Journal of Computational Mathematics
Vol.28, No.1, 2010, 72–86.

http://www.global-sci.org/jcm
doi:10.4208/jcm.2009.09-m2924

A NEW TRUST-REGION ALGORITHM FOR NONLINEAR CONSTRAINED OPTIMIZATION*

Lingfeng Niu and Yaxiang Yuan
LSEC, ICMSEC, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China
Email: niulf@lsec.cc.ac.cn; Email: yyx@lsec.cc.ac.cn

Abstract

We propose a new trust region algorithm for nonlinear constrained optimization problems. In each iteration of our algorithm, the trial step is computed by minimizing a quadratic approximation to the augmented Lagrange function in the trust region. The augmented Lagrange function is also used as a merit function to decide whether the trial step should be accepted. Our method extends the traditional trust region approach by combining a filter technique into the rules for accepting trial steps so that a trial step could still be accepted even when it is rejected by the traditional rule based on merit function reduction. An estimate of the Lagrange multiplier is updated at each iteration, and the penalty parameter is updated to force sufficient reduction in the norm of the constraint violations. Active set technique is used to handle the inequality constraints. Numerical results for a set of constrained problems from the CUTEr collection are also reported.

Mathematics subject classification: 90C30, 65K05.
Key words: Trust region method, Augmented Lagrange function, Filter method, active set.

1. Introduction

This paper presents a new trust region algorithm for general constrained optimization problems having the form:

$$\begin{aligned} \min \quad & f(x) && && (1.1a)\\ \text{subject to} \quad & c_i(x) = 0, && i \in \mathcal{E} = \{1, ..., m_e\} && (1.1b)\\ & c_i(x) \geq 0, && i \in \mathcal{I} = \{m_e + 1, ..., m\} && (1.1c) \end{aligned}$$

where $f : \Re^n \to \Re$ and $\mathbf{c} : \mathbf{\Re^n} \to \mathbf{\Re^m}$ are twice continuously differentiable functions.

The trust-region methods are a class of numerical methods for optimization. While line search type methods search the next iteration in a line, trust region methods try to find the next iteration point within a region. Such a region is called the trust region and it is normally a set (say, a ball or box) centered at the current iterate. The essential parts of a trust region method are finding the trial step in the trust region and deciding whether the trial step should be accepted [28].

The trial step of a trust region method is normally computed by solving a trust region subproblem. There are mainly three different approaches. The null space type method decomposes the trial step into a range space step and a null space step with the range space step

* Received October 6, 2008 / Revised version received April 22, 2009 / Accepted June 8, 2009 / Published online October 26, 2009 /

reducing the constraint violations and the null space step decreasing the Lagrange function in the null space (for example, see [1, 18, 26]). The second type of trust region subproblems is the the two-ball subproblem, which minimizes a quadratic approximation to the objective function subject to a reduction of the norm of the linearized constraints (see, e.g., [2, 21]). The third kind of trust region subproblems can be derived by exact penalty functions, which minimize an approximation to some penalty function. Such subproblems include the SL_1QP subproblem [10] and L_∞ subproblem [27]. The subproblem we use in our new method belongs to the third type. Our subproblem is to minimize an approximate augmented Lagrangian function. Augmented Lagrangian function [19] is an exact penalty function if the Lagrange multiplier is exact and the penalty parameter is sufficiently large.

Recently, merit-function-free algorithms have been attracted much attention from researchers, for example see [12, 25]. The basic idea of such algorithms is to regard the constrained optimization problem as a two objective problem. One is to decrease the original objective function while the other is to decrease the constraint violations. Algorithms based on merit functions normally require a monotone decrease in a merit function, while merit-function-free algorithms use non-monotone decrease conditions on the original objective function and the constraint violations. In our algorithm, we also use the filter idea and a trial step will not be thrown away unless it not only does not reduce the merit function, but also not accepted by a filter.

This paper is organized as follows. The next section introduces the motivation and the basic idea of our new algorithm based on the equality constrained problem. In section 3 we extend our algorithm to general constrained problems by employing the active set technique. Section 4 presents some numerical results for problems from the CUTEr collection [14] and gives a brief conclusion.

Throughout the paper $\|\cdot\|$ denotes the Euclidean norm. We denote the gradient of f by $g = g(x) = \nabla f(x)$, the Jacobian of the constraints by $A = A(x) = (\nabla c_1(x), \nabla c_2(x), \cdots, \nabla c_m(x))$. Superscripts $^{(k)}$ refer to iteration indices and $f^{(k)}$ is taken to mean $f(x^{(k)})$ etc.Subscripts $_k$ refer to elements of vector. Quantities related to a local solution are superscripted by $*$.

2. An Algorithm for Equality Constrained Optimization

In this section, we give a new algorithm for the equality constrained optimization

$$\min \quad f(x) \tag{2.1a}$$

$$\text{subject to} \quad c_i(x) = 0, \qquad i \in \mathcal{E}. \tag{2.1b}$$

Before giving the detailed descriptions of our algorithm, we need to address some issues such as the calculations of the trial step, techniques for updating the Lagrange multiplier and the penalty parameter, and the criteria for accepting the trial step.

2.1. Computing the trust region step

The trust region step is computed by minimizing a quadratic function which approximates the augmented Lagrangian function. The augmented Lagrangian function has the form

$$\Phi(x, \lambda, \sigma) = f(x) - \lambda^T c(x) + \sigma \|c(x)\|^2, \tag{2.2}$$

where $\lambda \in \Re^{|\mathcal{E}|}$ is the Lagrange multiplier and $\sigma \geq 0$ is the penalty parameter.

At the beginning of the $k-$th iteration, we have the current iterate $x^{(k)}$, the current approximate Lagrange multiplier $\lambda^{(k)}$, and the penalty parameter $\sigma^{(k)}$. Thus, the augmented Lagrangian function at the $k-$th iteration is

$$\Phi(x, \lambda^{(k)}, \sigma^{(k)}) = f(x) - (\lambda^{(k)})^T c(x) + \sigma^{(k)} \|c(x)\|^2. \tag{2.3}$$

Define the quadratic function

$$Q^{(k)}(d) = (g^{(k)} - A^{(k)}\lambda^{(k)})^T d + \frac{1}{2} d^T B^{(k)} d + \sigma^{(k)} \|c^{(k)} + (A^{(k)})^T d\|^2, \tag{2.4}$$

where $A^{(k)}$ is the Jacobian of the constraints computed at $x^{(k)}$, and $B^{(k)}$ is a symmetric matrix approximating either the Hessian of the Lagrangian function

$$W^{(k)} = \nabla^2 f(x^{(k)}) - \sum_{i \in \mathcal{E}} \lambda_i^{(k)} \nabla^2 c_i(x^{(k)}) \tag{2.5}$$

or

$$\overline{W}^{(k)} = \nabla^2 f(x^{(k)}) - \sum_{i \in \mathcal{E}} \lambda_i^{(k)} \nabla^2 c_i(x^{(k)}) + 2\sigma^{(k)} \sum_{i \in \mathcal{E}} c_i(x^{(k)}) \nabla^2 c_i(x^{(k)}). \tag{2.6}$$

