Yokohama Publishers
ISSN 1349-8169 ONLINE JOURNAL

Pacific Journal of Optimization
Vol. 12, No. 4, pp. 741–756, October 2016

A MODIFIED FILTER RETROSPECTIVE TRUST REGION METHOD FOR UNCONSTRAINED OPTIMIZATION*

Jun Chen, Wenyu Sun†, Raimundo J.B. de Sampaio and Jinyun Yuan

Abstract: In this paper, a new filter trust region algorithm is proposed for solving unconstrained nonlinear optimization problems. It modifies the retrospective ratio to a convex combined ratio for updating the trust-region radius. Moreover we use the filter technique to relax the acceptance of the trial point. The new algorithm is shown to be globally convergent to a first-order critical point. Numerical experiments on CUTEr problems indicate that the new algorithm is competitive.

Key words: *unconstrained optimization, filter method, trust region method, global convergence, numerical experiments.*

Mathematics Subject Classification: *65K05, 90C30, 90C26.*

1 Introduction

Consider the general unconstrained optimization problem

$$\min_{x\in\mathbb{R}^n} f(x), \tag{1.1}$$

where the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable and bounded below.

In unconstrained optimization, the trust-region method is a well-known strategy and has been proved successful in practice. Now the trust-region methods have been widely applied in science, engineering, management, and finance (see [22, 30]). Given a starting point x_0, the trust-region algorithm generates a sequence of iterates $\{x_k\}$ which we hope converges to a first order critical point of problem (1.1). At each iterate x_k, the trust-region method constructs a model function $m_k(x)$, usually a quadratic model function, to approximate the objective function within a so called "trust region", which is usually represented by a ball in Euclidean norm of radius Δ_k centered at x_k. Then we seek a trial step s_k by (approximately)

*This work was supported by the National Natural Science Foundation of China (grant No. 11571178, 11401308, 11431102), A Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD), the Research Program of Natural Science of Universities in Jiangsu Province (grant No. 12KJD110005), the Innovative Program for Graduates of Jiangsu Universities (grant No. CXZZ11_0871), and CAPES and CNPq of Brazil.

†Corresponding author.

minimizing the model $m_k(x)$ inside the trust region. To decide whether or not to accept the trial point $x_k + s_k$ and determine the trust-region radius Δ_{k+1}, we compute the ratio

$$\rho_k \triangleq \frac{f(x_k) - f(x_k + s_k)}{m_k(x_k) - m_k(x_k + s_k)} \tag{1.2}$$

which measures the agreement between the model function and the objective function. If the ratio ρ_k is larger than a small positive constant, say $\eta \in (0,\ 0.25]$, we accept the trial point $x_k + s_k$ as the next iterate x_{k+1} and choose the trust region radius $\Delta_{k+1} \geqslant \Delta_k$ (in this case the k-th iteration is called successful). Otherwise, we reject the trial point $x_k + s_k$ and re-solve the subproblem within the smaller trust region by decreasing Δ_k to get the new trial step. Due to the strong global convergence properties and high numerical efficiency, various researchers contributed to the development of trust-region methods [5,7,18,21,23–27,32,33]. There are several optimization monographs which provide a good summary of trust-region methods, such as Conn, Gould, Toint [5], Nocedal and Wright [22], and Sun and Yuan [30].

Recently, a variant of the trust-region methods, which is called retrospective trust-region method was proposed by Bastin et al. [1]. In this method, the most relevant information on the model's quality at the current iterate is used, instead of the information at the previous iterate. Note that, in the classical trust-region methods, the radius Δ_k at iterate x_k is updated by ρ_{k-1} at the end of the $(k-1)$-th iteration which uses the model function $m_{k-1}(x)$. But in retrospective trust-region algorithm the trust region radius Δ_k is determined at the beginning of the k-th iteration by the *retrospective ratio*

$$\widetilde{\rho}_k \triangleq \frac{f(x_k) - f(x_{k-1})}{m_k(x_k) - m_k(x_{k-1})} \tag{1.3}$$

instead of ρ_{k-1} in (1.2). Convergence analysis shows that this new trust-region method shares all the convergence properties of the classical trust-region method under the similar assumptions. The preliminary numerical experiments indicate that the retrospective trust-region method is competitive to the classical trust-region method.

The filter method was first proposed by Fletcher and Leyffer [11] in the context of constrained optimization. It attracted much attention in optimization society and was developed further due to its ability of avoiding the pitfalls of penalty function methods, see [10,12,19,20,31]. More recently, filter techniques were extended to solve unconstrained optimization problems, see [3,9,13,16,20,35]. The extensive numerical experiments show the encouraging performance of the filter-type algorithms. Filter methods, to some extent, have the properties that allow the increasing of the objective function values in iterative procedure, which is similar to that of the non-monotone optimization methods (see [6,17,28,29,34]). Many researches indicate that relaxing the monotonicity can improve the efficiency of algorithms, especially when the objective function features deep narrow curved valleys.

In this paper, we give a modified version to the retrospective trust-region algorithm and combine it with a relaxed filter technique. Our algorithm is proved robust and effective in theory and practice. The paper is organized as follows. In Section 2, we describe the modified filter retrospective trust-region algorithm (MFRTR). The convergence properties are analyzed in Section 3. Finally, in Section 4, we report the numerical results obtained by the experiments on a set of standard test problems from the CUTEr library [2,15].

In the end of this section, we introduce some notation used in the paper. The notation $\|\cdot\|$ denotes the Euclidean norm on $\mathbb{R}^n$. For a set S, $|S|$ denotes its cardinal number. For convenience, we denote the gradient of the objective function by

$$\nabla f(x) \triangleq g(x) = (g_1(x), g_2(x), \ldots, g_n(x))^T.$$

2 The New Algorithm

2.1 The weighted retrospective trust-region update

At each iterate x_k, we construct the quadratic model function $m_k(x)$ as follows:

$$m_k(x_k + s) = f(x_k) + g_k^T s + \frac{1}{2} s^T B_k s, \tag{2.1}$$

where $g_k = g(x_k)$ and B_k is the exact Hessian $\nabla^2 f(x_k)$ or its approximation. Then we obtain the trial step s_k by (approximately) solving the following trust-region subproblem:

$$\min\{m_k(x_k + s) : \ \|s\| \leqslant \Delta_k\}, \tag{2.2}$$

where Δ_k is the trust region radius at iteration k.

