	2
(9) Linear function rank 1	200	101	42	62	27	285	30	39	24
	300	102	40	66	25	300	30	35	16
	400	110	45	69	25	319	32	39	20
	500	108	42	70	25	317	31	47	32
(10) Linear function rank 1 with zero columns and rows	200	64	3	60	2	21	2	22	2
	300	67	3	64	3	21	1	21	1
	400	66	1	65	1	29	5	22	1
	500	78	6	67	2	24	2	26	4
(11) Discrete integral equation	200	54	19	24	4	39	14	27	18
	300	31	5	70	46	42	22	30	28
	400	44	12	43	32	32	16	31	29
	500	49	15	37	17	45	23	31	29
(12) CRAGGLVY	200	36	9	35	9	22	10	19	9
(13) GENHUMPS	200	26	0	26	0	74	0	9	0
(14) BROYDNBD	200	32	7	20	14	47	0	17	15
(15) PENALTY	200	71	33	69	33	59	33	59	37
	300	75	35	74	35	63	38	59	35
	400	74	33	76	35	64	36	61	37
	500	77	35	80	37	58	36	62	38
(16) PENALTY2	200	46	8	42	8	88	9	25	11
	300	149	72	159	129	78	50	150	116
	400	434	208	—	—	—	—	465	390
	500	—	—	330	228	—	—	—	—
(17) CHEBYQAD	200	75	55	79	70	39	7	83	67
	300	28	12	83	70	41	6	89	70
	400	95	63	99	85	108	64	104	82
	500	32	16	37	22	119	52	103	83

Table 1: Continued.

Problem	Dim	BTR		RATR		ΛATR		LATR	
		nf	ng	nf	ng	nf	ng	nf	ng
(18) GENROSE	200	194	161	253	248	265	172	241	205
	300	308	235	381	373	394	248	357	303
	400	405	301	487	479	499	323	468	399
	500	—	—	—	—	—	—	—	—
(19) INTEGREQ	200	15	1	14	1	214	0	6	1
	300	15	1	14	1	206	0	6	1
	400	15	1	14	1	203	0	6	1
	500	15	1	14	1	201	0	6	1
(20) FLETCHCR	200	113	88	151	146	157	91	158	132
	300	147	122	218	213	224	132	233	195
	400	183	157	292	284	295	175	309	258
	500	217	190	357	351	362	211	384	321
(21) ARGLINB	200	6	5	6	5	6	5	6	5
	300	7	6	7	6	7	6	7	6
	400	7	6	7	6	300	5	7	6
	500	7	6	7	6	424	5	7	6
(22) NONDIA	200	7	6	7	6	93	6	7	6
	300	7	6	7	6	7	6	7	6
	400	6	5	6	5	6	5	6	5
	500	6	5	6	5	6	5	6	5
(23) EG2	200	14	9	11	9	59	23	10	8
	300	14	9	12	9	60	27	6	5
	400	9	7	7	6	180	30	8	6
	500	12	5	17	13	111	34	14	11
(24) CURLY20	200	13	11	19	17	16	14	11	10
	300	13	11	21	18	17	15	13	12
	400	10	9	11	11	11	10	9	9
	500	10	9	11	11	11	10	9	9
(25) CUBE	200	52	20	39	25	95	16	37	29
	300	50	16	77	65	98	20	40	32
	400	52	16	45	31	83	16	35	28
	500	52	16	52	38	128	14	35	28
(26) EXPLIN1	200	30	2	28	2	12	3	13	4
	300	29	1	28	1	33	4	13	4
	400	31	2	29	2	23	4	21	16
	500	32	2	30	2	15	3	13	4
(27) SINQUAD	200	68	55	18	17	119	14	19	17
	300	186	175	16	15	225	20	64	60
	400	25	12	21	19	224	25	48	45
	500	31	15	43	41	229	40	35	33
(28) LIARWHD	200	29	1	28	1	50	2	14	4
	300	32	2	29	2	18	2	12	2
	400	30	1	30	1	30	2	12	1
	500	30	1	30	1	33	2	11	1

— means that the algorithm reaches 500 iterations.

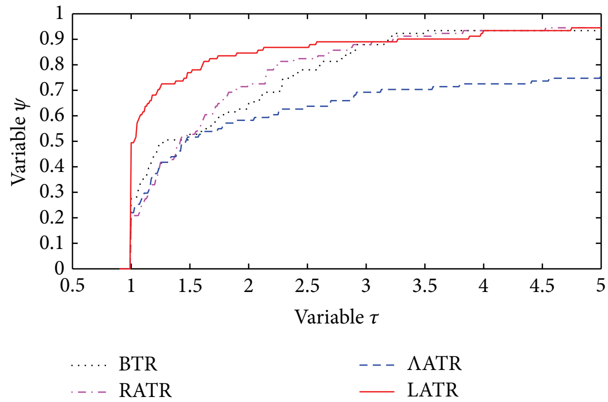

Figure 3: Performance profile comparing the sum of the number of function and gradient evaluations.

Conflict of Interests

The authors declare that they have no financial nor personal relationships with other people or organizations that can inappropriately influence their work; there is no professional nor another personal interest of any nature or kind in any product, service, and/or company that could be construed as influencing the position presented in or the review of the paper.

