A Larger Step-Size Line Search Self-adaptive Trust Region Method and Its Convergence

Lin Feng
School of Mathematics and Finance, Chongqing University of Arts and Sciences, Chongqing Yong chuan 402160, China

Abstract—In this paper, we give a trust region method for unconstrained optimization, in which its radius is automatically updated by the current information. And when the trial step is rejected, we apply a larger step-size line search to the method. The primary aim is to update the radius depending on the problem itself at each iteration. The next aim is to avoid resolving the subproblem, which decreases computation load. The global convergence as well as the local convergence of the method is analyzed under certain conditions.

Keywords—unconstrained optimization; trust region method; self-adaptive; larger step-size line search; convergence

I. INTRODUCTION

We consider the unconstrained optimization problems

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

where $f(x) : \mathbb{R}^n \to \mathbb{R}^1$ is a twice continuously differentiable function, which is bounded below. Throughout the paper, we adopt the notations as follows.

$\| \|$ denotes the Euclidean norm on $\mathbb{R}^n$.

$g(x) \in \mathbb{R}^n$ is the gradient of $f(x)$ at $x$.

$H(x) \in \mathbb{R}^{n \times n}$ is the Hessian matrix of $f(x)$ at $x$.

$\{ x_k \}$ is a sequence of iterative point generated by some algorithm and $f_k = f(x_k)$, $g_k = g(x_k)$, $H_k = H(x_k)$.

$B_k \in \mathbb{R}^{n \times n}$ is a symmetric matrix which is either the Hessian matrix or its approximation of $f(x)$ at $x_k$.

$\bar{B}_k$ is a safely positive definite matrix which satisfies a modified Cholesky factorization from Schnabel and Eskow [1]. $\bar{B}_k = B_k + E_k$, where $E_k$ is a diagonal matrix which makes $\bar{B}_k$ safely positive definite, and $E_k = 0$ if $B_k$ is positive definite.

Trust region methods are a class of effective methods for solving (1). They have the stronger global convergence and faster local convergence rate and can be applied to ill-conditioned problems, so many researchers have studied them [2-4]. A trust region method works as follows. At every iteration, solving the following trust region subproblem:

$$\min_{s \in \mathbb{R}^n} q_k(s) = g_k^T s + \frac{1}{2} s^T B_k s \text{ s.t. } \| s \| \leq \Delta_k,$$

(2)

where $s = x - x_k$, $g_k = \nabla f(x_k)$, $\Delta_k > 0$ is the trust region radius.

At point $x_k$, let $s_k$ be the solution to (2), then the predicted reduction of $f(x)$ is $\text{Pr} \Delta_k = q_k(0) - q_k(s_k)$, the actual reduction is $\Delta \text{re} \Delta_k = f(x_k) - f(x_k + s_k)$. The ratio between actual reduction and predicted reduction is $r_k = \frac{\Delta \text{re} \Delta_k}{\text{Pr} \Delta_k}$.

As is known, we usually use $r_k$ to decide $x_{k+1} = x_k + s_k$ whether as the new iterate point or not and how to adjust $\Delta_k$. If $x_{k+1} = x_k + s_k$ is accepted as a new iterate point, we increase $\Delta_k$; otherwise we resolve (2) with a reduced $\Delta_k$. That is, we have

$$x_{k+1} = \begin{cases} x_k + s_k, & r_k \geq \mu, \\ x_k, & r_k < \mu, \end{cases}$$

where $\mu \in (0, 1)$ is a constant, and we have

$$\Delta_{k+1} = \begin{cases} c_1 \Delta_k, & r_k \leq \mu, \\ c_2 \Delta_k, & r_k \geq \mu, \\ \Delta_k, & r_k \leq \mu, \end{cases}$$

(3)

where $0 \leq \mu_1 < \mu_2 < 1, 0 < c_1 < 1 < c_2$ are constants.

