Abstract

In this paper, we consider the quasi-Newton methods for the large scale unconstrained minimization problem. The conventional quasi-Newton methods such as the BFGS and DFP methods are not applicable to large scale problems since they use a dense approximate Hessian of the objective function. The true Hessian in large scale problems is usually sparse. Recently a method taking advantage of this sparsity has been proposed as the Matrix Completion Quasi-Newton (MCQN) Method. The MCQN method uses an approximate matrix of the inverse of the Hessian, and it is implemented with fewer space complexities when the true Hessian is sparse. However, the MCQN method converges slowly when an initial approximate matrix is not given appropriately. In addition, it cannot be applied if the true Hessian is dense.

To overcome the above difficulties, we propose the tridiagonal MCQN update in which the approximate Hessian is limited to be only tridiagonal. Because of this limitation, the memory to store the approximate Hessian is $O(n)$. However, when the true Hessian is not tridiagonal, the fast convergence is not guaranteed. Therefore, we combine the LBFGS and the tridiagonal MCQN methods. From several numerical experiments, we discuss the effectiveness of the proposed method.

Contents

1 Introduction

In the present paper, we consider the following unconstrained minimization problem.

$$\begin{array}{ll} \min & f(x) \\ \text{subject to} & x \in R^n. \end{array} \tag{1.1}$$

Throughout the paper, it is assumed that f is twice continuously differentiable, and n is huge.

For solving the unconstrained minimization problem, there exist several useful solution methods, including the steepest descent method, the Newton and quasi-Newton methods, and the conjugate gradient method [11]. The quasi-Newton method, in particular the BFGS method, is widely used, because it is easy to implement and has good convergence properties. The quasi-Newton method generates a sequence $\{x_k\}$ with using an approximate matrix B_k of the Hessian $\nabla^2 f(x_k)$ at the k-th iterate x_k. The approximate Hessian B_k is usually generated by the BFGS and DFP update formulas [4, 6]. Since the quasi-Newton method with the BFGS update converges superlinearly under some mild assumptions, it can obtain a solution of (1.1) with the relatively small number of iterations. However, the DFP and BFGS updates usually generate dense approximate Hessians, and therefore it requires $O(n^2)$ time and space complexities at each iteration. Thus we cannot apply the quasi-Newton method to large scale problems. To overcome this difficulty, the Limited memory BFGS (LBFGS) method [10] and the Matrix Completion quasi-Newton (MCQN) method [12] has been proposed.

The LBFGS method was proposed by Nocedal [10] in 1980. It constructs an approximate Hessian by using a few vector pairs and a diagonal matrix. Therefore, it can reduce the time and space complexities per iteration effectively. However, because of the lack of the full information on $n \times n$ matrix, only linear convergence is guaranteed [9]. Furthermore, this method converges very slowly for ill-conditioned problems.

The Hessian $\nabla^2 f(x)$ in large scale problems is usually sparse. The MCQN method takes advantage of this sparsity to update the approximate matrix H_k of $\nabla^2 f(x_k)^{-1}$ [12]. In the MCQN method, as a first step, some elements corresponding to nonzero elements of the true Hessian, that is, $(H_{k+1})_{ij}$ for $(i,j) \in E := \left\{(i,j) \;\middle|\; \left(\nabla^2 f(x)\right)_{ij} \neq 0 \; for\ some\ x \in R^n\right\}$, are updated by using the existing quasi-Newton updates, such as the BFGS and DFP updates. As a second step, we consider a maximum-determinant positive definite matrix completion of the matrix H_{k+1} with the elements $(H_{k+1})_{ij}, (i,j) \in E$ fixed. Then we adopt the maximum determinant matrix as the next approximate matrix H_{k+1}. Throughout this paper, a graph $G = (V, E)$ with $V = \{1, \ldots, n\}$ is called as a sparse graph. If the graph G is a chordal graph, the approximate matrix H_{k+1} is given as a product of sparse matrices explicitly [7]. However, the sparse graph is not necessarily chordal. In such cases, the approximate matrix H_{k+1} is updated by a set F instead of E, where F is an extension of E and its corresponding sparse graph $G = (V, F)$ is chordal. If the Hessian is sparse, then the MCQN method requires lower space and time complexities than those for the existing quasi-Newton methods. Theoretically, the MCQN method has nice convergence properties. In particular, it is shown that a sequence generated by the MCQN method has a local and superlinear convergence under the same assumptions of the DFP method. However even if $\nabla^2 f(x)$ is sparse, $|F|$ may be much larger than $|E|$. Moreover, if the true Hessian is dense, then $|E|$ is almost n^2. Therefore, it makes no sense to apply this method to problems which do not have the sparsity. Furthermore, if the initial approximate matrix H_0 is not given appropriately, the generated sequence converges to an optimal solution very slowly. We will explain the details of the slow convergence in Section 3.