If $B^{(k)} = \overline{W}^{(k)}$, $Q^{(k)}$ is the second order Taylor expansion of the augmented Lagrangian function $\Phi(x, \lambda^{(k)}, \sigma^{(k)})$. If $B^{(k)} = W^{(k)}$, we use the second order Taylor expansion of the Lagrangian function

$$L(x, \lambda) = f(x) - \lambda^T c(x)$$

and the linear approximation of the constraints. Thus, function (2.4) is a quadratic approximation to the augmented Lagrangian function, and our trust region subproblem is

$$\begin{aligned} \min \quad & Q^{(k)}(d) && \text{(2.7a)} \\ \text{subject to} \quad & ||d|| \le \Delta^{(k)}, && \text{(2.7b)} \end{aligned}$$

where $\Delta^{(k)} > 0$ is the trust region radius at the current iteration. An advantage of our approach is that subproblem (2.7), just as the standard subproblem of trust region algorithms for unconstrained optimization, is to minimize a quadratic function within a ball. This simple subproblem has been studied extensively and there are many mature methods for solving it [8, 13, 15, 16, 20, 22–24, 29]. Another advantage of our approach is that by using formulation (2.7) we do not need to worry about the possible inconsistency between the linearized constraints and the trust region constraint, nor about linearly dependency of the gradients of the constraints.

2.2. Updating the Lagrange Multiplier

The Lagrange multiplier $\lambda^{(k)}$ is updated from iteration to iteration, and we hope that it will converge to the true Lagrange multiplier λ^* at the solution x^*.

From the definition of the trial step $d^{(k)}$, there exists a non-negative parameter $\zeta^{(k)}$ such that

$$\begin{aligned} g^{(k)} - A^{(k)}(\lambda^{(k)} - 2\sigma^{(k)}({A^{(k)}}^T d^{(k)} + c^{(k)})) + B^{(k)} d^{(k)} + \zeta^{(k)} d^{(k)} = 0\,, && \text{(2.8a)} \\ \zeta^{(k)}(||d^{(k)}|| - \Delta^{(k)}) = 0\,. && \text{(2.8b)} \end{aligned}$$

A good estimate of the Lagrange multiplier is the least squares solution of

$$g(x) - A(x)\lambda = 0, \tag{2.9}$$

namely $\lambda = (A(x))^{+}g(x)$ (see [11]). Thus, it would be desirable to set the new estimate of the Lagrange multiplier λ^{new} to

$$\lambda^{(k)} - 2\sigma^{(k)}({A^{(k)}}^T d^{(k)} + c^{(k)}) - (A^{(k)})^{+}(B^{(k)} + \zeta^{(k)} I)d^{(k)}. \tag{2.10}$$

Ignoring the last term, because it is difficult to obtain, we derive our Lagrange multiplier update formula

$$\lambda^{trial} = \lambda^{(k)} - 2\sigma^{(k)}({A^{(k)}}^T d^{(k)} + c^{(k)}). \tag{2.11}$$

The above update formula for the Lagrange multiplier is new, though it can be viewed as a kind of approximation to the well known first-order Lagrange multiplier formula:

$$\lambda^{+} = \lambda^{(k)} - 2\sigma^{(k)} c(x_{k+1}) \tag{2.12}$$

(for example, see (14.2.5) in [7] and (12.2.15) in [11]) if $x_{k+1} = x_k + d_k$.

2.3. Updating the penalty parameter

Now we consider how to update the penalty parameter. Let x^* be a solution of (2.1) at which the LICQ is satisfied (that is, the gradients $\nabla c_i(x^*), i \in \mathcal{E}$, are linearly independent vectors), and the second sufficient conditions are satisfied for $\lambda = \lambda^*$. Then there is a threshold value $\overline{\sigma} \geq 0$ such that for all $\sigma \geq \overline{\sigma}$, x^* is a local minimizer of the augmented Lagrangian function $\Phi(x, \lambda^*, \sigma)$ which is defined by (2.2). However, it is not easy to estimate an accurate upper bound for such a threshold value $\overline{\sigma}$. Consider the points which are not too far from the feasible set (say, $\|c(x)\| \leq 1$), we have that

$$\begin{aligned}
\Phi(x, \lambda^*, \sigma) &= f(x) - (\lambda^*)^T c(x) + \sigma ||c(x)||^2 \\
&\leq f(x) + (\sigma ||c(x)|| - (\lambda^*)^T c(x)) \\
&= f(x) + \frac{1}{2}\sigma ||c(x)|| + \left(\frac{1}{2}\sigma ||c(x)|| - (\lambda^*)^T c(x)\right).
\end{aligned} \tag{2.13}$$

If $\sigma \geq 2\|\lambda^*\|$, the first two terms in the last line of (2.13) is an exact penalty function and the last term in the last line of (2.13) is non-negative. Thus, it is natural for us to impose the condition

$$\sigma^{(k+1)} \geq 2\|\lambda^{(k+1)}\|. \tag{2.14}$$

The main aim of the penalty term is to force the iterates converging to the feasible set. Hence we increase the penalty parameter by at least twice if the trial step fails to reduce the norm of the constraint violations by half, namely

$$\sigma^{(k+1)} \geq 2\sigma^{(k)} \tag{2.15}$$

if $||c(x^{(k)} + d^{(k)})|| \geq 0.5||c^{(k)}||$.

To summarize, our update formula for the new penalty parameter is

$$\sigma^{(k+1)} = \begin{cases} \max\left(2\sigma^{(k)}, 2||\lambda^{(k+1)}||\right), & \text{if } ||c(x^{(k)} + d^{(k)})|| \geq 0.5||c^{(k)}||; \\ \max\left(\sigma^{(k)}, 2||\lambda^{(k+1)}||\right), & \text{otherwise.} \end{cases} \tag{2.16}$$

The above update for the parameter is different from any of the known techniques, though our approach used for deriving the formula is similar to that in [21].

2.4. Criteria for accepting the trial point

First, the augmented Lagrangian function is used as a merit function to decide whether the trial step $d^{(k)}$ should be accepted. The ratio of the actual reduction to the predicted reduction in the merit function

$$\rho^{(k)} = \frac{\Phi(x^{(k)}, \lambda^{(k)}, \sigma^{(k)}) - \Phi(x^{(k)} + d^{(k)}, \lambda^{trial}, \sigma^{(k)})}{Q^{(k)}(0) - Q^{(k)}(d^{(k)})} \tag{2.17}$$

is computed. If this ratio is positive, the trial step is accepted (namely $x^{(k+1)} = x^{(k)} + d^{(k)}$). Otherwise, we do not simply throw away the trial step. Instead, we give it another chance by testing whether it can be accepted by the filter.