(3) shows that we adjust $\Delta_k$ only by increasing or reducing the initial trust region radius at a constant ratio according to $r_k$ and without using the problems (1) itself, which reduces the efficiency of the methods. Based on this fact, more and more self-adaptive trust region methods are presented [5-7]. Sartenaer[8] proposed a method for automatically determining the initial trust region radius. By using the idea of Sartenaer [8], Zhang et al.[9] put forward an adaptive trust region method, where

$$\Delta_k = c^T g_k^T M_k$$

(4)

(4) is a constant, $P$ is a

Copyright © 2016, the Authors. Published by Atlantis Press. 
This is an open access article under the CC BY-NC license (http://creativecommons.org/licenses/by-nc/4.0/).
nonnegative integer variables, \( M_k = \|B_k \| \). If the trial step \( s^k \) at the last iteration is accepted, set \( p = 0 \) and \( \Delta_k = \|s^k \|/M_k \), otherwise set \( p = p + 1 \). Sang et al.[10] gave a trust region method in which the trust region radius \( \Delta_k \) converging to zero. The numerical results show that the adaptive trust region algorithms improve the accuracy of solution and decrease the number of iteration for some functions.

In the traditional trust region algorithms, when the trial step \( s^k \) is not accepted, the trust region subproblem need resolving which increase the amount of calculation. Jorge Nocedal et al. [11] first proposed the idea of combining trust region method with line search method to establish a new method. In [12], they delivered a trust region method with a line search where the trial step is accepted at each iteration and no subproblems are resolved, decreasing the amount of calculation. In this paper, the trial step \( s^k \) is always a descent direction by the method in [12], and if the trial step \( s^k \) is rejected, a larger step-size line search in [13] is performed along \( s^k \) to achieve a new iterative point.

Based on the work of [12], [9] and [10], we present a new adaptive trust region method, where we let

\[
\Delta_k = \mu_k \|B_k^{-1}g_k \| \tag{4}
\]

which obliges the adjustment of \( \Delta_k \) to depend on the problem itself at each iteration.

II. ADAPTIVE TRUST REGION METHOD

In this section, a new adaptive trust region method is put forward. If \( r_k \geq \mu \), we set \( x_{k+1} = x_k + s^k \); otherwise a step size \( \lambda^k \) is obtained by the following larger step-size line search in [13].

\[
f(x_k + \lambda^k d_k) - f(x_k) \leq \rho \lambda^k g_k^T d_k + \frac{1}{2} \lambda^k d_k^T B_k d_k \tag{5}
\]

where \( \lambda^k \) is the maximum value among \( h_k, \beta h_k , \beta^2 h_k, \ldots \).

\[
0 < \rho, \beta < 1, h_k = \frac{g_k^T d_k}{d_k^T B_k d_k} \quad \text{and} \quad x_{k+1} = x_k + \lambda^k d_k \tag{6}
\]

Now we present a description of our adaptive trust region algorithm.

Algorithm 1.

Step 1 Select \( x_0 \in R^n, \delta > 0, \Delta_0 > 0, 0 < \rho, \beta < 1, h_0 > 0, \mu_1 > 0, B_0 = I, 0 < c_1 < 1 < c_2, 0 < c_3 < 1, 0 \leq \eta_1 < \eta_2 < 1, k := 0 \).

Step 2 Compute \( s^k \). If \( \|s^k \| \leq \epsilon \), stop, then \( x^* = x_k \); otherwise go to Step 3.

Step 3 Solve the subproblem (2) to get the approximate solution \( s^k \).

Step 4 Calculate \( r_k \). If \( r_k \geq \eta_1 \), then \( x_{k+1} = x_k + s^k \). If \( \eta_1 \leq r_k \leq \eta_2 \), then \( \mu_{k+1} = \mu_k \); if \( r_k > \eta_2 \) and \( \|s^k \| \geq c_1 \Delta_k \), then \( \mu_{k+1} = c_2 \mu_k \). And go to Step 6.