To overcome the above difficulties of the MCQN method, in the present paper, we propose a method that combines the LBFGS and the MCQN methods. The proposed method is based on three ideas. As the first idea, the approximate Hessian $B_k = H_k^{-1}$ by the MCQN update is limited to be only tridiagonal, that is, the set F is given as $F = \{(i, j)|\, |i - j| \leq 1\}$. Then the graph $G = (V, F)$ is chordal, and hence, the time and space complexities per iteration of the MCQN update with F are $O(n)$. This method is called as a Tri-MCQN method. As mentioned above, the approximate matrix updated by MCQN is sometimes inappropriate. Then the generated sequence might converge to an optimal solution slowly. As the second idea, we also apply the full BFGS updates as well as Tri-MCQN. We construct the k-th approximate matrix H_k by applying the Tri-MCQN update until $(k - m)$-th iteration and by the existing BFGS update after the reminder m iterations, that is, from $(k-m+1)$-th to k-th iteration. The last m times BFGS updates are implemented by the LBFGS update with m vector pairs. Note that the usual LBFGS constructs an approximate matrix with the initial matrix $H_{k-m} = y_k^\top s_k / y_k^\top y_k I$. Therefore the second idea is regarded as a change of this initial diagonal matrix H_{k-m} for the matrix updated by Tri-MCQN. As a consequence, the approximate Hessian of the proposed method is expected to be better than that of the LBFGS update. However, we discussed in Section 3, if the initial approximate matrix H_0 is far from the inverse of the true Hessian $\nabla^2 f(x_0)$, the method based on the first and the second ideas may be worse than the LBFGS. Therefore, as the third idea, if a search direction calculated with H_k by the above two ideas is not suitable, we perform a restart, that is, we discard the current matrix H_{k-m} updated by the Tri-MCQN method and set an initial matrix again. In Section 4, we give concrete criteria on the restart.

The present paper is organized as follows. In Section 2, the existing quasi-Newton methods are introduced, and these advantages and disadvantages are discussed. In Section 3, we present an example of a problem in which a sequence updated by the MCQN method converges very slowly. In Section 4, in order to overcome the disadvantages of the existing methods, the method which combines the LBFGS and the MCQN is proposed. In Section 5, a number of numerical experiments for large scale unconstrained problems are presented. We make some concluding remarks in Section 6.

2 The existing quasi-Newton Methods

In this section, we describe the existing quasi-Newton methods, and discuss about these advantages and disadvantages.

2.1 Quasi-Newton Methods

The quasi-Newton method generates a sequence $\{x_k\}$ by using the approximate Hessian of the objection function f. Throughout the rest of this paper, B_k denotes an approximate Hessian of $\nabla^2 f(x_k)$. An inverse of the approximate Hessian B_k is denoted by $H_k = B_k^{-1}$. Usually, we suppose that B_k and H_k are positive definite. The quasi-Newton method adopts a search direction d_k given by

$$d_k = -H_k \nabla f(x_k). \tag{2.1}$$

When H_k is positive definite, d_k is a descent direction of f. Thus, we can choose a stepsize so as to satisfy the following Wolfe conditions.

$$f(x_k + \alpha d_k) \leq f(x_k) + \xi_1 \alpha \nabla f(x_k)^\top d_k, \tag{2.2}$$

$$\xi_2 \nabla f(x_k)^\top d_k \leq \nabla f(x_k + \alpha d_k)^\top d_k, \tag{2.3}$$

where ξ_1, ξ_2 are constants such that $0 < \xi_1 < \xi_2 < 1$.

Using the direction d_k and the stepsize α_k, the next iterate x_{k+1} is given by

$$x_{k+1} = x_k + \alpha_k d_k.$$

The matrix H_k is updated by some formulae in each iteration. We will discuss the formulae later. A general form of the quasi-Newton method is described as follows.

The quasi-Newton method

Step 0: Choose an initial point x_0 and an initial symmetric positive definite matrix $H_0 \in R^{n \times n}$. Set $k = 0$.

Step 1: If x_k satisfies the termination criterion, output x_k as an optimal solution and stop. Otherwise go to Step 2.