One of the key ingredients in a trust-region algorithm is the strategy for choosing the trust-region radius Δ_k at k-th iteration. In the classical trust-region algorithm, we compute the ratio ρ_{k-1} which is defined by (1.2) at the previous iteration $k-1$, and then use it to determine the trust-region radius Δ_k. Recently, Bastin et al. [1] proposed a retrospective trust-region method that determines Δ_k according to the retrospective ratio $\widetilde{\rho}_k$ defined by (1.3), which measures the agreement of the current model function $m_k(x)$ and the objective function at x_k. Therefore this technique synchronizes the radius update with the change in models. In their opinion, the most relevant information on the model's quality at the current iterate would be more useful than that at the previous iterate. Extensive numerical results [1] show the efficiency of this new trust-region algorithm. However, due to the success of the classical trust-region methods to some extend, we think that it is not wise to discard completely the ratio ρ_{k-1} for determining Δ_k. So, based on using the retrospective ratio $\widetilde{\rho}_k$, we also take into account ρ_{k-1} defined by (1.2). Therefore, we introduce the convex combination of the ratio $\widetilde{\rho}_k$ and ρ_{k-1} which uses the information of both the previous and the current model functions to improve the efficiency of the trust-region method. More specially, we define

$$\widehat{\rho}_k = \lambda \widetilde{\rho}_k + (1 - \lambda)\rho_{k-1}, \tag{2.3}$$

where $\lambda \in [0, 1]$, and then update the trust-region radius Δ_k according to $\widehat{\rho}_k$ instead of the single retrospective ratio $\widetilde{\rho}_k$ in [1].

2.2 The multidimensional filter

In the convergence analysis of the algorithm to the first-order critical point, we can regard unconstrained optimization (1.1) as two aims: the first one is to decrease the objective function f, and the second one is to drive the gradient ∇f to zero. So, to increase the chance of accepting the new trial point, we consider using a **filter** mechanism related to the gradient of f which uses the concept of domination from multi-objective optimization according to the above two aims.

Definition 2.1 ([11]). A point x_1 is said to dominate another point x_2 if and only if

$$|g_j(x_1)| \leqslant |g_j(x_2)| \quad \text{for all } j = 1, 2, \ldots, n. \tag{2.4}$$

In this case $g(x_1)$ is said to dominate $g(x_2)$.

In the context of unconstrained optimization, this means that x_1 is at least as good as x_2 with respect to encourage convergence to the first-order critical point. Based on this definition, we can define a structure called a *filter*, which will be used in our trust-region type algorithm as a criterion for testing acceptance of a trial step.

Note that we do not wish to accept a new point x_k^+ if its gradient $g(x_k)$ is arbitrarily close to being dominated by another point already in the filter. The following strategy we proposed is a modification of (2.5) in [16] and (2.9) in [4].

Definition 2.2. A point x is acceptable for the filter $\mathcal{F}$ if and only if for all $g(x_l) \in \mathcal{F}$,

$$|g_j(x)| \leqslant |g_j(x_l)| - \gamma_g \Phi(\mathcal{F}) \quad \textit{for at least one } j \in \{1, 2, \ldots, n\}, \tag{2.5}$$

where we set $\Phi(\mathcal{F}) \triangleq \min\{\|g_k\| : g_k \in \mathcal{F}\}$ and $\gamma_g \in (0,\, 1/\sqrt{n})$.

We introduce $\Phi(\mathcal{F})$ to relax the acceptable condition (2.5) in [16] and allow more iterates to be potentially acceptable.

When x_k^+ is acceptable for filter $\mathcal{F}$, we add $g(x_k^+)$ to the filter and remove every $g(x_l) \in \mathcal{F}$ such that

$$|g_j(x_l)| > |g_j(x_k)| \quad \text{for all } j \in \{1, 2, \ldots, n\}.$$

2.3 New algorithm

Based on the framework of the classical trust-region algorithm, we apply the proposed weighted ratio $\widehat{\rho}_k$ in (2.3) to update the trust-region radius and use the modified filter mechanism (2.5) to test the acceptance of the trial step. Further, we use the relaxed filter technique and the parameter `NONCONVEX` switching strategy to efficiently solve the trust-region model subproblem. The new algorithm is described as follows.

Algorithm 2.3. MFRTR

Step 0: ***Initialization.***

Given the initial point $x_0 \in \mathbb{R}^n$, and an initial trust region radius $\Delta_0 > 0$. The constants $\gamma_g \in (0,\, 1/\sqrt{n})$, λ, η, η_1, η_2, γ_1, γ_2, and γ_3 are also given and satisfy $0 \leqslant \lambda \leqslant 1,\ 0 < \eta < 1,\ 0 < \eta_1 < \eta_2 < 1$, and

$$0 < \gamma_1 \leqslant \gamma_2 < 1 \leqslant \gamma_3.$$

Compute $f(x_0)$, set $k = 0$. Initialize the filter $\mathcal{F}$ to the empty set and choose $f_{sup} = f(x_0)$. Define a logical flag `NONCONVEX`.

Step 1: ***Model definition.***

Compute $g_k = g(x_k)$. *If* $g_k \leq \epsilon$, set the first-order critical point $x^* = x_k$ and **stop**; *else* compute B_k and define the model function $m_k(x_k + s)$ by (2.1).