Step 5 Find \( \lambda^k \) to meet (5), and set \( x_{k+1} = x_k + \lambda^k s^k \), \( \mu_{k+1} = c_1 \mu_k \).

Step 6 Modify \( B_k \) and get \( B_{k+1} \). Make use of (4) to calculate \( \Delta_{k+1} \).

Let \( k := k + 1 \), then go to Step 2.

Note: (i) Let \( s^k \) be the solution to the subproblem (2), then

\[
\text{Proof:} \quad \text{The result is proved by induction for } k \.
\]

If \( k = 0 \), then \( f(x_0) \leq f(x_0) \), so \( x_0 \in L(x_0) \). The result is true.

Now we prove if \( x_k \in L(x_0) \), then \( x_{k+1} \in L(x_0) \).
If \( x_i \in L(x_0) \), then \( f(x_i) \leq f(x_0) \).

(i) If \( k \in I \), we have \( r_i < \eta_i \).

By Step 5 in Algorithm 1, we have
\[
f(x_i) - f(x_{i+1}) = f(x_i) - f(x_i + \Delta x_i) \geq -\rho x_i^T g_i x_i \geq 0.
\]
Thus, \( f(x_{i+1}) \leq f(x_i) \leq f(x_0) \), this indicate \( f(x_{i+1}) \leq f(x_0) \).

(ii) If \( k \in J \), then \( r_i \geq \eta_i \). By (6), we can get
\[
f(x_i) - f(x_{i+1}) \geq \eta_i(q_i(0) - q_i(s_i)) > 0.
\]
Thus, \( f(x_{i+1}) < f(x_i) \leq f(x_0) \), it follows that \( f(x_{i+1}) \leq f(x_0) \).

By (i) and (ii), we know, if \( x_i \in L(x_0) \), then \( x_{i+1} \in L(x_0) \).

From the principle of induction it follows that the result is true.

Lemma 2. If \( \{x_i\} \) is a point sequence generated by the Algorithm 1, then \( \{f(x_i)\} \) is monotonically decreasing and convergent.

Proof. From the proof of Lemma 1, it is known that \( f(x_{i+1}) \leq f(x_i) \), \( \forall k \).

This implies \( \{f(x_i)\} \) is monotonically decreasing.

And by A1, we can obtain \( \{f(x_i)\} \) is convergent.

Lemma 3. \( \text{Ares}_k - \text{Prees}_k = O(\|s_i\|) \).

Lemma 4. Suppose that A3 and A1 hold, \( \{x_i\} \) is an infinite point sequence generated by the Algorithm 1 and \( \|g_i\| > \varepsilon \),
\( \varepsilon \in (0, 1) \) is a constant, then there exists a non-negative integer \( P \) such that for all \( k \) we have
\[
r_{k+P} \geq \eta_i.
\]

Proof. Assume that there exists a positive integer \( k_0 \), for \( \forall p \), we have
\[
r_{k_0+p} < \eta_i, p = 0, 1, 2, \ldots.
\]

By A1 and Step 5 in algorithm 1, we can get
\[
\Delta_{k, k+1} \rightarrow 0, P \rightarrow \infty.
\]

Therefore \( \|k_{k_0+P} \| \rightarrow 0, P \rightarrow \infty \).

Thus by Lemma 3 and (6) for sufficiently large \( P \), we have
\[
\left| r_{k_0+P} - \right| = \frac{\text{Ared}_{k_0+p} - \text{Preed}_{k_0+p}}{\text{Preed}_{k_0+p}} \leq O(\|s_{k_0+p}\|) \leq O(\|s_{k_0+p}\|) \rightarrow 0.
\]

This suggests for sufficiently large \( P \), \( r_{k_0+p} \geq \eta_i \) holds, this contradict with (7). We complete the proof.