Step 2: Obtain the search direction $d_k = -H_k \nabla f(x_k)$.

Step 3: Choose the step size α_k satisfying the Wolfe conditions (2.2) and (2.3), and set $x_{k+1} = x_k + \alpha_k d_k$.

Step 4: Update H_{k+1} by some quasi-Newton update formulae.

Step 5: Set $k := k + 1$ and go to Step 1.

For the rapid convergence, we usually enforce the matrix H_{k+1} to satisfy the following secant condition.

$$H_{k+1} y_k = s_k, \tag{2.4}$$

where

$$s_k = x_{k+1} - x_k \tag{2.5}$$

$$y_k = \nabla f(x_{k+1}) - \nabla f(x_k). \tag{2.6}$$

There are various update methods of H_k that satisfy the secant condition (2.4). Among them, BFGS and DFP are well-known.

The DFP update adopts a solution of the following problem as H_{k+1}.

$$\begin{array}{ll} \min_H & \psi(H_k^{-\frac{1}{2}} H H_k^{-\frac{1}{2}}) \\ \text{subject to} & H y_k = s_k \\ & H = H^\top \\ & H \succeq 0, \end{array} \tag{2.7}$$

where $\psi : R^{n \times n} \to R$ is a strictly convex function defined as $\psi(A) = trace(A) - \ln det(A)$. Moreover, $H \succeq 0$ ($H \succ 0$) means that H is positive semidefinite (positive definite) . When H_k is positive definite and the condition $s_k^\top y_k > 0$ is satisfied, the optimal solution H_{k+1}^{DFP} of the problem (2.7) is unique, and given by

$$H_{k+1}^{\mathrm{DFP}} = H_k - \frac{H_k y_k (H_k y_k)^\top}{y_k^\top H_k y_k} + \frac{s_k s_k^\top}{s_k^\top y_k}.$$

The BFGS update is similar to the DFP update. The next matrix B_{k+1}^{BFGS} is a solution of the problem (2.7) where H_k, H, s_k and y_k are replaced with B_k, B, y_k and s_k, respectively. The inverse of B_{k+1}^{BFGS} is given by

$$H_{k+1}^{\mathrm{BFGS}} = H_k - \frac{H_k y_k s_k^\top + s_k (H_k y_k)^\top}{s_k^\top y_k} + \left(1 + \frac{y_k^\top H_k y_k}{s_k^\top y_k}\right) \frac{s_k s_k^\top}{s_k^\top y_k}. \tag{2.8}$$

Even if the true Hessian $\nabla^2 f(x)$ is sparse, the approximate Hessian $B_{k+1} = H_{k+1}^{-1}$ updated by DFP and BFGS becomes dense. Therefore, those updates require $O(n^2)$ memory to store the matrix B_{k+1} or H_{k+1}, and then these update formulas cannot be used for large scale problems.

2.2 Limited memory BFGS method

In order to overcome the disadvantage of the quasi-Newton method, the LBFGS method is proposed by Nocedal in 1980. Unlike the BFGS method which stores a dense matrix, the LBFGS method stores only a few vector pairs and a diagonal matrix. It never explicitly forms or stores a full matrix. Instead, it maintains a history of the past updates of x_k and $\nabla f(x_k)$. Therefore, the time and space complexities per iteration are cut down dramatically. The essential idea of LBFGS is described as follows.

First, note that the BFGS formula (2.8) at the k-th update is written as

$$H_k = V_{k-1}^\top H_{k-1} V_{k-1} + T_{k-1}, \tag{2.9}$$

where

$$V_i = I - \frac{y_i s_i^\top}{s_i^\top y_i}, \; T_i = \frac{s_i s_i^\top}{s_i^\top y_i}.$$

Using the formula (2) recurrently from the $(k-m)$-th iteration, we have

$$\begin{aligned} H_k \;=\; & (V_{k-m} V_{k-m+1} \cdots V_{k-2} V_{k-1})^\top H_{k-m} (V_{k-m} V_{k-m+1} \cdots V_{k-2} V_{k-1}) \\ & + (V_{k-m+1} V_{k-m+2} \cdots V_{k-2} V_{k-1})^\top T_{k-m} (V_{k-m+1} V_{k-m+2} \cdots V_{k-2} V_{k-1}) + \cdots \\ & + (V_{k-2} V_{k-1})^\top T_{k-3} (V_{k-2} V_{k-1}) + V_{k-1}^\top T_{k-2} V_{k-1} + T_{k-1}. \end{aligned}$$

Here H_{k-m} is the approximate Hessian at the $(k-m)$-th iteration. The matrix H_{k-m} is dense in general. The LBFGS method replaces the dense matrix H_{k-m} with some diagonal matrix H_{k-m}^0. Usually we use H_{k-m}^0 given by

$$H_{k-m}^0 = \frac{s_k^\top y_k}{y_k^\top y_k} I, \tag{2.10}$$

where I is the unit matrix.

