A Journal of Software Engineering and Applications, **2013, 6, 49-52**
doi:10.4236/jsea.2013.65B010 Published Online May 2013 (http://www.scirp.org/journal/jsea)

An Improved Line Search and Trust Region Algorithm

Qinghua Zhou, Yarui Zhang, Xiaoli Zhang

College of Mathematics & Computer Science, Hebei University, Baoding, 071002, China.
Email: qinghua.zhou@gmail.com

Received 2013

ABSTRACT

In this paper, we present a new line search and trust region algorithm for unconstrained optimization problems. The trust region center locates at somewhere in the negative gradient direction with the current best iterative point being on the boundary. By doing these, the trust region subproblems are constructed at a new way different with the traditional ones. Then, we test the efficiency of the new line search and trust region algorithm on some standard benchmarking. The computational results reveal that, for most test problems, the number of function and gradient calculations are reduced significantly.

Keywords: Trust Region Algorithms; Trust Region Subproblem; Line Search; Unconstrained Optimization

1. Introduction

Trust region algorithm is one of the most often used methods for solving the unconstrained optimization problem

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

Generally, a trial step is calculated by solving the trust region subproblem

$$\begin{aligned} &\min_{d\in R^n} g_k^T d + \frac{1}{2} d^T B_k d \equiv \phi_k(d) \\ &s.t. \quad \|d\| \le \Delta_k, \end{aligned} \tag{2}$$

Where $g_k = \nabla f(x_k)$ is the gradient at the current approximate solution, B_k is a $n\times n$ symmetric matrix which approximates the Hessian of f at x_k, Δ_k is the current trust region radius and here $\|\cdot\|$ refers to the 2-norm.

Trust region algorithms are blessed with both strong theoretical convergence properties and effectiveness in practice[1-4]. It has been studied by many researchers. The traditional trust region is normally an area centered at the current iterate x_k as indicated in model (2). In 2003, based on the traditional trust region algorithm, Ma [5] proposed a new trust region algorithm, whose trust region radius is $\Delta_k\|g_k\|$ and whose trust region center is $x_k - \Delta_k g_k$. After carefully studying the idea of traditional trust region algorithms and Ma [5], Zhou et al [6] also proposed an improved trust region based on an algorithm depicted in [7].

The improved trust region is a ball centered at the point $x_k - \Delta_k g_k/\|g_k\|$ and whose radiusis Δ_k. In this paper, we present another new trust region algorithm, which is based on the idea of Zhou et al [6] and to investigate the performance of the algorithms with different trust region subproblems.

Our aim is to improve the algorithm proposed in [6,7] and to make it more effective in practical implementation. The main difference between the algorithm in [7] and our algorithm is that in the former one the trust region subproblem has the model like equation (2), while in our algorithm the trust region subproblem is reconstructed at different trust region center and radius.

The paper is organized as follows. In Section 2, we describe our algorithm for unconstrained optimization problems which combines the new trust region and line search algorithm. In Section 3, primary numerical results are presented. Finally a short discussion about conclusion and future works is given in section 4.

2. The New Line Search and Trust Region Algorithm

2.1. Traditional Line Search and Trust Region Algorithm

In this subsection, we consider the line search and trust region method for the unconstrained optimization problem

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

where $f(x)$, the objective function, is a real-valued continuously differentiable function. Its solution is approximated by iteratively solving the subproblem (2).

Let d_k be the solution of (2). The predicted reduction is defined by the reduction of the approximate model, that is,

$$pred_k = \phi_k(0) - \phi_k(d_k),$$

and the actual reduction is defined by the reduction of the objective function, that is,

$$ared_k = f(x_k) - f(x_k + d_k).$$

The ratio between the two reductions is defined by

$$\gamma_k = \frac{ared_k}{pred_k}. \tag{3}$$

It is well known that the ratio plays a key role in the traditional trust region algorithm to determine whether the trial step is acceptable or not and to adjust the new trust region radius. If the trial step is not successful, we will reject it, reduce the trust region radius and resolve the subproblem (2); otherwise, we will accept the trial step and enlarge the trust region radius.

The next iterate x_{k+1} is determined by the following formula:

$$x_{k+1} = \begin{cases} x_k + d_k & if\ \gamma_k > c_0, \\ x_k & otherwise, \end{cases}$$

where $c_0 \in [0,1)$ is a small constant.

The trust region radius for the next iteration is chosen by the following formula:

$$\Delta_{k+1} \in \begin{cases} \left[c_3\|d_k\|, c_4\Delta_k\right] & if\ \gamma_k \le c_2, \\ \left[\Delta_k, c_1\Delta_k\right] & other\ wise, \end{cases}$$

where $c_i\ (i = 1,2,3,4)$ are positive constants that satisfy $c_1 > 1 > c_4 > c_3$ and $c_2 \in (c_0, 1)$.