A pair $(h^{(k)}, f^{(k)})$ is said to dominate another pair $(h^{(l)}, f^{(l)})$ if and only if both $f^{(k)} \leq f^{(l)}$ and $h^{(k)} \leq h^{(l)}$. A filter $\mathcal{F}$ is a list of pairs (f, h) such that no pair dominates any other [**?**]. For equality constrained optimization, $h^{(k)} = \|c^{(k)}\|$. In order to make the filter an efficient tool for forcing convergence in real computation, we require either the objective function value or the constraint violation to be reduced sufficiently in a successful step. Hence, only when

$$h^{new} < (1 - \gamma_\vartheta) h^{(k)} \quad \text{or} \quad f^{new} < f^{(k)} - \gamma_\vartheta h^{new} \quad \text{for all } (h^{(k)}, f^{(k)}) \in \mathcal{F}, \tag{2.18}$$

we accept the new point $x^{new} = x^{(k)} + d^{(k)}$ and add it into the filter. The parameter $\gamma_\vartheta \in (0, 1)$ is a very small constant, which sets a small "margin" around the border of the dominated part of the (h, f)-space in which we shall reject trial points [7].

There are two advantages to combine the filter and traditional actual vs predicted reduction ratio test together. Firstly, the utilization of the filter increases the opportunity of trial points to be accepted. Intuitively, it is likely that the new algorithm will accelerate the convergence of the traditional trust region method. Secondly, since the filter is only used as the supplement of the merit function, it contains only a few elements. Thus, there is no need to consider removing excessive elements from the filter. Consequently, the maintenance of the filter in the new algorithm is simpler than that in the method which uses the filter as the only criterion for trial point acceptance [12].

2.5. An algorithm for equality constrained optimization

Base on the above preparation we can give our new trust region algorithm for equality constrained problems:

Algorithm A

Step 0 **Initialization.** Let $x^{(0)} \in \Re^n$, $\Delta^{(0)} > 0$, $\lambda^{(0)} \in \Re^m$, $\sigma^{(0)} > 0$, and a symmetric matrix $B^{(0)}$ be given, as well as constants $0 < \eta_1 \leq \eta_2 < 1, \gamma_\vartheta \in (0, 1), \epsilon_s, \epsilon_c$. Compute $f^{(0)}$ and $c^{(0)}$. Set the initial filter as $\mathcal{F} = \{(c^{(0)}, f^{(0)}), (10c^{(0)}, -\infty)\}$ and $k = 0$.

Step 1 **Step calculation.** Solve the trust region subproblem (2.7) obtaining $d^{(k)}$.

Step 2 **Termination test.** If $||d^{(k)}|| > \epsilon_s$, go to Step 3;
If $||c^{(k)}|| < \epsilon_c$, stop and return $x^{(k)}$ as a solution;
Set $\sigma^{(k)} := 10\sigma^{(k)}$, go to Step 1.

Step 3 **Acceptance of the trial point.**

3.1 `Trial point information calculation.` Let $x^{trial} = x^{(k)} + d^{(k)}$. Evaluate $f(x^{trial})$, $c(x^{trial})$ and the corresponding Lagrange multiplier denoted by λ^{trial} according to formula (2.11).

3.2 `Acceptance determined by Penalty Function.` Compute $\rho^{(k)}$ by (2.17); If $\rho^{(k)} \geq 0$, set $x^{(k+1)} = x^{(k)} + d^{(k)}$ and go to Step 4.

3.3 `Acceptance determined by Filter.` If x^{trial} is accepted by the filter, update filter, set $x^{(k+1)} = x^{(k)} + d^{(k)}$ and go to Step 4; else set $x^{(k+1)} = x^{(k)}$, go to Step 4.4.

Step 4 **Parameters update.**

4.1 `Lagrange multiplier update.` Set $\lambda^{(k+1)} = \lambda^{trial}$;

4.2 `Penalty parameter update.` Update $\sigma^{(k+1)}$ by (2.16);

4.3 `Generate the next approximate Hessian.` Update $B^{(k+1)}$;

4.4 `Trust-region radius update.` Set

$$\Delta^{(k+1)} = \begin{cases} \max\left(2\Delta^{(k)}, 2||d^{(k)}||\right), & \text{if } \rho^{(k)} \geq \eta_2, \\ \Delta^{(k)}, & \text{if } \rho^{(k)} \in [\eta_1, \eta_2], \\ \min\left(0.5\Delta^{(k)}, 0.5||d^{(k)}||\right), & \text{if } \rho^{(k)} < \eta_1, \end{cases} \tag{2.19}$$

4.5 `Incremental iteration index.` Set $k := k + 1$ and go to Step 1.

In the above algorithm, how to update $B^{(k+1)}$ is not given. Possible choices are $B^{(k+1)} = W^{(k+1)}$ or $B^{(k+1)} = \overline{W}^{(k+1)}$. We can also use quasi-Newton update formulae [11] to generate $B^{(k+1)}$.

3. Extend the Algorithm to Inequality Constraints Case

Now we consider extending the algorithm to problems with inequality constraints. We introduce slack variables $s_i, i \in \mathcal{I}$ and transform the inequality constraints to equality constraints and bound constraints:

$$\begin{aligned} \min \quad & f(x) && && (3.1a) \\ \text{subject to} \quad & c_i(x) = 0, && i \in \mathcal{E}; && (3.1b) \\ & c_i - s_i = 0, && i \in \mathcal{I}; && (3.1c) \\ & s_i \geq 0, && i \in \mathcal{I}. && (3.1d) \end{aligned}$$

One way for handling the non-negative constraints on the slack variables is the popular interior point approach, which replaces the inequality constraints by the log-penalty function. For example, see [3–5]. Here we use a different approach. Keeping the inequality constraint

(3.1d) and replacing the equality constraint problem (3.1a)-(3.1c) by the minimization of the corresponding augmented Lagrange function, we obtain the following problem:

$$\min_{x,s} \quad f(x) - \sum_{i\in\mathcal{E}} \lambda_i c_i(x) - \sum_{i\in\mathcal{I}} \lambda_i (c_i(x) - s_i) + \sigma \sum_{i\in\mathcal{E}} (c_i(x))^2 + \sigma \sum_{i\in\mathcal{I}} (c_i(x) - s_i)^2 \tag{3.2a}$$

$$\text{s.t.} \quad s_i \geq 0, \qquad \text{for all } i \in \mathcal{I}. \tag{3.2b}$$

For a given x, the objective function in (3.2a) is a convex quadratic function with respect to each of the slack variables s_i. Thus we can easily find the optimal value for each s_i:

$$s_i = \max\left(c_i(x) - \lambda_i/(2\sigma), 0\right), \qquad \text{for all } i \in \mathcal{I}. \tag{3.3}$$