A product of some vector $w \in R^n$ and H_k is calculated by using vectors s_i, y_i $(i = k-m, \ldots, k-1)$ and H_{k-m}^0 without a explicit calculation of H_k. Therefore, only $2m$ vectors and the diagonal matrix H_{k-m}^0 is required to implement the algorithm. Thus, the time and space complexities of the LBFGS method per iteration are $O(mn)$, which are much lower than those of the usual BFGS method. On the other hand, only linear convergence is guaranteed for the LBFGS method, because little information of the true Hessian can be included in the approximate matrix when m is small. Actually, it is reported that the LBFGS converges slowly for ill-conditioned problems. However, when an accurate solution is not required, the LBFGS method is very practical.

2.3 Matrix Completion quasi-Newton Method

We describe the MCQN method, and its advantages and disadvantages.

The Hessian $\nabla^2 f$ of the large scale problem is usually sparse. Then, it is better to use a sparse matrix with the same sparse structure of $\nabla^2 f(x)$ as an approximate Hessian. Thus we add a sparsity constraint to the problem (2.7) of DFP.

$$\begin{array}{ll} \min_{\mathrm{H}} & \psi(H_k^{-\frac{1}{2}} H H_k^{-\frac{1}{2}}) \\ \text{subject to} & H y_k = s_k \\ & H = H^\top \\ & H \succeq 0 \\ & (H^{-1})_{ij} = 0 \ \ \forall (i,j) \notin F, \end{array} \tag{2.11}$$

where $F \supseteq E := \left\{(i,j) \mid \left[\nabla^2 f(x)\right]_{ij} \neq 0 \ for\ some\ x \in R^n\right\}$. Here we assume that (1) $(i,i) \in F, i = 1,2,\ldots,n,$ (2) $(i,j) \in F \Rightarrow (j,i) \in F$. It is desirable to choose F as $F \approx E$.

Unfortunately, an optimal solution of the problem (2.11) cannot be obtained explicitly. Therefore, the MCQN adopts an approximate solution H_{k+1} obtained by the following algorithm.

The MCQN method

Step 1: Obtain a partial matrix $\left(\bar{H}_{k+1}\right)_{ij}, \forall (i,j) \in F$ using the existing quasi-Newton updates from H_k.

Step 2: Obtain a solution H_{k+1} of the following problem with $\left(\bar{H}_{k+1}\right)_{ij}, (i,j) \in F$ as the given constants.

$$\begin{array}{ll} \min_{\mathrm{H}} & \psi(H_k^{-\frac{1}{2}} H H_k^{-\frac{1}{2}}) \\ \text{subject to} & H_{ij} = \left(\bar{H}_{k+1}\right)_{ij} \ \ \forall (i,j) \in F \\ & H = H^\top \\ & H \succeq 0 \\ & (H^{-1})_{ij} = 0 \ \ \forall (i,j) \notin F \end{array} \tag{2.12}$$

The problem (2.12) replaces the secant condition $H y_k = s_k$ of the problem (2.11) by the constraints $H_{ij} = \left(\bar{H}_{k+1}\right)_{ij}, \forall (i,j) \in F$. Thus, the update matrix H_{k+1} is regarded as a kind of approximate solution of (2.11). In the rest of this section, $\bar{H}_{k+1}$ is simply written as $\bar{H}$.

The problem (2.12) still appears to be difficult to solve. However, if F satisfies the following condition, then the solution is given explicitly [7].

(*The chordal condition*) An undirected graph $G' = \left(V, \bar{F}\right)$ with a vertex set $V = \{1,\ldots,n\}$ and an edge set $\bar{F} := \{F \setminus (i,i) \mid i = 1,\ldots,n\}$ is a chordal graph.

It is desirable to choose $F = E$ for fully exploiting the sparse pattern of the problem. However, the sparse graph $G = \left(V, \bar{E}\right)$ of the Hessian is not necessarily chordal. Therefore, it is proposed to use a set F satisfying $F \supseteq E$ and *the chordal condition*.