Step 2: ***Weighted retrospective trust-region radius update.***

If $k = 0$, go to Step 3. *If* $x_k = x_{k-1}$, then choose $\Delta_k \in [\gamma_1 \Delta_{k-1},\, \gamma_2 \Delta_{k-1}]$ *(unsuccessful iteration)*; *else* define

$$\widetilde{\rho}_k = \frac{f(x_{k-1}) - f(x_k)}{m_k(x_{k-1}) - m_k(x_k)}, \tag{2.6}$$

$$\widehat{\rho}_k = \lambda \widetilde{\rho}_k + (1 - \lambda)\rho_{k-1} \tag{2.7}$$

where ρ_{k-1} was calculated in STEP 4 of the preceding iteration, and set

$$\Delta_k \in \begin{cases} [\gamma_1\Delta_{k-1},\ \gamma_2\Delta_{k-1}], & \textit{if} \quad \widehat{\rho}_k < \eta_1, \\ [\gamma_2\Delta_{k-1},\ \Delta_{k-1}], & \textit{if} \quad \widehat{\rho}_k \in [\eta_1,\ \eta_2), \\ [\Delta_{k-1},\ \gamma_3\Delta_{k-1}], & \textit{if} \quad \widehat{\rho}_k \geqslant \eta_2. \end{cases} \tag{2.8}$$

STEP 3: ***Step calculation.***
Determine an approximate solution s_k to the trust-region subproblem (2.2). *If* the model m_k is nonconvex, set `NONCONVEX = TRUE`; *else* set it to be `FALSE`. Compute the trial point $x_k^+ = x_k + s_k$.

STEP 4: Compute the function value $f(x_k^+)$ and define the ratio:

$$\rho_k = \frac{f(x_k) - f(x_k^+)}{m_k(x_k) - m_k(x_k^+)}. \tag{2.9}$$

If $f(x_k^+) > f_{sup}$, set $x_{k+1} = x_k$ and go to STEP 6.

STEP 5: ***Acceptance of the trial point by the filter $\mathcal{F}$.***

If `NONCONVEX = TRUE` *:*

If $\rho_k \geqslant \eta$ then
set $x_{k+1} = x_k^+$, $f_{sup} = f(x_{k+1})$ and reinitialize the filter $\mathcal{F}$ to the empty set;
else set $x_{k+1} = x_k$.

If `NONCONVEX = FALSE` *:*
Compute $g_k^+ = g(x_k^+)$.

- ◇ If x_k^+ is acceptable for the filter $\mathcal{F}$ according to (2.5):
  Set $x_{k+1} = x_k^+$ and add g_k^+ to the filter $\mathcal{F}$ if $\rho_k < \eta$;
- ◇ If x_k^+ is **not** acceptable for the filter $\mathcal{F}$ according to (2.5):
  If $\rho_k \geqslant \eta$, then set $x_{k+1} = x_k^+$; *else* set $x_{k+1} = x_k$.

STEP 6: Set $k := k + 1$, and go to STEP 1.

Remark 2.4. In the algorithm, if $x_{k+1} = x_k^+$, the k-th iteration is called successful.

Remark 2.5. The parameter `NONCONVEX` is set TRUE if the negative curvature of the model is met when solving the trust-region subproblem. Here a generalized Lanczos trust-region algorithm (see [14]) is employed to solve the trust-region subproblem and, at the same time, determine the parameter `NONCONVEX` during the procedure.

Remark 2.6. According to our numerical experiments and Gould et al. [16], employing the parameter `NONCONVEX` switching rule can increase the efficiency of the algorithm to some extent. When the parameter `NONCONVEX` is set to be TRUE, that means the model m_k is nonconvex. In this case we accept the trial step and reset the filter to the empty set if there is the sufficient decrease of the objective function. Otherwise, when `NONCONVEX` is FALSE which means that the model m_k is convex, we employ the filter mechanism to test the acceptance of the trial step.

3 Convergence Analysis

In this section, for the convenience of analysis of the convergence properties of Algorithm 2.3, we let the parameter ϵ in our algorithm to be zero. Then we will prove that Algorithm 2.3 either terminates in finitely many iterations, or generates an infinite sequence which is globally convergent to the first-order critical points. The main convergence analysis is an extension of the results in [5, 16, 30].

Now we give some standard assumptions in the following.

A1 The objective function $f(x) : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable.

A2 The generated sequence $\{x_k\}$ remains in a bounded and closed domain of $\mathbb{R}^n$.

A3 For all k, matrix sequence $\{B_k\}$ is uniformly bounded, i.e., there is a positive scalar $\kappa_{umh} > 1$ such that

$$\|B_k\| \leqslant \kappa_{umh} - 1, \ k = 1, 2, \cdots .$$

Obviously, the assumptions A1 and A2 together imply that there exist constants l, u, $l \leqslant u$ and $\kappa_{ufh} \geqslant 1$ such that

$$f(x_k) \in [l, u], \qquad \left\|\nabla^2 f(x_k)\right\| \leqslant \kappa_{ufh}. \tag{3.1}$$

For the purpose of our analysis, we define

$$\mathcal{S} = \{k \mid x_{k+1} = x_k + s_k\},$$

the set of *successful iterations*;

$$\mathcal{A} = \{k \mid g(x_k^+) \text{ is added to } \mathcal{F}\},$$

the set of *adding filter iterations*;

$$\mathcal{D} = \{k \mid \rho_k \geqslant \eta_1\},$$

the set of *sufficient descent iterations*; and

$$\mathcal{N} = \{k \mid \texttt{NONCONVEX = TRUE}\},$$

the set of *nonconvex iterations*. It is easy to show that $\mathcal{A} \subseteq \mathcal{S}$ and

$$\mathcal{S} \cap \mathcal{N} = \mathcal{D} \cap \mathcal{N}. \tag{3.2}$$

First, we state a crucial property of the algorithm.

Lemma 3.1. *According to the procedure of Algorithm 2.3, we have that, for all* $k \geqslant 0$,

$$f(x_0) - f(x_{k+1}) \geqslant \sum_{\substack{j=0 \\ j \in \mathcal{S} \cap \mathcal{N}}}^{k} [f(x_j) - f(x_{j+1})]. \tag{3.3}$$

Proof. We denote $\mathcal{S} \cap \mathcal{N} \cap \{1, 2, \ldots, k\} = \{k_i : i = 1, 2, \ldots, t\}$. At iteration k_i, we can get $f_{sup} = f(x_{k_i+1})$. Then it ensures that

$$f(x_{k_{i+1}}) \leqslant f(x_{k_i+1}) < f(x_{k_i}), \qquad f(x_{k_t+1}) \geqslant f(x_{k+1}).$$

These inequalities give that

$$\begin{aligned} \sum_{\substack{j=0 \\ j\in\mathcal{S}\cap\mathcal{N}}}^{k} [f(x_j) - f(x_{j+1})] &= \sum_{i=1}^{t} [f(x_{k_i}) - f(x_{k_i+1})] \\ &\leqslant \sum_{i=1}^{t} [f(x_{k_i}) - f(x_{k_{i+1}})] \\ &\leqslant f(x_0) - f(x_{k+1}), \end{aligned}$$

which completes the proof. □

In trust-region methods, the solution of the subproblem (2.2) provides, at iteration k, a *sufficient decrease* on the model, which plays an important role in the convergence analysis.