Substituting the above relation into (3.2a), we derive an unconstrained minimization problem on variables x:

$$\min_x \Phi_{\mathcal{A}(x,\lambda,\sigma)}(x, \lambda, \sigma) \tag{3.4}$$

where $\mathcal{A}(x, \lambda, \sigma) = \mathcal{E} \cup \{i \in \mathcal{I} \mid c_i(x) < \lambda_i/2\sigma\}$ is the active set, and

$$\Phi_{\mathcal{A}}(x, \lambda, \sigma) = f(x) - \sum_{i\in\mathcal{A}} \lambda_i c_i(x) + \sigma \sum_{i\in\mathcal{A}} (c_i(x))^2 \tag{3.5}$$

is the augmented Lagrangian function based on the active constraints. If $x^{(k)}$, $\lambda^{(k)}$ and $\sigma^{(k)}$ are known, we define the working set as:

$$\mathcal{A}^{(k)} = \mathcal{A}(x^{(k)}, \lambda^{(k)}, \sigma^{(k)}) = \mathcal{E} \cup \{i \in \mathcal{I} \mid c_i^{(k)} < \lambda_i^{(k)}/2\sigma^{(k)}\}. \tag{3.6}$$

Similar to the previous section, we can compute the trial step by minimizing a quadratic approximation to (3.5). The subproblem at the $k-$th iteration can be written as

$$\min_d \quad Q_{\mathcal{A}}^{(k)}(d) \tag{3.7a}$$

$$\text{subject to} \quad \|d\| \leq \Delta^{(k)}, \tag{3.7b}$$

where

$$Q_{\mathcal{A}}^{(k)}(d) = (g^{(k)} - A_{\mathcal{A}}^{(k)} \lambda_{\mathcal{A}}^{(k)})^T d + \frac{1}{2} d^T B_{\mathcal{A}}^{(k)} d + \sigma^{(k)} \|c_{\mathcal{A}}^{(k)} + (A_{\mathcal{A}}^{(k)})^T d\|^2, \tag{3.8}$$

with

$$c_{\mathcal{A}}^{(k)} = (\ c_i(x^{(k)}) \mid \quad i \in \mathcal{A}^{(k)})^T, \tag{3.9}$$

$$\lambda_{\mathcal{A}}^{(k)} = (\ \lambda_i^{(k)} \quad \mid \quad i \in \mathcal{A}^{(k)})^T, \tag{3.10}$$

$$A_{\mathcal{A}}^{(k)} = (\ \nabla c_i(x^{(k)}) \mid \quad i \in \mathcal{A}^{(k)})\ , \tag{3.11}$$

and $B_{\mathcal{A}}^{(k)}$ is an approximation to the Hessian of the Lagrangian function based on $\mathcal{A}^{(k)}$. Let $d^{(k)}$ be the solution of subproblem (3.7), and we use $d^{(k)}$ as our trial step.

As for the update of Lagrange multipliers, it is very similar to the equality constrained case, except that we require all the multipliers corresponding to the inequality constraints to be non-negative. For constraints not in the active set, we simply set those multipliers to be zero. Thus, we can obtain the following formula for updating the Lagrange multipliers:

$$\lambda_i{}^{trial} = \begin{cases} \lambda_i{}^{(k)} - 2\sigma^{(k)}(\nabla c_i{}^{(k)})^T d^{(k)} + c_i{}^{(k)}), & i \in \mathcal{E}, \\ max\{\lambda_i{}^{(k)} - 2\sigma^{(k)}(\nabla c_i{}^{(k)})^T d^{(k)} + c_i{}^{(k)}), 0\}, & i \in \mathcal{A}^{(k)} \setminus \mathcal{E}, \\ 0, & \text{otherwise.} \end{cases} \tag{3.12}$$

The filter for general constrained problems is also similar to that for equality constrained problems, except that the definition of $h(x)$ is changed to

$$h(x) = \sqrt{\sum_{i\in\mathcal{E}}(c_i(x))^2 + \sum_{i\in\mathcal{I}}\left[\min(c_i(x),0)\right]^2}\ . \tag{3.13}$$

Therefore, we can give our algorithm for general nonlinear optimization problems as follows.

Algorithm B

Step 0 **Initialization.** Let $x^{(0)} \in \Re^n$, $\Delta^{(0)} > 0$, $\lambda^{(0)} \in \Re^m$, $\sigma^{(0)} > 0$ be given, as well as constants $0 < \eta_1 \le \eta_2 < 1, \gamma_\vartheta \in (0,1), \epsilon_s, \epsilon_c$. Compute $f(x^{(0)})$, $c(x^{(0)})$, initial working set $\mathcal{A}^{(0)}$ and $B^{(0)}$. Set $\mathcal{F} = \{(c_0, f_0), (10c_0, -\infty)\}$ and $k = 0$.

Step 1 **Step calculation.** Solve (3.7) obtaining a trial step $d^{(k)}$.

Step 2 **Termination test.** If $||d^{(k)}|| > \epsilon_s$, go to Step 3;
If $h^{(k)} < \epsilon_c$, stop and return $x^{(k)}$ as a solution;
Set $\sigma^{(k)} := 10\sigma^{(k)}$, go to Step 1.

Step 3 **Acceptance of the trial point.**

3.1 `Trial point information calculation.` Let $x^{trial} = x^{(k)} + d^{(k)}$. Evaluate $f(x^{trial})$, $c(x^{trial})$, $\mathcal{A}^{trial}$ and the corresponding Lagrange multiplier denoted by λ^{trial} according to formula (3.12).

3.2 `Acceptance determined by Penalty Function.` Compute the ratio of the actual reduction to the predicted reduction, i.e.

$$\rho^{(k)} = \frac{\Phi_{\mathcal{A}^{(k)}}(x^{(k)}, \lambda^{(k)}, \sigma^{(k)}) - \Phi_{\mathcal{A}^{trial}}(x^{trial}, \lambda^{trial}, \sigma^{(k)})}{Q^{(k)}_{\mathcal{A}^{(k)}}(0) - \overline{Q}^{(k)}_{\mathcal{A}^{(k)}}(d^{(k)})} \tag{3.14}$$

If $\rho^{(k)} > 0$, set $x^{(k+1)} = x^{trial}$ and go to Step 4.

3.3 `Acceptance determined by Filter.`
If x^{trial} is accepted by the filter, set $x^{(k+1)} = x^{(k)} + d^{(k)}$, update the filter and go to Step 4; else $x^{(k+1)} = x^{(k)}$, go to Step 4.5.

Step 4 **Parameters update.**

4.1 `Lagrange multiplier update.` Set $\lambda^{(k+1)} = \lambda^{trial}$;

4.2 `working set update.` Set $\mathcal{A}^{(k+1)} = \mathcal{A}^{trial}$.