Theorem 1. Suppose A1 holds and there exists \( \delta_i > 0 \), such that \( \|g_i\| \leq \delta_i \), \( \{x_i\} \) is the point sequence generated by the Algorithm 1, then
\[
\lim_{k \to \infty} \|g_k\| = 0.
\]

Proof. Suppose (9) is not true, then there exists a constant \( \varepsilon > 0 \) such that
\[
\|g_k\| \geq \varepsilon, \forall k.
\]

Now we pay our attention to
\[
\lim_{k \to \infty} \mu_k = 0.
\]

By Lemma 2, (6), (10) and A2, we have
\[
+\infty > \sum_{i=1}^{\infty} \left| f(x_i) - f(x_{i+1}) \right| \geq \sum_{k=0}^{\infty} \left| f(x_k) - f(x_{k+1}) \right|
\]
\[
\geq \sum_{k=0}^{\infty} \eta_i(q_i(0) - q_i(s_i)) \geq \sum_{k=0}^{\infty} \frac{1}{2} \|g_k\| \min \left\{ \Delta_k, \|g_k\| / \|s_k\| \right\}
\]
\[
= \sum_{k=0}^{\infty} \frac{1}{2} \eta_i \|g_k\| \min \left\{ \mu_k \|B_k^{-1} g_k\|, \|g_k\| / \|s_k\| \right\}
\]
\[
\geq \sum_{k=0}^{\infty} \frac{1}{2} \eta \varepsilon \min \left\{ \mu_k \|B_k^{-1} g_k\|, \|g_k\| / \|s_k\| \right\} < +\infty,
\]

which indicates this implies
\[
\lim_{k \to \infty} \mu_k \|B_k^{-1} g_k\| = 0.
\]

If \( J \) is a finite set, then \( \mu_{k+1} = c_i \mu_k \) for sufficiently large \( k \),
thus we obtain \( \mu_k = 0 \).

If \( J \) is an infinite set, we have \( \|B_k^{-1} g_k\| = 0 \), then
Further \( g_k = 0 \), this contradict with (10), hence
\[
\left\| B_k^{-1} g_k \right\| \neq 0. \quad (13)
\]
And by (12), we get \( \lim_{k \to \infty} \mu_k = 0. \)

From the above, we complete the proof of \( \lim_{k \to \infty} \mu_k = 0. \)

By (12) and Steps 4 in Algorithm 1, we can deduce \( \lim_{k \to \infty} \mu_k = 0. \)

Then, it follows that \( \Delta_k \to 0, k \to \infty \) from (4).

Further, \( \| s_k \| \to 0, k \to \infty. \)

Hence, for sufficiently large \( k \), we have
\[
\left| r_k - 1 \right| = \frac{\text{Ared}_k - \text{Pr ed}_k}{\text{Pr ed}_k} \leq \frac{\sigma \left( \| x_k \| + \alpha \left( \| y_k \| - \| B_k \| \right) \right) \Delta_k}{\left( \Delta_k \right)} \to 0,
\]
\[
\frac{1}{2} \left\| s_k \right\| \min \left\{ \Delta_k, \left\| \frac{g_k}{\| B_k \|} \right\| \right\} \to 0,
\]
This means for sufficiently large \( k \), \( \left| r_k - 1 \right| \to 0. \) This also implies for sufficiently large \( k \), \( \mu_{k+1} \geq \mu_k \), this is a contradiction with \( \lim_{k \to \infty} \mu_k = 0. \) Thus, the proof of Theorem 1 is completed.

The superlinear convergence of the Algorithm 1 is discussed below.

Now we give two hypotheses.

A4: The infinite point sequence \( \{ x_k \} \) generated by Algorithm 1 satisfies
\[
\lim_{k \to \infty} x_k = x^*, \lim_{k \to \infty} \| g_k \|= \| g^* \|= 0
\]
and \( \nabla^2 f(x^*) \) is positive definite.