Lemma 3.2. *Let s_k be the solution of the trust-region subproblem (2.2), then there is a scalar $\kappa_{mdc} \in (0, 1)$, such that*

$$m_k(x_k) - m_k(x_k + s_k) \geqslant \kappa_{mdc} \|g_k\| \min\{\|g_k\|/\beta_k, \Delta_k\}, \tag{3.4}$$

where $\beta_k = 1 + \|B_k\|$.

Proof. See the proof of Theorem 6.3.1 in [5] and Lemma 6.1.5 in [30]. □

Similar to Theorem 6.4.1 in [5] and Lemma 6.1.6 in [30], from above lemmas, we can obtain

$$|f(x_k + s_k) - m_k(x_k + s_k)| \leqslant \kappa_{ubh} \Delta_k^2 \tag{3.5}$$

and

$$|f(x_{k-1}) - m_k(x_{k-1})| \leqslant \kappa_{ubh} \Delta_{k-1}^2, \tag{3.6}$$

where

$$\kappa_{ubh} \triangleq \max\{\kappa_{ufh}, \kappa_{umh}\}. \tag{3.7}$$

Lemma 3.3. *Let assumptions A1–A3 hold. Moreover, suppose that $g_{k-1} \neq 0$. If the trust region radius Δ_{k-1} satisfies*

$$\Delta_{k-1} \leqslant \min\{1 - \eta, 1 - \eta_2, \frac{1 - \eta_2}{3 - 2\eta_2}\} \frac{\kappa_{mdc}}{\kappa_{ubh}} \|g_{k-1}\|, \tag{3.8}$$

then $\rho_{k-1} \geqslant \eta$ and

$$\Delta_k \geqslant \Delta_{k-1}. \tag{3.9}$$

Proof. Observe, from Algorithm 2.3, that η and η_2 lie in the interval $(0, 1)$, which implies that

$$1 - \eta < 1, \ 1 - \eta_2 < 1, \ \frac{1 - \eta_2}{3 - 2\eta_2} < 1. \tag{3.10}$$

The conditions (3.8), (3.10), (3.7), and $\kappa_{mdc} \in (0, 1)$ imply that

$$\Delta_{k-1} \leqslant \frac{\|g_{k-1}\|}{\kappa_{ubh}} \leqslant \frac{\|g_{k-1}\|}{\kappa_{umh}}. \tag{3.11}$$

As a consequence, we apply (3.4), (3.6), (3.11) and (3.8) to deduce that

$$\begin{aligned} |\rho_{k-1}-1| &= \left|\frac{f(x_{k-1})-f(x_k)}{m_{k-1}(x_{k-1})-m_{k-1}(x_k)}-1\right| \\ &= \left|\frac{m_{k-1}(x_k)-f(x_k)}{m_{k-1}(x_{k-1})-m_{k-1}(x_k)}\right| \\ &\leqslant \frac{\kappa_{ubh}\Delta_{k-1}^2}{\kappa_{mdc}\,\|g_{k-1}\|\min\{\frac{\|g_{k-1}\|}{\kappa_{umh}},\ \Delta_{k-1}\}} \\ &= \frac{\kappa_{ubh}\Delta_{k-1}}{\kappa_{mdc}\,\|g_{k-1}\|} \\ &\leqslant 1-\eta. \end{aligned} \tag{3.12}$$

This inequality immediately gives $\rho_{k-1} \geqslant \eta$. So the $(k-1)$th iteration should be successful and $x_k = x_{k-1} + s_{k-1} \neq x_{k-1}$.

Now, by using the similar derivation in [1, 5, 30], we get that Δ_{k-1} satisfies (3.8) and further that $|\widetilde{\rho}_k - 1| \leqslant 1-\eta_2$. Moreover, (3.12) also gives $|\rho_{k-1}-1| \leqslant 1-\eta_2$. Thus we obtain

$$|\widehat{\rho}_k - 1| = |\lambda\widetilde{\rho}_k + (1-\lambda)\rho_{k-1} - 1| \leqslant \lambda\,|\widetilde{\rho}_k - 1| + (1-\lambda)\,|\rho_{k-1}-1| \leqslant 1-\eta_2.$$

Therefore $\widehat{\rho}_k \geqslant \eta_2$, and (2.8) ensures that (3.9) holds. □

Now we prove the finite termination of the sequence of the iterates when there are only finitely many successful iterations.

Theorem 3.4. *Let the assumptions A1–A3 and the sufficient decrease condition (3.4) hold. Suppose that there are only finitely many successful iterations, i.e., $|\mathcal{S}| < +\infty$. Then the algorithm terminates in Step 1 with, for some iteration k, $x_k = x^*$ and $\nabla f(x^*) = 0$.*

Proof. Obviously, when $|\mathcal{S}| < +\infty$, there are two cases according to the mechanism of the algorithm:

(i) For some k, $\nabla f(x_k) = 0$. So the algorithm terminates in finitely many iterations with the first-order critical point $x^* = x_k$;

(ii) There exist infinitely many unsuccessful iterations.