4.3 `Penalty parameter update.` Update $\sigma^{(k+1)}$ by

$$\sigma^{(k+1)} = \begin{cases} \max\left(2\sigma^{(k)}, 2||\lambda^{(k+1)}||\right), & \text{if } h(x^{(k)} + d^{(k)}) \ge 0.5h(x^{(k)}); \\ \max\left(\sigma^{(k)}, 2||\lambda^{(k+1)}||\right), & \text{otherwise.} \end{cases} \tag{3.15}$$

4.4 `Generate the next approximate Hessian.` Update $B^{(k+1)}$.

4.5 `Trust-region radius update.` Set Δ_{k+1} by (2.19).

4.6 `Incremental iteration index` Set $k := k + 1$ and go to Step 1.

Obviously, Algorithm B reduces to Algorithm A when there is no inequality constraints.

4. Numerical Results and Discussion

We implemented our new algorithm in MATLAB and run the experiments with MATLAB 7.0 in Fedora Core Linux environment. The computer used is a DELL Inspiron 2200 laptop with 256MB RAM memory and Celeron processor.

The test problems are from CUTEr [14] collection. There are 930 test problems in CUTEr, whose SIF file size is less then 100 Kbytes. Due to the memory limitation, 924 problems can be decoded by SifDec in our machine. Among the problems which can be decoded, 748 are constrained optimization, which consist of 458 problems with simple bounds and 290 problems without simple bounds. For the bound constraints, although they can be treated as the general inequalities, we prefer to handle them explicitly, which wil be one of the extensions of our algorithm in the future. Thus, for the current experiments, only the nonlinear constrained optimization problems without simple bounds in CUTEr are chosen: except linear programming, there are 171 equality constrained problems and 115 general constrained problems.

The parameters used in our implementation for the following numerical tests are:

$$\Delta_0 = 1,\ \sigma_0 = 1,\ ,\ \eta_1 = 0.1,\ \eta_2 = 0.9,$$
$$\gamma_\vartheta = 0.0001,\ \epsilon_s = 0.00001,\ \epsilon_c = 0.00001.$$

Exact Hessian matrices were used to construct the trust region subproblems in all the runs. Namely, for equality constrained problems, we set $B^{(k)} = W^{(k)}$, where $W^{(k)}$ is defined by (2.5), while for general constrained problems, we use

$$B^{(k)} = \nabla^2 f(x^{(k)}) - \sum_{i \in \mathcal{A}^{(k)}} \lambda_i^{(k)} \nabla^2 c_i(x^{(k)}). \tag{4.1}$$

CUTEr problems provide initial Lagrange multiplier λ_0. However, for most CUTEr problems, this initial multiplier λ_0 is the null vector. For such problems, we begin to utilize the Lagrange multiplier only when the constraints violation is less than a tolerance which is set to 0.1. Moré's subroutine gqptar [16] is called at each iteration to solve the subproblem.

To examine the effectiveness of the new algorithm, we also run LANCELOT [5], which is a well-known package based on minimizing the augmented Lagrangian function, with the same set of test problems and compare the results. To have a fair comparison, we let LANCELOT also use the exact Hessian matrices to form the 2-norm trust region subproblems and solve them accurately. Both algorithms use the same termination accuracy 0.00001, which is LANCELOT's default setting. Since our algorithm is implemented in MATLAB and LANCELOT is in Fortran, the numbers of function evaluations and gradient evaluations are recorded instead of the computing time. If the algorithm can not converge after 1,000 objective function evaluations, it is considered as "failed" on the problem. For the 286 problems we tested, there are 5 equality constrained problems and 8 general constrained problems on which both our method and LANCELOT failed. And there are 20 equality constrained problems and 12 general constrained problems on which two algorithms found local optima with different objective function values. We report the results on all the other problems which both algorithms obtained the same optimal objective function values in Tables 4.1 and 4.2. Columns "f, c" and "g, A" give the number of function and gradient evaluations respectively. Column "n" and "m" show the dimension of

the problems (n variables and m constraints). An entry of "F" indicates that the algorithm is terminated without finding a solution after 1,000 function evaluations.

Table 4.1: Results for equality constrained problems.

	Alg.A		LANCELOT		Pro. Dim.			Alg.A		LANCELOT		Pro. Dim.	
Problem	f,c	g,A	f,c	g,A	n	m	Problem	f,c	g,A	f,c	g,A	n	m
AIRCRFTA	2	2	5	5	8	5	HS26	27	27	22	20	3	1
ARGTRIG	3	3	7	6	200	200	HS27	22	20	14	14	3	1
ARTIF	13	13	230	223	102	100	HS28	4	4	7	7	3	1
AUG2DC	12	12	30	30	703	300	HS39	18	13	20	20	4	2
AUG2D	13	13	30	30	703	300	HS40	12	9	16	15	4	3
BDVALUE	8	7	2	2	102	100	HS42	16	13	13	13	4	2
BDVALUES	32	26	211	211	102	100	HS46	32	31	19	18	5	2
BOOTH	3	3	3	3	2	2	HS47	24	22	21	20	5	3
BRATU2D	32	20	3	3	484	400	HS48	4	4	8	8	5	2
BRATU2DT	7	7	8	8	484	400	HS49	31	31	18	18	5	2
BRATU3D	3	3	5	5	125	27	HS50	8	8	11	11	5	3
BROWNALE	6	6	3	3	200	200	HS51	8	8	7	7	5	3
BROYDN3D	5	5	6	6	500	500	HS52	11	11	13	13	5	3
BT10	13	13	19	19	2	2	HS56	11	8	18	16	7	4
BT11	12	12	19	19	5	3	HS61	9	8	16	15	3	2
BT12	14	10	21	21	5	3	HS6	14	11	30	27	2	1
BT1	11	11	48	41	2	1	HS77	12	12	23	22	5	2
BT2	11	11	22	22	3	1	HS78	10	8	12	11	5	3
BT3	10	10	16	16	3	5	HS79	7	7	11	11	5	3
BT4	8	6	28	27	3	2	HS7	9	8	17	17	2	1
BT5	17	13	16	15	3	2	HS8	5	5	10	9	2	2
BT6	12	12	26	24	5	2	HS9	5	5	6	6	2	1
BT7	24	19	48	46	5	3	HYDCAR20	111	98	F	F	99	99
BT8	31	27	25	23	5	2	HYDCAR6	50	43	785	767	29	29
BT9	18	13	20	20	4	2	HYPCIR	4	4	9	8	2	2
BYRDSPHR	14	10	22	21	3	2	INTEGREQ	2	2	4	4	102	100
CATENARY	171	122	58	56	501	166	JUNKTURN	54	34	131	108	510	350
CATENA	449	296	495	417	501	166	LCH	39	25	F	F	600	1
CBRATU2D	6	6	5	5	98	50	LUKVLE10	17	15	F	F	1000	998
CBRATU3D	16	13	5	5	128	16	LUKVLE11	46	46	18	16	998	664
CHAIN	560	251	465	413	402	201	LUKVLE13	45	29	480	406	998	664
CHANDHEU	22	22	14	14	100	100	LUKVLE14	80	51	F	F	998	664
CHNRSBNE	57	41	79	64	50	98	LUKVLE15	67	54	106	101	997	747
CLUSTER	8	8	10	10	2	2	LUKVLE16	47	26	14	14	997	747
CUBENE	10	8	43	37	2	2	LUKVLE17	97	89	340	292	997	747
DECONVNE	59	39	16	13	61	40	LUKVLE18	100	93	187	166	997	747
DTOC1L	9	9	11	10	598	396	LUKVLE1	11	11	24	23	1000	998
DTOC2	12	12	21	21	598	396	LUKVLE2	525	211	36	36	1000	499
DTOC3	5	5	26	26	299	198	LUKVLE3	13	13	29	29	100	2
DTOC4	24	24	17	17	299	198	LUKVLE4	185	109	F	F	1000	499
DTOC5	4	4	23	23	999	499	LUKVLE5	25	21	35	32	102	96
DTOC6	25	17	39	39	201	100	LUKVLE6	20	18	37	37	999	499
DRCAVTY1	5	1	60	55	196	100	LUKVLE7	23	15	44	43	1000	4
DRCAVTY2	5	1	52	49	196	100	LUKVLE8	60	53	351	338	1000	998
DRCAVTY3	5	1	48	43	196	100	LUKVLE9	92	68	190	169	1000	6
EIGENA2	5	5	7	7	6	3	MARATOS	8	5	9	9	2	1
EIGENACO	15	12	19	19	110	55	MWRIGHT	14	11	19	18	5	3
EIGENAU	9	8	16	16	110	110	METHANB8	4	4	41	41	31	31
EIGENB2	10	8	39	33	6	3	METHANL8	11	11	151	147	31	31