Next we describe the traditional line search and trust region algorithm.

Algorithm 1

Step 1 given $x_1 \in R^n$ and $\Delta_1 > 0$, choose constants c_1, c_2, c_3 and c_4 that satisfy $0 < c_3 < c_4 < 1 < c_1$, $0 < c_2 < 1$; set $k := 1$.

Step 2 solve (2) inaccurately so that $\|d_k\| \le \Delta_k$, and so that the model has sufficient reduction.

Step 3 compute $f(x_k + d_k)$. If $f(x_k + d_k) < f(x_k)$, go to Step 4; else find the minimum positive integer i_k such that $f\left(x_k + d_k^{(i_k)}\right) < f(x_k)$; compute

$$x_{k+1} = x_k + d_k^{(i_k)}, \quad \Delta_{k+1} \in \left\{\|x_{k+1} - x_k\|, c_4\Delta_k\right\};$$

go to Step 5.

Step 4 set

$$x_{k+1} = x_k + d_k,$$
$$\gamma_k = \frac{f(x_k) - f(x_k + d_k)}{\varphi_k(0) - \varphi_k(d_k)}.$$

If $\gamma_k \ge c_2$ and $\|d_k\| < \Delta_k$, set $\Delta_{k+1} = \Delta_k$, else define

$$\Delta_{k+1} \in \begin{cases} \left[c_3\|d_k\|, c_4\Delta_k\right] & if\ \gamma_k < c_2, \\ \left[\Delta_k, c_1\Delta_k\right] & if\ \gamma_k \ge c_2\ and\ \|d_k\| = \Delta_k. \end{cases}$$

Step 5 compute g_{k+1} and B_{k+1}; set $k := k+1$; go to Step 2.

2.2. New Algorithm

In this subsection, we describe the algorithm which combines the new trust region and line search. Throughout this paper, we use $\|\cdot\|$ to represent the Euclid norm and denote $g(x_k)$ by g_k, $B(x_k)$ by B_k, etc. Vectors are all column vectors unless a transpose is used.

In the traditional trust region, the trust region subproblem is (2). In Zhou et al [6], the trust region subproblem is as follows.

$$\begin{aligned} &\min_{d \in R^n} g_k^T d + \frac{1}{2} d^T B_k d \equiv \phi(d) \\ &s.t. \quad \left\| d + \frac{\Delta_k}{\|g_k\|} g_k \right\| \le \Delta_k, \end{aligned} \tag{4}$$

where d is the trial step, $x_k - \Delta_k g_k / \|g_k\|$ is the center, Δ_k is the radius of the trust region.

After studying [6], we have test some other radii, such as $0.5\Delta_k$, $0.75\Delta_k$ and $1.5\Delta_k$. By doing this, we want to examine the effect by choosing different trust region radius at the negative gradient direction. In the numerical results, we find that the numerical result with parameter $1.5\Delta_k$ is better than that with $0.5\Delta_k$ and $0.75\Delta_k$. So we select $1.5\Delta_k$ as the new trust region radius.

Then we give the trust region subproblem as follows.

$$\begin{aligned} &\min_{d \in R^n} g_k^T d + \frac{1}{2} d^T B_k d \equiv \phi_k(d) \\ &s.t. \quad \left\| d + \frac{1.5\Delta_k}{\|g_k\|} g_k \right\| \le 1.5\Delta_k, \end{aligned} \tag{5}$$

where d is the trial step, $x_k - 1.5 \cdot \Delta_k g_k / \|g_k\|$ is the center, $1.5\Delta_k$ is the trust region radius. If we let

$$\tilde{d} = \frac{2}{3} d + \frac{\Delta_k}{\|g_k\|} g_k,$$

then (5) converts to

$$\begin{aligned} &\min_{\tilde{d} \in R^n} h_k^T \tilde{d} + \frac{1}{2} \tilde{d}^T B_k \tilde{d} \equiv \phi'\left(\tilde{d}\right) \\ &s.t. \quad \left\|\tilde{d}\right\| \le \Delta_k, \end{aligned} \tag{6}$$

where $h_k = \frac{2}{3} g_k - \frac{\Delta_k B_k g_k}{\|g_k\|}$.