A5: The sequence \( \{ x_k \} \) generated by Algorithm 1 and \( \{ B_k \} \) satisfy
\[
\lim_{k \to \infty} \left\| B_k - \nabla^2 f(x_k) s_k \right\| = 0.
\]

Theorem 2. Suppose \( \{ x_k \} \) is an infinite point sequence generated by Algorithm 1 and A1-A5 hold, then \( \{ x_k \} \) converges to \( x^* \) superlinearly.

Proof. From that \( \nabla^2 f(x^*) \) is positive definite, we know for sufficiently large \( k \), there exists \( \delta > 0 \) such that
\[
s_k^T B_k s_k \geq \delta \| s_k \|^2.
\]
Because \( x_k \) is the solution to (2), we can get
\[
g_k + (B_k + \lambda_k I)s_k = 0.
\]
Hence, for \( k \in J \), we have
\[
\Pr e_{x_k} = -g_k^T s_k - \frac{1}{2} s_k^T B_k s_k = [(B_k + \lambda_k I)s_k]^T s_k - \frac{1}{2} s_k^T B_k s_k
\]
\[
= \lambda_k \| s_k \|^2 + \frac{1}{2} s_k^T B_k s_k \geq \frac{1}{2} s_k^T B_k s_k + \frac{1}{2} \delta \| s_k \|^2 = \left( \frac{1}{2} \delta + \lambda_k \right) \| s_k \|^2,
\]
\[
A r e_{x_k} = -g_k^T s_k - \frac{1}{2} s_k^T \nabla^2 f(x^*) s_k + \alpha \| s_k \|^2
\]
\[
= \Pr e_{x_k} + \frac{1}{2} \Delta_k + \frac{1}{2} (B_k - \nabla^2 f(x^*)) s_k + \alpha \| s_k \|^2
\]
\[
= \Pr e_{x_k} + \alpha \| s_k \|^2.
\]
On the other hand, for \( k \in I \), we have
\[
g_k + (B_k + \lambda_k I) x_k^* - s_k = 0.
\]
Hence, for \( k \in I \), we obtain
\[
\Pr e_{x_k} = -g_k^T s_k - \frac{1}{2} s_k^T B_k s_k
\]
\[
= (\lambda_k - \frac{1}{2} \delta) s_k^T B_k s_k + \frac{1}{2} s_k^T \nabla^2 f(x^*) s_k + \alpha \| s_k \|^2
\]
\[
A r e_{x_k} = -g_k^T s_k - \frac{1}{2} s_k^T \nabla^2 f(x^*) s_k + \alpha \| s_k \|^2
\]
\[
= \Pr e_{x_k} + \frac{1}{2} s_k^T (B_k - \nabla^2 f(x^*)) s_k + \alpha \| s_k \|^2
\]
\[
= \Pr e_{x_k} + \alpha \| s_k \|^2.
\]
Therefore,
\[
\lim_{k \to \infty} r_k = \frac{\text{Ared}_k}{\Pr e_{x_k}} = 1.
\]

Now we prove \( \{ \mu_k \} \) to be bounded. Suppose \( \mu_k \to \infty \), by (13),
we know that \( \exists \sigma > 0, s \| f(x_k) - f(x^*) \| > \sigma, s \| g_k \| \)
So \( \| y_k \| > c_1 \Delta_k = c_1 \mu_k \| B_k^{-1} g_k \| > c_1 \mu_k \| s_k \|. \)

From \( \nabla^2 f(x^*) \) is positive definite, we know that there exist \( W > w > 0 \) such that
\[
\| x_k - x^* \| \leq f_k - f(x^*) \leq W \| x_k - x^* \|^2,
\]
and \( \| x_k - x^* \| \leq \| s_k \| \leq W \| x_k - x^* \|^2 \) for sufficiently large \( k \).