We only need to prove that the case (ii) does not happen. For obtaining a contradiction, we assume that k_0 is the index of the last successful iterate and the algorithm proceeds with

$$x_{k_0+1} = x_{k_0+j} \quad \text{and} \quad \|\nabla f(x_{k_0+j})\| = \varepsilon > 0, \ \text{for all } j \geqslant 1. \tag{3.13}$$

This implies that

$$\lim_{j\to\infty} \Delta_{k+j} = 0 \tag{3.14}$$

since the trust region radius decreases at unsuccessful iteration. However, because $\|\nabla f(x_{k_0+j})\| = \varepsilon > 0$, by using Lemma 3.4 in [16], we have that (3.14) is impossible and we deduce a contradiction to (3.13). Thus we confirm that the case (ii) does not happen and the algorithm must terminate at some iterate $x_k = x^*$. □

Now we are in a position to establish the main convergence property of the algorithm when there are infinitely many successful iterations, i.e., $|\mathcal{S}| = +\infty$. In this situation, we first consider that there are infinitely many gradients added to the filter $\mathcal{F}$ in the course of the algorithm, and then consider there are finitely many gradients added to the filter.

Theorem 3.5. *Suppose that A1–A3 and (3.4) hold, and that $|\mathcal{A}| = |\mathcal{S}| = +\infty$. Then we have*

$$\liminf_{k\to\infty} \|g_k\| = 0. \tag{3.15}$$

Proof. By contradiction. We assume that

$$\|g_k\| \geqslant \kappa_{lbg} \tag{3.16}$$

for all k large enough and for some $\kappa_{lbg} > 0$, and we define $\mathcal{A} = \{k_i\}$.

The assumption (3.16) and Theorem 3.5 in [16] imply that $|\mathcal{S} \cap \mathcal{N}|$ is finite and therefore that the filter is no longer reset to the empty set for k sufficiently large. Moreover, since the assumptions A1 and A2 imply that the sequence of gradients $\{g(x_k)\}$ remains in a bounded and closed domain of $\mathbb{R}^n$, there must exist a subsequence $\{k_l\} \subset \{k_i + 1\}$ such that

$$\lim_{l\to\infty} g(x_{k_l}) = g_\infty \qquad \text{with } \|g_\infty\| \geqslant \kappa_{lbg}. \tag{3.17}$$

By the definition of $\{k_l\}$, x_{k_l} is acceptable for the filter $\mathcal{F}$ in each iteration $k_l - 1$. This implies that for each l sufficiently large, there exists an index $j_l \in \{1, 2, \ldots, n\}$ such that

$$|g_{j_l}(x_{k_l})| - \left|g_{j_l}(x_{k_{l-1}})\right| < -\gamma_g \Phi(\mathcal{F}) \tag{3.18}$$

since the filter is not reset for l large enough. From (3.16) and the definition of $\Phi(\mathcal{F})$, we can say that $\Phi(\mathcal{F}) \geqslant \kappa_{lbg}$. Hence we deduce from (3.18) that

$$|g_{j_l}(x_{k_l})| - \left|g_{j_l}(x_{k_{l-1}})\right| < -\gamma_g \kappa_{lbg}$$

for all l sufficiently large. But the left-hand side of this inequality tends to zero when l tends to infinity because of (3.17), which yields a contradiction. □

Theorem 3.6. *Suppose that the assumptions A1–A3 and (3.4) hold and that $|\mathcal{S}| = +\infty$ but $|\mathcal{A}| < +\infty$. Then (3.15) holds.*

Proof. The proof is similar to that of Theorem 3.8 in [16]. □

4 Numerical Experiments

In this section, we consider to evaluate the performance of Algorithm 2.3 implemented in Fortran 95. The numerical results are obtained by running our algorithm on the set of 156 unconstrained optimization problems from the CUTEr library [2, 15]. The names of the problems with their dimensions are detailed in Table 1. For the problems whose dimension may be adjusted, we choose a reasonably small value in order not to overload our computing environment. In each case, the starting points are the standard ones provided by the CUTEr library. All tests are performed in double precision on Dell computer (Intel Core2 Duo, 2.93GHz, 2G RAM) under Fedora 8 Linux and the Intel Fortran compiler (version 10.1.018) with default options.

We would like to compare our algorithm MFRTR with some variants of trust-region algorithms described in the following:

Table 1: The 156 test problems and their dimensions.

Problem	n	Problem	n	Problem	n
AKIVA	2	EIGENBLS	110	OSBORNEA	5
ALLINITU	4	EIGENCLS	462	OSBORNEB	11
ARGLINA	200	ENGVAL1	100	OSCIPATH	15
ARGLINB	200	ENGVAL2	3	PALMER1C	8
ARGLINC	200	ERRINROS	50	PALMER1D	7
ARWHEAD	100	EXPFIT	2	PALMER2C	8
BARD	3	EXTROSNB	5	PALMER3C	8
BDQRTIC	100	FLETCBV2	100	PALMER4C	8
BEALE	2	FLETCBV3	100	PALMER5C	6
BIGGS6	6	FLETCHBV	100	PALMER6C	8
BOX	100	FLETCHCR	100	PALMER7C	8
BOX3	3	FMINSRF2	64	PALMER8C	8
BRKMCC	2	FMINSURF	121	PENALTY1	10
BROWNAL	200	FREUROTH	100	PENALTY2	10
BROWNBS	2	GENHUMPS	10	PENALTY3	50
BROWNDEN	4	GENROSE	100	PFIT1LS	3
BROYDN7D	100	GROWTHLS	3	PFIT2LS	3
BRYBND	100	GULF	3	PFIT3LS	3
CHAINWOO	100	HAIRY	2	PFIT4LS	3
CHNROSNB	50	HATFLDD	3	POWELLSG	4
CLIFF	2	HATFLDE	3	POWER	75
COSINE	100	HATFLDFL	3	QUARTC	25
CRAGGLVY	100	HEART6LS	6	ROSENBR	2
CUBE	2	HEART8LS	8	S308	2
CURLY10	100	HELIX	3	SBRYBND	50
CURLY20	100	HILBERTA	2	SCHMVETT	100
CURLY30	100	HILBERTB	10	SCOSINE	100
DECONVU	61	HIMMELBB	2	SCURLY10	10
DENSCHNA	2	HIMMELBF	4	SCURLY20	100
DENSCHNB	2	HIMMELBG	2	SCURLY30	100
DENSCHNC	2	HIMMELBH	2	SENSORS	100
DENSCHND	3	HUMPS	2	SINEVAL	2
DENSCHNE	3	HYDC20LS	99	SINQUAD	100
DENSCHNF	2	JENSMP	2	SISSER	2
DIXMAANA	300	KOWOSB	4	SNAIL	2
DIXMAANB	90	LIARWHD	36	SPARSINE	100
DIXMAANC	90	LOGHAIRY	2	SPARSQUR	100
DIXMAAND	90	MANCINO	100	SPMSRTLS	100
DIXMAANE	300	MARATOSB	2	SROSENBR	100
DIXMAANF	300	MEXHAT	2	TOINTGOR	50
DIXMAANG	300	MEYER3	3	TOINTGSS	100
DIXMAANH	90	MODBEALE	10	TOINTPSP	50
DIXMAANI	90	MOREBV	50	TOINTQOR	50
DIXMAANJ	300	MSQRTALS	100	TQUARTIC	100
DIXMAANK	15	MSQRTBLS	49	TRIDIA	30
DIXMAANL	90	NCB20	110	VARDIM	200
DIXON3DQ	100	NCB20B	21	VAREIGVL	50
DJTL	2	NONCVXU2	100	VIBRBEAM	8
DQDRTIC	100	NONCVXUN	100	WATSON	12
DQRTIC	50	NONDIA	30	WOODS	4
EDENSCH	36	NONDQUAR	100	YFITU	3
EIGENALS	110	NONMSQRT	49	ZANGWIL2	2