At iterations of traditional line search and trust region

algorithm, a trial step d_k is generated by solving the trust region subproblem (2). While in each iteration of our line search and trust region algorithm, the trial step d_k is generated by solving the trust region subproblem (5) or (6). As in [10], we solve (6) inaccurately subject to $\left\|\tilde{d}_k\right\| \le \Delta_k$, then $d_k = \frac{3}{2}\left(\tilde{d} - \frac{\Delta_k g_k}{\|g_k\|}\right)$. Let d_k satisfies

$$\phi_k(0) - \phi_k(d_k) \ge \tau \|g_k\| \min\left\{\Delta_k, \|g_k\|/\|B_k\|\right\}, \quad (7)$$

and

$$d_k^T g_k \le -\tau \|g_k\| \min\left\{\Delta_k, \|g_k\|/\|B_k\|\right\}, \quad (8)$$

where $\tau \in (0,1)$ is a constant.

We name our line search and trust region algorithm as "L-NTR1" for convenience.

For completeness, we describe our improved line search and trust region algorithm. It is analogous with Algorithm 1.

Algorithm 2

Step 1 given $x_1 \in R^n$ and $\Delta_1 > 0$, choose constants c_1, c_2, c_3 and c_4 that satisfy $0 < c_3 < c_4 < 1 < c_1$, $0 < c_2 < 1$; set $k := 1$.

Step 2 solve (6) inaccurately so that $\left\|\tilde{d}_k\right\| \le \Delta_k$, let $d_k = \frac{3}{2}\left(\tilde{d} - \frac{\Delta_k g_k}{\|g_k\|}\right)$. So that (7) and (8) are satisfied.

Step 3 compute $f(x_k + d_k)$. If $f(x_k + d_k) < f(x_k)$, go to Step 4; else find the minimum positive integer i_k such that $f\left(x_k + d_k^{(i_k)}\right) < f(x_k)$; compute

$$x_{k+1} = x_k + d_k^{(i_k)}, \quad \Delta_{k+1} \in \left\{\|x_{k+1} - x_k\|, c_4 \Delta_k\right\};$$

go to Step 5.

Step 4 compute $x_{k+1} = x_k + d_k$, and

$$\gamma_k = \frac{f(x_k) - f(x_k + d_k)}{\phi_k(0) - \phi_k(d_k)}.$$

If $\gamma_k \ge c_2$ and $\|d_k\| < \Delta_k$, set $\Delta_{k+1} = \Delta_k$, else define

$$\Delta_{k+1} \in \begin{cases} \left[c_3\|d_k\|, c_4\Delta_k\right] & \text{if } \gamma_k < c_2, \\ \left[\Delta_k, c_1\Delta_k\right] & \text{if } \gamma_k \ge c_2 \text{ and } \|d_k\| = \Delta_k. \end{cases}$$

Step 5 compute g_{k+1} and B_{k+1}; set $k := k+1$; go to Step 2.

3. Numerical Results

In this section, we implement our new line search and trust region algorithm and compare it both with the traditional line search and trust region algorithm (L-TTR) and the improved one (L-NTR) which is proposed in our earlier work [6].

The test problems are those given by Moré, Garbow and Hillstrom [8], and we use the same numbering system as that in [8]. In all the tests, the trial step is calculated approximately by the algorithm given by Nocedal & Yuan in [9]. The algorithm is terminated when the norm of the gradient $\|g_k\|$ is less than $\varepsilon = 10^{-8}$, or when the number of calculations exceeds 30000. Next we give the results of compared different trust region radii in **Table 1**.

Where "P" represents the problem, "n" represents the dimension of the problem. "nf" and "ng" represent the numbers of function calculations and gradient calculations, respectively.

From the **Table 1**, we see that the algorithm with trust region radius of $1.5\Delta_k$ performs better than the other two settings. For most of problems, it needs less computation numbers both on function value and on gradient.

Then, we compare our line search and trust region algorithm (L-NTR1) both with L-TTR and L-NTR. The detailed results are listed in Table 3. Furthermore, for depicting the performance of different algorithms, we say that an algorithm wins only if the number of function calculations required to solve a test problem is smaller than or equal to 95% of the one required by another algorithm. For example, the number of L-TTR for calculating the objective function is 26, and the one of L-NTR1 is 27, but we say the two algorithms are balance for the problem 15. The comparison of the experimental results is given in **Table 2**.

Table 1. The computational results with different trust region radii.