Hence, \( \| x_k - x^* \| \leq f_k - f(x^*) \geq f(x_k + s_k) - f(x^*) \)
\[
\geq w \| x_k + s_k - x^* \|^2 \geq w \{ f_k - \| x_k - x^* \|^2 \}
\]
\[
\geq w (c_1 \mu_k \sigma w - 1)^2 \| x_k - x^* \|^2,
\]
This shows \( \mu_k \to \infty \) is impossible. This implies
\[ \| s_k \| < c_s \Delta_k < \Delta_k \] for sufficiently large \( k \). Therefore, for sufficiently large \( k \), the trust region constrain is inactive. By the standard results from Dennis and Moré, we declare Algorithms 1 is convergent superlinearly.

IV. CONCLUSIONS

For the unconstrained optimization problems, we put forward a larger step-size line search self-adaptive trust region method. The trust region radius use a new adaptive adjustment strategy, which makes full use of the information at the current iteration point at each iteration. When the trial step is not accepted, a larger step-size line search is performed into the method to get a new iterative point. The method doesn’t resolve the trust region subproblem, so the amount of computation is diminished. Under certain conditions, Global convergence and superlinear convergence of the method are established.

ACKNOWLEDGMENT

The author would like to thank this paper is supported in partial by Scientific Research Subject (NO:Y2015SC) of Chongqing University of Arts and Sciences.

REFERENCE

[1] R.B. Schnabel, and E. Eskow, “A new modified Cholesky factorization,” SIAM J. Sci. Stat. Comput., vol.11, pp.1136-1158, 1990.
[2] Priscila S. Ferreira, Elizabeth W. Karas, and Mael Sachini, “A globally convergent trust-region algorithm for unconstrained derivative-free optimization,” Comp. Appl. Math., vol.34, pp.1075-1103, 2015.
[3] O. Burdakov, L. J. Gong, S. Zikrin, and Y. X. Yuan, “On efficiently combining limited-memory and trust-region techniques,” Math. Prog. Comput., 2016.
[4] Q. Y. Zhou, J. Chen, and Z. W. Xie, “A nonmonotone trust region method based on simple quadratic models,” J. Comput. & Appl. Math., vol.272, pp.107-115, 2014.
[5] F. S. Wang, and C. L. Wang, “An adaptive nonmonotone trust-region method with curvilinear search for minimax problem,” Appl. Math. Comput., vol.219, pp.8033-8041, 2013.
[6] K. Amini, and M. Ahookhosh, “A hybrid of adjustable trust-region and nonmonotone algorithms for unconstrained optimization,” Appl. Math. Modell., vol.38, pp.2601-2612, 2014.
[7] A. Kamandi, K. Amini, and M. Ahookhosh, “An improved adaptive trust-region algorithm,” Optim. Lett., 2016.
[8] A. Sartenaer, “Automatic determination of an initial trust region in nonlinear Programming,” SIAM J. Sci. Comput., vol.18, pp.1788-1803, 1997.
[9] X. S. Zhang, J. L. Zhang, and L. Z. Liao, “An adaptive trust region method and its convergence,” Sci. China (Series A), vol.45, pp.620-631, 2002.
[10] Z. Y. Sang, and Q. Y. Sun, “A self-adaptive trust region method with line search based on a simple subproblem model,” J. Comput. & Appl. Math., vol.232, pp.514-522, 2009.
[11] J. Nocedal, and Y. X. Yuan, “Combining trust region and line search techniques,” Technical Report, NAM06, Department of Computer Science, Northwestern University, Illinois, USA, 1991.
[12] J. Nocedal, and Y. X. Yuan, “Combining trust region and line search techniques,” Advances in Nonlinear Programming, Kluwer, Dordrecht, vol.14, pp.153-175, 1998.
[13] Z. J. Shi, and J. Shen, “New inexact line search method for constrained optimization,” J. Optim. Theory & Appl., vol.127(2), pp.425-445, 2005.