FTR: This variant is the filter-trust-region algorithm proposed in [16] except that the algorithmic flag "RESTRICT" is not used in our implementation for simplicity.

RTR: Retrospective trust-region algorithm proposed in [1].

BTR: Basic trust-region algorithm presented in Conn et al. [5], or Algorithm 6.1.1 proposed in Sun and Yuan [30].

To make the comparison as fair as possible, all these algorithms discussed above use the parameters with $\eta = 0.1,\ \eta_1 = 0.1,\ \eta_2 = 0.9,\ \gamma_1 = 0.25,\ \gamma_2 = 2,\ \Delta_0 = 1.0$, and

$$\gamma_g = \min\{0.001,\ \frac{1}{2\sqrt{n}}\}.$$

In addition, we choose

$$f_{sup} = \min\{10^6\,|f(x_0)|\,, f(x_0) + 1000\}.$$

In particular, to justify the use of the weighted retrospective ratio (2.7), we set the weight λ to be five different values, i.e.,

$$\lambda = 0,\ 0.25,\ 0.5,\ 0.75,\ \text{and } 1 \tag{4.1}$$

in coding retrospective-type trust-region algorithms above. Since the exact Hessian matrix of $f(x)$ at iterate x_k is used to construct the quadratic model (2.1), in this paper we mainly consider the medium test problems. We approximately solve the trust-region subproblem (2.2) by running the Fortran-90 module VF05 with the default options from the well-known HSL Mathematical Software Library, which is the implementation of Generalized Lanczos trust-region algorithm (GLTR) [14]. In the computational experiments, each algorithm is stopped if

$$\|\nabla f(x_k)\| \leqslant 10^{-5},$$

or if the number of iterations exceeds 20000, or if the operation time exceeds 10 minutes. In the latter two cases the algorithm is failed.

In numerical tests we choose iteration counts (**iter**) and number of gradient evaluations (**ng**) required to satisfy the convergence test as two measures to make numerical comparison. Efficiency and robustness comparisons are made by using the performance profiles proposed by Dolan and Moré [8].

Now we are in a position to report our numerical results.

We consider the impact of varying λ. We compare five different algorithmic variants of MFRTR by setting λ to be the five choices in (4.1). The performance profiles in terms of iteration counts and number of gradient evaluations are plotted in Fig. 1 and Fig. 2, respectively. It is not difficult to see in these figures that the algorithm with $\lambda = 0.5$ performs more efficient than the other cases.

Moreover, we compare the MFRTR algorithm using $\lambda = 0.5$ with other variants of trust-region algorithms BTR, RTR and FTR described above. Fig. 3 and Fig. 4 give the performance profiles for four algorithms in terms of the iteration counts and the number of gradient evaluations. These figures show that the algorithm MFRTR outperforms the other algorithms.