Table 4.1: Results for equality constrained problems (continued).

	Alg.A		LANCELOT		Pro. Dim.			Alg.A		LANCELOT		Pro. Dim.	
Problem	f,c	g,A	f,c	g,A	n	m	Problem	f,c	g,A	f,c	g,A	n	m
EIGENB	97	63	145	121	110	110	MSQRTA	7	7	22	17	100	100
EIGENBCO	14	10	20	16	6	2	MSQRTB	7	7	21	18	100	100
EIGENC2	23	15	60	49	30	15	OPTCTRL3	36	36	54	54	302	200
EIGENC	51	36	104	86	462	462	OPTCTRL6	36	36	54	54	302	200
EIGENCCO	35	25	86	70	462	231	ORTHRDM2	6	6	31	27	203	100
ELEC	40	28	48	39	75	25	ORTHRDS2	56	46	83	73	503	250
FLOSP2TH	356	275	F	F	363	323	ORTHREGA	25	25	146	136	517	256
FLOSP2TL	65	43	F	F	363	323	ORTHREGB	3	3	10	9	27	6
FLOSP2TM	486	351	F	F	363	323	ORTHREGC	15	15	30	28	505	250
GENHS28	5	5	11	11	10	8	ORTHREGD	57	46	37	34	503	250
GOTTFR	8	7	17	14	2	2	ORTHRGDM	8	8	32	29	503	250
GRIDNETB	13	13	17	17	924	484	ORTHRGDS	118	86	114	105	503	250
HAGER1	1	1	8	8	201	100	POWELLBS	12	12	48	42	2	2
HAGER2	1	1	6	6	201	100	POWELLSQ	45	37	24	20	2	2
HAGER3	1	1	7	7	201	100	RECIPE	19	19	17	17	3	3
HATFLDF	20	14	62	53	3	3	RSNBRNE	14	11	28	25	2	2
HATFLDG	8	7	16	14	25	25	S316-322	13	13	27	27	2	1
HEART6	504	411	F	F	6	6	SINVALNE	25	16	36	31	2	2
HEART8	40	30	149	127	8	8	SPMSQRT	8	8	13	11	100	164
HIMMELBA	3	3	3	3	2	2	TRIGGER	15	12	21	18	7	6
HIMMELBC	5	5	9	8	2	2	YATP1SQ	10	10	46	37	120	120
HIMMELBE	3	3	6	6	3	3	YATP2SQ	11	11	241	218	120	120
HS100LNP	17	10	23	22	7	2	YFITNE	21	20	88	73	3	17
HS111LNP	24	20	69	62	10	3	ZANGWIL3	7	7	8	8	3	3

We use the performance profile proposed by Dolan and Moré [9] to display the performance of each implementation on the set of test problems, which has some advantages over other existing benchmarking tools, especially for large test sets where tables tend to be overwhelming. Let $l_{p,s}$ denote the number of objective function evaluations required to solve problem p by solver s. Define the performance ratio as

$$r_{p,s} = \frac{l_{p,s}}{l_p^*},$$

where l_p^* is the smallest number of objective function evaluations required by any solver to solve problem p. Therefore, $r_{p,s} \geq 1$ for all p and s. If a solver does not solve a problem, the ratio$r_{p,s}$ is assigned a large number M, which satisfies $r_{p,s} < M$ for all p, s, where solver s succeeds in solving problem p. Then the performance profile for each solver s is defined as the cumulative distribution function for the performance ratio $r_{p,s}$, which is

$$P_s(\tau) = \frac{\text{no. of problems s.t. } r_{p,s} \leq \tau}{\text{total no. of problems}}.$$

Obviously, $P_s(1)$ represents the percentage of problems for which the number of objective function evaluations required is the smallest. For more details about the performance profile please see [9]. The performance profile will also be used to analyze the number of gradient evaluations required. We give the performance profile base on the computational results in Tables 4.1 and 4.2 in Figures 4.1 and 4.2, respectively.

Table 4.2: Results for general problems.