Acknowledgments

This research is partly supported by Chinese NSF under Grant 11171003, Chinese Ministry of Education, Science and Technology Research Key Project no. 2011039, and Department of Education of Jilin Province Project no. 2009158 and no. 201215102.

References

[1] A. R. Conn, N. I. M. Gould, and P. L. Toint, *Trust-Region Methods*, MPS/SIAM Series on Optimization, Society for Industrial and Applied Mathematics (SIAM); Mathematical Programming Society (MPS), Philadelphia, Pa, USA, 2000.

[2] J. M. B. Walmag and E. J. M. Delhez, "A note on trust-region radius update," *SIAM Journal on Optimization*, vol. 16, no. 2, pp. 548–562, 2005.

[3] A. Sartenaer, "Automatic determination of an initial trust region in nonlinear programming," *SIAM Journal on Scientific Computing*, vol. 18, no. 6, pp. 1788–1803, 1997.

[4] X. Zhang, J. Zhang, and L. Liao, "An adaptive trust region method and its convergence," *Science in China A*, vol. 45, no. 5, pp. 620–631, 2002.

[5] X. S. Zhang, "Trust region method in neural network," *Acta Mathematicae Applicatae Sinica*, vol. 12, pp. 1–10, 1996.

[6] Z.-J. Shi and J.-H. Guo, "A new trust region method for unconstrained optimization," *Journal of Computational and Applied Mathematics*, vol. 213, no. 2, pp. 509–520, 2008.

[7] J. Fu, W. Sun, and R. J. B. de Sampaio, "An adaptive approach of conic trust-region method for unconstrained optimization problems," *Journal of Applied Mathematics & Computing*, vol. 19, no. 1-2, pp. 165–177, 2005.

[8] Z. Sang and Q. Sun, "A self-adaptive trust region method with line search based on a simple subproblem model," *Journal of Computational and Applied Mathematics*, vol. 232, no. 2, pp. 514–522, 2009.

[9] Z. Yu and Q. Li, "A self-adaptive trust region method for the extended linear complementarity problems," *Applications of Mathematics*, vol. 54, no. 1, pp. 53–65, 2009.

[10] N. I. M. Gould, D. Orban, A. Sartenaer, and P. L. Toint, "Sensitivity of trust-region algorithms to their parameters," *4OR. A Quarterly Journal of Operations Research*, vol. 3, no. 3, pp. 227–241, 2005.

[11] J.-L. Zhang and X.-S. Zhang, "A nonmonotone adaptive trust region method and its convergence," *Computers & Mathematics with Applications*, vol. 45, no. 10-11, pp. 1469–1477, 2003.

[12] J. Fu and W. Sun, "Nonmonotone adaptive trust-region method for unconstrained optimization problems," *Applied Mathematics and Computation*, vol. 163, no. 1, pp. 489–504, 2005.

[13] J. Zhang, K. Zhang, and S. Qu, "A nonmonotone adaptive trust region method for unconstrained optimization based on conic model," *Applied Mathematics and Computation*, vol. 217, no. 8, pp. 4265–4273, 2010.

[14] M. Ahookhosh and K. Amini, "A nonmonotone trust region method with adaptive radius for unconstrained optimization problems," *Computers & Mathematics with Applications*, vol. 60, no. 3, pp. 411–422, 2010.

[15] N. Andrei, "An unconstrained optimization test functions collection," *Advanced Modeling and Optimization*, vol. 10, no. 1, pp. 147–161, 2008.

[16] Z. Shi and S. Wang, "Nonmonotone adaptive trust region method," *European Journal of Operational Research*, vol. 208, no. 1, pp. 28–36, 2011.

[17] Z. Sang and Q. Sun, "A new non-monotone self-adaptive trust region method for unconstrained optimization," *Journal of Applied Mathematics and Computing*, vol. 35, no. 1-2, pp. 53–62, 2011.

[18] Z. Cui and B. Wu, "A new modified nonmonotone adaptive trust region method for unconstrained optimization," *Computational Optimization and Applications*, vol. 53, no. 3, pp. 795–806, 2012.

[19] L. Hei, "A self-adaptive trust region algorithm," *Journal of Computational Mathematics*, vol. 21, no. 2, pp. 229–236, 2003.

[20] J. Nocedal and S. T. Wright, *Numerical Optimization*, Springer, Berlin, Germany, 2000.

[21] W. Sun and Y.-X. Yuan, *Optimization Theory and Methods, Nonlinear programming*, vol. 1 of *Springer Optimization and Its Applications*, Springer, New York, NY, USA, 2006.

[22] J. J. Moré, B. S. Garbow, and K. E. Hillstrom, "Testing unconstrained optimization software," *Association for Computing Machinery. Transactions on Mathematical Software*, vol. 7, no. 1, pp. 17–41, 1981.

[23] N. I. M. Gould, D. Orban, and P. L. Toint, "GALAHAD, a library of thread-safe Fortran 90 packages for large-scale nonlinear optimization," *Association for Computing Machinery. Transactions on Mathematical Software*, vol. 29, no. 4, pp. 353–372, 2003.

[24] E. D. Dolan and J. J. Moré, "Benchmarking optimization software with performance profiles," *Mathematical Programming*, vol. 9, pp. 1201–1213, 2002.