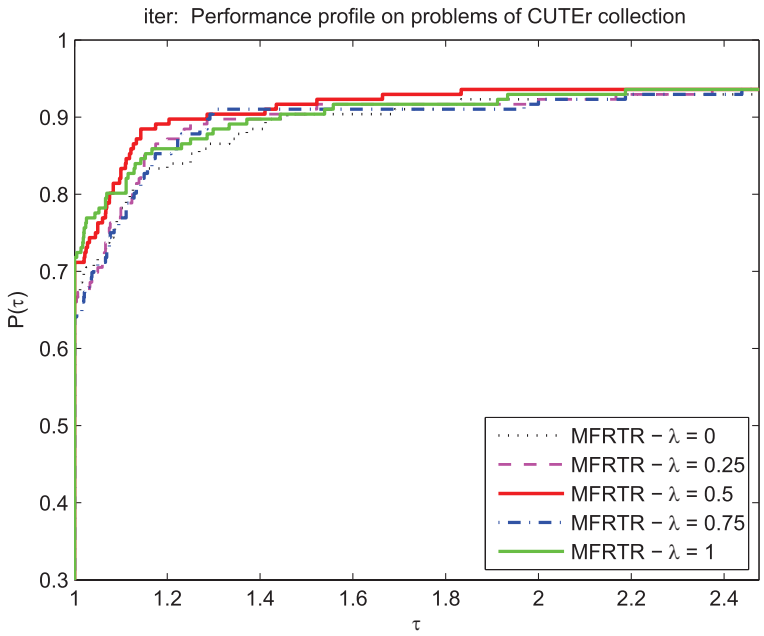

Figure 1: Performance profile on the iteration counts for MFRTR with different λ

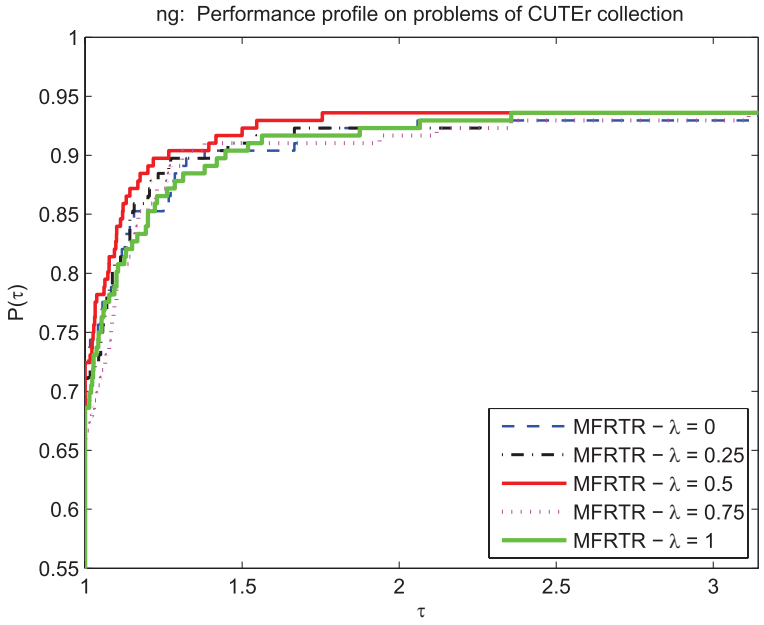

Figure 2: Performance profile on the number of gradient evaluations for MFRTR with different λ

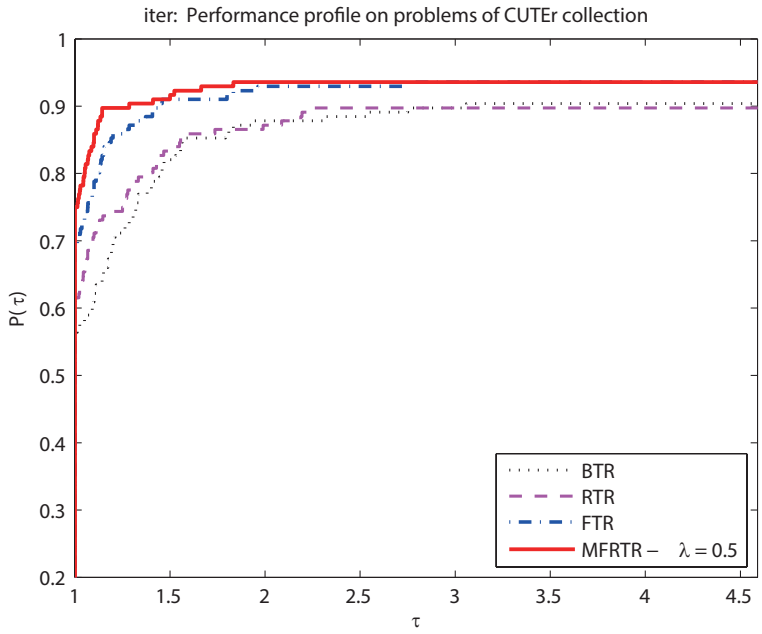

Figure 3: Performance profile in terms of the iteration counts

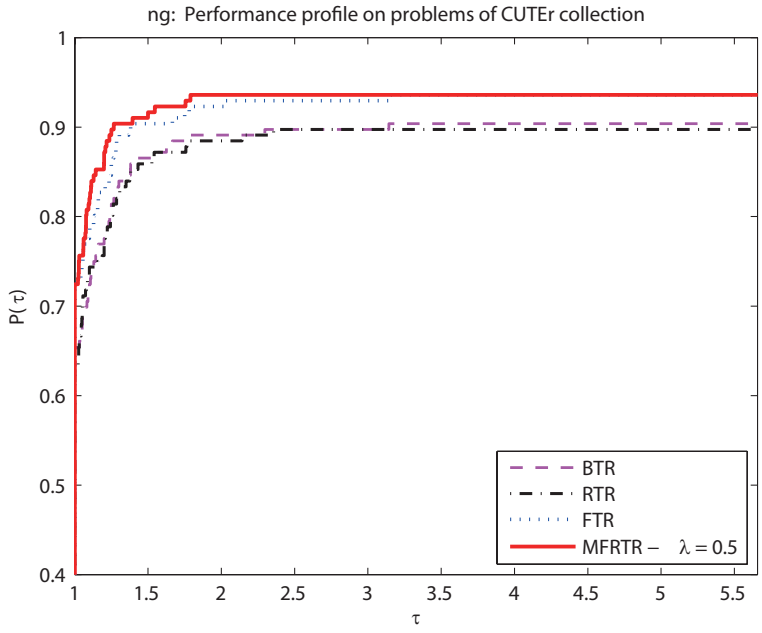

Figure 4: Performance profile in terms of the number of gradient evaluations

5 Conclusions

In this paper we propose a modified retrospective trust region method via a convex combination of the retrospective ratio and the original updating ratio and via embedding a relaxed filter strategy. Under the mild conditions, we prove the first-order convergence property of our algorithm. Numerical experiments performed in the CUTEr library indicate that the new algorithm is competitive to the traditional trust-region algorithm and some existing variants. For the future research, we will extend this idea to constrained optimization and nonsmooth optimization.

Acknowledgements

We are grateful to two anonymous referees for their constructive and helpful comments and suggestions which are employed to improve our manuscript.

References

[1] F. Bastin, V. Malmedy, M. Mouffe, Ph.L. Toint and D. Tomanos, A retrospective trust-region method for unconstrained optimization, *Math. Program.* 123 (2010) 395–418.

[2] I. Bongartz, A.R. Conn, N.I.M. Gould and Ph.L. Toint, CUTE: Constrained and Unconstrained Testing Environment, *ACM Trans. Math. Software* 21 (1995) 123–160.

[3] Y. Chen and W. Sun, A dwindling filter line search method for unconstrained optimization, *Math. Comp.* 84 (2015) 187–208.

[4] C.M. Chin and R. Fletcher, On the global convergence of an SLP-flter algorithm that takes EQP steps, *Math. Program.* 96 (2003) 161–177.