		$0.5\Delta_k$	$0.75\Delta_k$	$1.5\Delta_k$
P	n	nf/ng	nf/ng	nf/ng
1	3	34(33)	44(30)	46(31)
2	6	31(26)	35(30)	47(37)
3	3	29(24)	25(20)	35(26)
4	2	11(11)	12(12)	15(15)
5	3	43(39)	45(34)	35(29)
6	4	23(21)	19(16)	19(16)
8	2	14(13)	14(14)	12(11)
9	3	33(28)	25(18)	21(15)
10	2	12(11)	13(12)	11(9)
11	4	43(38)	34(32)	27(25)
12	3	31(29)	27(23)	23(21)
13	3	24(20)	21(19)	20(15)
14	2	17(17)	14(14)	13(13)
15	8	44(26)	27(23)	27(21)
16	2	12(11)	13(12)	11(9)
17	4	41(37)	34(31)	35(27)
18	2	15(14)	15(14)	15(14)

Table 2. The statistic results of experiments.

Algorithms	Wins	Balances
L-NTR1	10	1
L-NTR	6	1
L-NTR1	10	2
L-TTR	5	2

Table 3. Comparison with different algorithms.

		L-TTR	L-NTR	L-NTR1 ($1.5\Delta_k$)
P	n	nf/ng	nf/ng	nf/ng
1	3	27(25)	32(27)	46(31)
2	6	21(19)	33(24)	47(37)
3	3	24(20)	25(18)	35(26)
4	2	11(11)	13(13)	15(15)
5	3	153(109)	45(41)	35(29)
6	4	22(18)	25(17)	19(16)
8	2	15(13)	13(12)	12(11)
9	3	21(19)	26(20)	21(15)
10	2	14(13)	13(12)	11(9)
11	4	31(26)	25(23)	27(25)
12	3	32(26)	26(23)	23(21)
13	3	28(24)	24(22)	20(15)
14	2	15(15)	14(14)	13(13)
15	8	26(22)	32(20)	27(21)
16	2	14(13)	13(12)	11(9)
17	4	43(37)	33(32)	35(27)
18	2	14(13)	15(14)	15(14)

Clearly, our proposed new algorithm L-NTR1 is better than L-TTR and L-NTR. The numbers of wins for the two algorithms L-NTR1 and L-NTR are 10 and 6, respectively. The numbers of wins for the two algorithms L-NTR1 and L-TTR are 10 and 5, respectively. That is to say, in the 17 problems, our algorithm L-NTR1 wins 10 times. So our new line search and trust region method (L-NTR1) performs much better than L-TTR and L-NTR. (**Table 3**)

4. Conclusions and Future Works

In this paper, we investigate the performance of a new algorithm where the trust region subproblem is constructed by introducing negative gradient. By introducing the improvement of the trust region subproblem, the line search and trust region algorithm is improved. The numerical test results indicate that the number of function calculations is reduced significantly for most test problems. In general, we think the new line search and trust region algorithm may be more effective in practice. In the near future, we will still devote ourselves to studying the effect with different choices of the trust region radius to investigate when the trust region algorithm performs better.

5. Acknowledgements

We want to thank Professor Y. X. Yuan for providing the original FORTRAN code of the line search and trust region algorithm. Furthermore, this work is supported by the National Nature Science Foundation of China (Grant No. 60903088, 11101115) ， the Natural Science Foundation of Hebei Province (Grant No. A2010000188) and 100-Talent Programme of Hebei Province (CPRC002).

REFERENCES

[1] J. J. Moré and D. C. Sorensen, “Computing a Trust Region Step,” *SIAM Journal on Scientific and Statistical Computing*, Vol. 4, No. 3, 1983, pp. 553-572. doi:10.1137/0904038

[2] M. J. D. Powell, “Convergence Properties of a Class of Minimization Algorithms, ” O. L. Mangasarian, R. R. Meyer and S. M. Robinson *eds.*, *Nonlinear Programming*, Academic Press, New York, 1975, pp. 1-27.

[3] R. Fletcher, “Practical Methods of Optimization, John Wiley and Sons,” New York, 1987.

[4] A. R. Conn, N. I. M. Gould and Ph. L. Toint, “Trust-Region Methods,” *SIAM*, Philadelphia, 2000. doi:10.1137/1.9780898719857

[5] G. X. Ma, “A Modified Trust Region Methods for Unconstrained Optimization (in Chinese),” Master's Thesis, 2003.

[6] Q. H. Zhou, Y. R. Zhang, F. X. Xu and Y. Geng, “An Improved Trust Region Method for Unconstrained Optimization,” *accepted by Science China Mathematics*.

[7] J. Nocedal and Y. Yuan, “Combining Trust Region and Line Search, Y. Yuan, *ed.*,” *Advances in Nonlinear Programming*, Kluwer, 1998, pp. 153-175.

[8] J. J. Moré, B. S. Garbow and K. H. Hillstrom, “Testing Unconstrained Optimization Software,” *ACM Transaction Mathematics*, Vol. 7, No. 1, 1981, pp. 17-41.