Next, when F satisfies *the chordal condition*, we see that the solution of the problem (2.12) is expressed as a product of sparse matrices. Let $\{C_r\}$ be a set of maximum cliques of the chordal graph $\left(V, \bar{F}\right)$. Then families $\{S_r\}$ and $\{U_r\}$ are defined as

$$S_r := C_r \setminus (C_{r+1} \cup C_{r+2} \cup \cdots \cup C_l),\ r = 1, \dots, l$$
$$U_r := C_r \cap (C_{r+1} \cup C_{r+2} \cup \cdots C_l),\ r = 1, \dots, l,$$

where l is the number of the maximum cliques. Note that $V = \cup_{r=1}^{l} S_r$ and $S_i \cap S_j = \emptyset$. We can thus obtain a perfect elimination ordering [7] of the vertices, in which the vertices in S_r are given consecutive numbers for each r. Let P denote a permutation matrix corresponding to the ordering. Then, the solution of the problem (2.12) is given as follows.

$$\bar{H} = P^\top \left[L_1^{(k+1)}\right]^\top \left[L_2^{(k+1)}\right]^\top \cdots \left[L_{l-1}^{(k+1)}\right]^\top D^{(k+1)} \left[L_{l-1}^{(k+1)}\right] \cdots \left[L_2^{(k+1)}\right]\left[L_1^{(k+1)}\right] P, \tag{2.13}$$

where the matrices $L_r^{(k+1)}$ and $D^{(k+1)}$ are given by

$$\left[L_r^{(k+1)}\right]_{ij} = \begin{cases} 1 & i = j \\ \left[\left(\bar{H}_{U_rU_r}\right)^{-1} \bar{H}_{U_rS_r}\right]_{ij} & (i, j) \in U_r \times S_r \\ 0 & \text{otherwise} \end{cases} \tag{2.14}$$

for $r = 1, 2, \dots, l-1$, and

$$D^{(k+1)} = \begin{pmatrix} D_{S_1S_1}^{(k+1)} & & & \\ & D_{S_2S_2}^{(k+1)} & & \\ & & \ddots & \\ & & & D_{S_lS_l}^{(k+1)} \end{pmatrix} \tag{2.15}$$

with

$$D_{S_rS_r}^{(k+1)} = \begin{cases} \bar{H}_{S_rS_r} - \bar{H}_{S_rU_r}\left(\bar{H}_{U_rU_r}\right)^{-1}\bar{H}_{U_rS_r} & r \le l-1 \\ \bar{H}_{S_rS_r} & r = l. \end{cases}$$

Note that H_{k+1} is calculated by using only $\bar{H}_{ij}, \forall(i, j) \in F$. Moreover, if $|F|$ is small, the matrices $L_r^{(k+1)},\ r = 1, \dots, l$ are sparse lower triangle matrices and $D^{(k+1)}$ is a block diagonal matrix.

The MCQN method requires lower space and time complexities than the quasi-Newton method. Moreover, it is shown that any sequence generated by the MCQN update converges locally and superlinearly under the same assumptions of the DFP update [12]. The MCQN can be implemented with only $(H_k)_{ij}, \forall(i, j) \in F$. Then its space complexity is $O(|F|)$ and the time complexity per iteration is $O(\Sigma_{r=1}^{l}|C_r|^3)$. On the other hand, when we implement it with $(H_k)_{ij}, \forall(i, j) \in F$ and $((H_k)_{U_rU_r})^{-1}, r = 1, 2, \dots, l-1$, the space complexity of the MCQN update is $O(|F| + \Sigma_{r=1}^{l}|U_r|^2)$ and the time complexity per iteration is $O(\Sigma_{r=1}^{l}|C_r|^2)$.

The MCQN method requires an additional work to construct the set F from the set E of the true Hessian [1]. In addition, depending on the problems, the size of F becomes large compared to E. Furthermore, if the initial approximate matrix is not given appropriately, the generated sequence converges to the optimal solution very slowly. In the next section, we describe such an example.

3 An example where the MCQN method converges slowly

The MCQN method sometimes converges slowly as compared to the LBFGS method. In this section, we present a simple example where such a situation actually occurs.