	Alg.B		LANCELOT		Pro. Dim.			Alg.B		LANCELOT		Pro. Dim.	
Problem	f,c	g,A	f,c	g,A	n	m	Problem	f,c	g,A	f,c	g,A	n	m
CB2	17	11	15	15	3	3	LISWET9	28	28	F	F	102	100
CB3	9	9	13	13	3	3	LUKVLI11	46	46	19	18	998	664
CHACONN1	12	9	13	13	3	3	LUKVLI12	195	148	63	52	997	747
CHACONN2	9	9	13	13	3	3	LUKVLI13	276	93	63	53	998	664
CHARDIS1	4	4	369	295	400	199	LUKVLI15	100	79	110	108	97	72
CONGIGMZ	15	13	48	45	3	5	LUKVLI16	58	57	24	26	997	747
DEMYMALO	12	12	20	18	3	3	LUKVLI17	40	28	84	78	97	72
DIPIGRI	11	11	31	28	7	4	LUKVLI18	31	31	23	26	997	747
ELATTAR	83	56	234	216	7	102	LUKVLI1	163	114	F	F	1000	998
EXPFITA	20	19	29	29	5	22	LUKVLI2	114	96	40	39	100	49
EXPFITB	36	36	46	46	5	102	LUKVLI3	40	27	23	23	1000	2
EXPFITC	153	104	84	84	5	502	LUKVLI4	71	48	65	56	100	49
GPP	21	21	86	86	250	498	LUKVLI5	192	132	55	53	102	96
GIGOMEZ1	12	12	26	23	3	3	LUKVLI6	17	17	37	38	999	499
GIGOMEZ2	8	8	17	17	3	3	LUKVLI7	19	19	145	129	1000	4
GIGOMEZ3	9	9	14	14	3	3	LUKVLI8	125	72	315	301	100	98
GOFFIN	10	10	11	11	51	50	LUKVLI9	423	268	70	59	100	6
HAIFAS	11	8	20	18	13	9	MADSEN	14	10	21	21	3	6
HALDMADS	21	20	33	33	6	42	MADSSCHJ	85	80	522	465	198	101
HS100MOD	10	10	66	61	7	4	MAKELA1	13	8	11	11	3	2
HS100	11	11	31	28	7	4	MAKELA2	21	13	29	28	3	3
HS10	25	18	21	21	2	1	MAKELA3	34	24	55	48	21	20
HS113	14	14	37	35	10	8	MAKELA4	8	8	8	8	21	40
HS11	10	10	19	19	2	1	MIFFLIN1	9	7	8	8	3	2
HS12	8	8	17	16	2	1	MIFFLIN2	14	9	27	25	3	2
HS14	10	10	15	15	2	2	MINMAXBD	156	48	266	237	5	20
HS22	8	8	11	11	2	2	MINMAXRB	16	13	51	45	3	4
HS268	5	5	21	21	5	5	OPTMASS	34	25	201	164	70	55
HS29	8	6	19	18	3	1	PENTAGON	31	31	8	8	6	15
HS43	11	11	23	22	4	3	POLAK1	22	21	22	21	3	2
HS88	43	35	49	48	2	1	POLAK3	53	46	42	36	12	10
HS89	39	35	63	58	3	1	POLAK4	14	13	16	15	3	3
HS90	134	104	51	50	4	1	POLAK5	71	54	11	11	3	2
HS91	126	100	49	49	5	1	POLAK6	40	26	97	87	5	4
HS92	244	164	50	49	6	1	PT	22	22	33	33	2	501
KISSING	130	91	140	123	40	91	POWELL20	34	34	47	47	100	100
KIWCRESC	142	108	17	16	3	2	ROSENMMX	22	15	61	54	5	4
LISWET10	26	26	F	F	102	100	S268	5	5	21	21	5	5
LISWET11	25	25	F	F	102	100	SPIRAL	88	57	122	107	3	2
LISWET12	204	204	F	F	102	100	TFI1	118	93	52	47	3	101
LISWET1	27	27	320	320	102	100	TFI2	15	15	27	27	3	101
LISWET2	31	31	355	356	102	100	TFI3	21	21	24	24	3	101
LISWET3	26	26	18	19	402	400	VANDERM1	60	47	16	15	5	9
LISWET4	28	28	19	20	402	400	VANDERM2	111	65	26	24	10	19
LISWET5	26	26	18	19	402	400	VANDERM3	30	27	26	23	4	7
LISWET6	27	27	20	20	402	400	VANDERM4	57	54	84	76	5	9
LISWET7	45	45	F	F	402	400	WOMFLET	181	46	66	33	3	3
LISWET8	38	38	F	F	102	100							

From the performance profile, we can see that the new algorithm outperforms LANCELOT on the selected test problems from CUTEr. Further analyzing the computation results, we found the faster convergence benefits from the combining use of merit function and filter together.

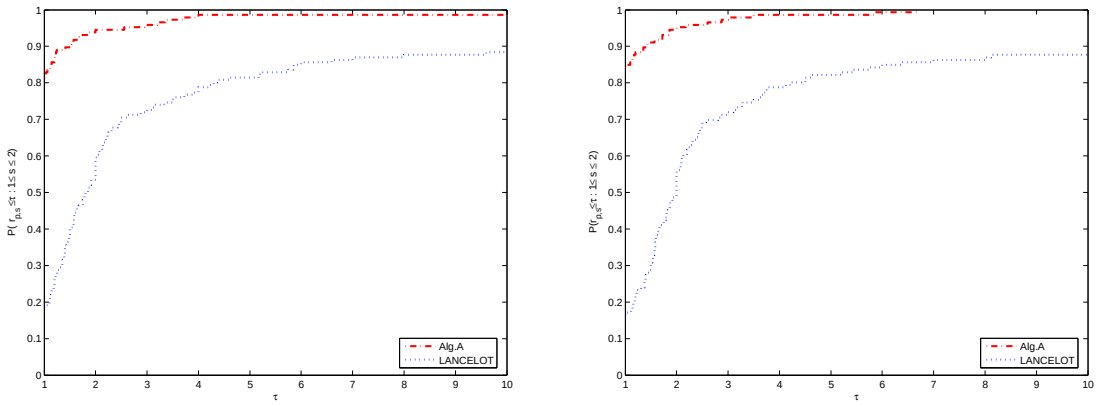

Fig. 4.1. Performance profile between Algorithm A and LANCELOT for the number of function evaluations (left) and gradient evaluations (right)

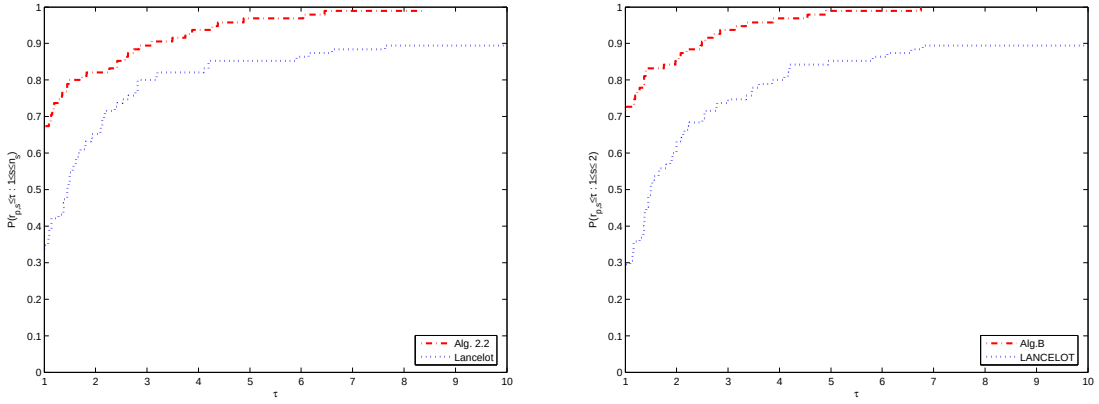

Fig. 4.2. Performance profile between Algorithm B and LANCELOT for the number of function evaluations (left) and gradient evaluations (right)

These new trial point acceptance criteria accelerate the convergence of the iterations on most of the problems, which validates the effectiveness of our original idea in designing the algorithm. This may also explain why LANCELOT has 9 more failures than our new algorithm on the test problems. However, sometimes the new acceptance criteria seem too aggressive on a few problems such as HS90-HS92, which results in the new algorithm needing more iterations than LANCELOT. How to recognize those too aggressive steps and try to avoid them is important for improving our algorithm further.