[5] A.R. Conn, N.I.M. Gould and Ph. L. Toint, *Trust-Region Methods*, SIAM, Philadelphia, 2000.

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

[7] J.E. Dennis and R.B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, SIAM, Philadelphia, 1996.

[8] E.D. Dolan and J.J. Moré, Benchmarking optimization software with performance profiles, *Math. Program.* 91 (2002) 201–213.

[9] M. Fatemi and N. Mahdavi-Amiri, A filter trust-region algorithm for unconstrained optimization with strong global convergence properties, *Comput. Optim. Appl.* 52 (2012) 239–266.

[10] R. Fletcher, N.I.M. Gould, S. Leyffer, Ph.L. Toint and A. Wächter, Global convergence of a trust-region SQP-filter algorithm for general nonlinear programming, *SIAM J. Optim.* 13 (2002) 635–659.

[11] R. Fletcher and S. Leyffer, Nonlinear programming without a penalty function, *Math. Program.* 91 (2002) 239–269.

[12] R. Fletcher, S. Leyffer and Ph.L. Toint, On the global convergence of a filter–SQP algorithm, *SIAM J. Optim.* 13 (2002) 44–59.

[13] N.I. M. Gould, S. Leyffer and Ph.L. Toint, A multidimensional filter algorithm for nonlinear equations and nonlinear least-squares, *SIAM J. Optim.* 15 (2004) 17–38.

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

[15] N.I.M. Gould, D. Orban, and Ph.L. Toint, CUTEr and SifDec: A constrained and unconstrained testing environment, revisited, *ACM T. Math. Software* 29 (2003) 373–394.

[16] N.I.M. Gould, C. Sainvitu and Ph.L. Toint, A filter-trust-region method for unconstrained optimization, *SIAM J. Optim.* 16 (2005) 341–357.

[17] L. Grippo, F. Lampariello and S. Lucidi, A nonmonotone line search technique for Newton's method, *SIAM J. Numer. Anal.* 23 (1986) 707–716.

[18] Q. Han, W. Sun, J. Han and R.J.B. Sampaio, An adaptive conic trust-region method for unconstrained optimization, *Optim. Methods Softw.* 20 (2005) 665–677.

[19] C. Li and W. Sun, On filter-successive linearization methods for nonlinear semidefinite programming, *Sci. China Math.* 52 (2009) 2341–2361.

[20] W.H. Miao and W. Sun. A filter trust-region method for unconstrained optimization, *Numer. Math. J. Chinese Univ.* 29 (2007) 88–96.

[21] J.J. Moré, Recent developments in algorithms and software for trust region methods, in: *Mathematical Programming: The State of the Art*, A. Bachem, M. Grötschel and B. Korte (eds.), Springer-Verlag, Heidelberg, 1983, pp. 258–287.

[22] J. Nocedal and S.J. Wright, *Numerical Optimization*, Second Ed., Springer-Verlag, New York, 2006.

[23] M.J.D. Powell, A new algorithm for unconstrained optimization, in: *Nonlinear Programming*, J. B. Rosen, O. L. Mangasarian and K. Ritter (Eds.), Academic Press, London, 1970, pp. 31–65.

[24] M.J.D. Powell, On the global convergence of trust region algorithms for unconstrained minimization, *Math. Program.* 29 (1984) 297–303.

[25] M.J.D. Powell and Y. Yuan, A trust region algorithm for equality constrained optimization, *Math. Program.* 49 (1991) 189–211.

[26] L. Qi, Trust region algorithms for solving nonsmooth equations, *SIAM J. Optim.* 5 (1995) 219–230.

[27] G.A. Shultz, R.B. Schnabel and R.H. Byrd, A family of trust-region-based algorithms for unconstrained minimization with strong global convergence properties, *SIAM J. Numer. Anal.* 22 (1985) 47–67.

[28] W. Sun, Nonmonotone trust region method for solving optimization problems, *Appl. Math. Comput.* 156 (2004) 159–174.

[29] W. Sun, J.Y. Han and J. Sun, Global convergence of nonmonotone descent methods for unconstrained optimization problems, *J. Comput. Appl. Math.* 146 (2002) 89–98.

[30] W. Sun and Y. Yuan, *Optimization Theory and Methods: Nonlinear Programming*, Springer-Verlag, New York, 2006.

[31] A. Wächter and L.T. Biegler, Line search filter methods for nonlinear programming: Motivation and global convergence, *SIAM J. Optim.* 16 (2005) 1-31.

[32] Z.H. Wang and Y. Yuan, A subspace implementation of quasi-newton trust region methods for unconstrained optimization, *Numer. Math.* 104 (2006) 241–269.

[33] Y. Yuan, On subproblem of trust region algorithms for constrained optimization, *Math. Program.* 47 (1990) 53–63.

[34] H.C. Zhang and W.W. Hager, A nonmonotone line search technique and its application to unconstrained optimization, *SIAM J. Optim.* 14 (2004) 1043–1056.

[35] Y. Zhang, W. Sun and L. Qi, A nonmonotone filter Barzilai-Borwein method for optimization, *Asia-Pac. J. Oper. Res.* 27 (2010) 55–69.

Manuscript received 20 May 2015
revised 7 October 2015
accepted for publication 10 October 2015

Jun Chen
School of Mathematical Sciences, Jiangsu Key Laboratory for NSLSCS
Nanjing Normal University, Nanjing 210023, China;

School of Mathematics and Information Technology
Nanjing Xiaozhuang University, Nanjing 211171, China
Email address: junchennj@163.com

Wenyu Sun
School of Mathematical Sciences, Jiangsu Key Laboratory for NSLSCS
Nanjing Normal University, Nanjing 210023, China
Email address: wysun@njnu.edu.cn

Raimundo J.B. de Sampaio
PPGEPS, Pontifical Catholic University of Parana (PUCPR)
80215-901, Curitiba, Parana, Brazil
E-mail address: raimundo.sampaio@pucpr.br

Jinyun Yuan
Department of Mathematics
Universidade Federal do Parana (UFPR)
81531-980, Curitiba, Parana, Brazil
Email address: jin@ufpr.br