The other thing we observed from the experiment results is that, the advantage of saving the numbers of function and gradient evaluations for the general constrained problems is not as obvious as for the equality constrained problems. This stems from the fact Algorithm B and LANCELOT using different methods to handle inequality constraints. For our new algorithm, the inequality constraints are handled by the active set method directly. The inexact estimation of the active set during the computing process will cause the increase of iteration numbers. On the other hand, LANCELOT introduces slack variables explicitly and converts the inequality constraints to the equalities. This approach can avoid the increase of iteration numbers during the active set estimation process.

Last but not least, since the merit function in our algorithm is an approximate augmented Lagrangian function, it is expected that Maratos effect will not happen, thus no second order correction steps are needed to force local superlinear convergence. Indeed, fast local convergence behavior of the new algorithm is observed from our numerical experiments. However, we have not yet established the global convergence and local superlinear convergence results for our algorithm, which is a problem that we are still studying.

Acknowledgments. This work was partially supported by NSFC Grant 10831006 and CAS grant kjcx-yw-s7. Part of the results were presented at the International Conference on Numerical Analysis and Optimization dedicated to M.J.D. Powell's 70th birthday, held in September, 2006 in Beijing. The authors would like to thank an anonymous referee whose comments on an earlier version of the paper helped us greatly.

References

[1] R.H. Byrd, R.B. Schnabel and G.A. Shultz, A trust region algorithm for nonlinearly constrained optimization, *SIAM J. Numer. Anal.*, **24** (1987), 1152-1170.

[2] M.R. Celis, J.E. Dennis and R.A. Tapia, A trust region algorithm for nonlinear equality constrained optimization, in: P.T. Boggs, R.H. Byrd and R.B. Schnabel, eds., *Numerical Optimization*, SIAM, Philadelphia, USA, 1985.

[3] L. Chen and D. Goldfarb, Interior point l_2-penalty methods for nonlinear programming with strong global convergence properties, *Math. Program.*, **108** (2006), 1-36.

[4] T.F. Coleman and Y. Li, An interior trust region approach for nonlinear minimization subject to bounds, *SIAM J. Optimiz.*, **6** (1996), 418-445.

[5] A.R. Conn, N.I.M. Gould, D. Orban and Ph.L. Toint, A primal-dual trust region algorithm for non-convex nonlinear programming, *Math. Program.*, **87** (2000), 215-249.

[6] A.R. Conn, N.I.M. Gould and Ph.L. Toint, *LANCELOT: a Fortran package for large-scale nonlinear optimization (Release A)*, Springer, Heidelberg, New York. USA, 1992.

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

[8] J.E. Dennis and H.H.W. Mei, Two new unconstrained optimizaiton algorithms which use function and gradient values, *J. Optimiz. Theory App.*, **28** (1979), 453-482.

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

[10] R. Fletcher, A model algorithm for composite NDO problem, *Math. Program. Study*, **17** (1982), 67-76.

[11] R. Fletcher, *Practical Methods of Optimization, Second Edition*, John Wiley and Sons, New York, USA, 1987.

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

[13] D. Gay, Computing optimizal local constrained step, *SIAM J. Sci. Stat. Comp.*, **2** (1981), 186-197.

[14] N.I.M, Gould, D. Orban and Ph.L. Toint, *CUTEr (and SifDec), a constrained and unconstrained testing environment, revisited*, Technical Report TR/PA/01/04, CERFACS, Toulouse, France, 2001.

[15] G.H. Golub and U. Von Matt, Quadratically constrained least squares and quadratic problems, *Numer. Math.*, **59** (1991), 561-580.

[16] J.J. Moré and D.C. Sorensen, Computing a trust region step, *SIAM J. Sci. Comput.*, **4** (1983), 553-572.

[17] J. Nocedal and S. Wright, *Numerical Optimization*, Springer, New York, USA, 1999.

[18] E.O. Omojokun, *Trust Region Algorithms for Optimization with Nonlinear Equality and Inequality Constraints*, Ph.D. Thesis, University of Colorado at Boulder, USA, 1989.

[19] M.J.D. Powell, A method for nonlinear constraints in minimization problems, in: R. Fletcher, ed., *Optimization*, Academic Press, London, 1969.

[20] M.J.D. Powell, A new algorithm for unconstrained optimization, in: J.B. Rosen, O.L. Mangasarian and K. Ritter, eds., *Nonlinear Programming*, Academic Press, New York, 1970, 31-66.

[21] M.J.D. Powell and Y.X. Yuan, A trust region algorithm for equality constrained optimization. *Math. Program.*, **49** (1991), 189-213.

[22] F. Rendl and H. Wolkowicz, A semidefinite framework for trust region subproblems with applications to large scale minimization, *Math. Program.*, **77** (1997), 273-299.

[23] T. Steihaug, The conjugate gradient method and trust regions in large scale optimization, *SIAM J. Numer. Anal.*, **20** (1983), 626-637.

[24] Ph.L. Toint, Towards an efficient sparsity exploiting Newton method for minimization, in: I.S. Duff, ed., *Sparse Matrices and Their Uses*, Academic Press, London, 1981, 57-88.

[25] M. Ulbrich and S. Ulbrich, Non-monotone trust region methods for nonlinear equality constrained optimization without a penalty function, *Math. Program., Serial B.*, **95** (2003), 103-135.

[26] A. Vardi, A trust region algorithm for equality constrained minimization: convergence properties and implementation, *SIAM J. Numer. Anal.*, **22** (1985), 575-591.

[27] Y.X. Yuan, On the convergence of a new trust algorithm, *Numer. Math.*, **70** (1995), 515-539.

[28] Y.X. Yuan, A review of trust region algorithms for optimization, in: J.M. Ball and J.C.R. Hunt, eds., *ICIAM 99: Proceedings of the Fourth International Congress of Industrial and Applied Mathematics*, Oxford University Press, 2000, 271-282.

[29] Y.X. Yuan, On the truncated conjugate gradient method, *Math. Program.*, **87** (2000), 561-573.