The main reason for the slow convergence of the MCQN method is that it does not exploit the information on the elements of $(i, j) \notin F$. Each $H_{ij}, (i, j) \in F$ of the MCQN with the BFGS update in the second step is given as

$$H_{ij}^{\mathrm{BFGS}} = (H_k)_{ij} + \left(\frac{1}{s_k^\top y_k} + \frac{(y_k)^\top H_k y_k}{(s_k^\top y_k)^2} \right) s_i s_j - \frac{(H_k y_k)_i (s_k)_j + (s_k)_j (H_k y_k)_j}{s_k^\top y_k}, \ \forall (i, j) \in F. \tag{3.1}$$

In general, the numerators of the second and third terms in (3.1) are very small in comparison with their denominator. Although the pure BFGS also has the same circumstance, the total sum of the changes of the pure BFGS update is

$$\sum_i^n \sum_j^n \left| (H_k)_{ij} - H_{ij}^{\mathrm{BFGS}} \right| = \sum_{i=1}^n \sum_{j=1}^n \left| \left(\frac{1}{s_k^\top y_k} + \frac{(y_k)^\top H_k y_k}{(s_k^\top y_k)^2} \right) s_i s_j - \frac{(H_k y_k)_i (s_k)_j + (s_k)_j (H_k y_k)_j}{s_k^\top y_k} \right|,$$

which is not so small. On the other hand, the total sum of the changes of the MCQN is often small. This happens because the elements $H_{ij}^{\mathrm{BFGS}}, (i, j) \notin F$ are not used, and are given as a solution of the maximum-determinant positive definite matrix completion.

To see this, consider the unconstrained minimization problem of $f(x) = \sum_{i=1}^n x_i^2/2$. Let $x_0 = (1, \dots, 1)^\top$ and $H_0 = \delta I$ where δ is some small constant. Then the next iterate is given as $x_1 = (1 - \delta, \dots, 1 - \delta)^\top$. Moreover, $s_0 = (-\delta, \dots, -\delta)^\top$, $y_0 = (-\delta, \dots, -\delta)^\top$, $H_0 y_0 = (-\delta^2, \dots, -\delta^2)^\top$ and $s_0^\top y_0 = n\delta^2$. Therefore, the BFGS update is given by

$$(H_1)_{ij}^{\mathrm{BFGS}} = \begin{cases} \delta + \dfrac{1}{n} - \dfrac{\delta}{n} & i = j \\ \dfrac{1}{n} - \dfrac{\delta}{n} & i \neq j. \end{cases}$$

Since $E = \{(i, j) \mid i = j\}$, we have $(H_k)_{ij} = 0, \ i \neq j$. Moreover, when δ is small and n is large, the diagonal elements of H_k remain small during lots of iterations. As a result, $d_k = -H_k \nabla f(x_k)$ is also small, and hence $\|x_{k+1} - x_k\|$ is small. Therefore, if the MCQN method is applied to this problem with $H_0 = \delta I$, a lot of iterations are needed.

4 The proposed method

One of the drawbacks of the LBFGS method is that the convergence could be slow if the initial approximate Hessian H_{k-m}^0 is not appropriate. For the MCQN method, there exist the following problems.

- An additional work is required to obtain the chordal extention F of E.
- There is no difference between the MCQN and the usual quasi-Newton method for a problem that has a dense Hessian.
- The convergence may be slow if the initial approximate matrix is not appropriate.

In order to overcome such difficulties, we propose a method that combines the LBFGS and MCQN method. The proposed method is based on the following three ideas.

(1)Restriction of the approximate Hessian B_k to be tridiagonal

If the Hessian $\nabla^2 f(x)$ is tridiagonal, the corresponding sparse graph (V, E) is already chordal. Then, as discussed in detail below, the time and space complexities per iteration of the MCQN with E are $O(n)$. Therefore, even if the Hessian is not tridiagonal, we enforce $B_k = H_k^{-1}$ by the MCQN to be tridiagonal, that is, $F = \{(i,j) \mid |i-j| \leq 1\}$. In the rest of this paper, this MCQN method is called as the Tridiagonal MCQN (Tri-MCQN) method. The approximate Hessian updated by this method is written as H^{Tri}.

The sparse graph corresponding to the tridiagonal matrix is a chain graph as in Figure 1. The chain graph is chordal.

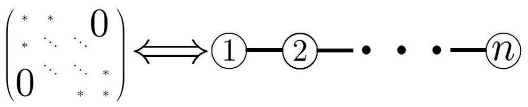


Figure 1: The tridiagonal matrix and the chain graph

The number of maximal cliques of the chain graph is $n-1$, and the maximal cliques are $C_r = \{r, r+1\}$, $r = 1, \ldots, n-1$. Then, $S_r = \{r\}$, $r = 1, \ldots, n-2, S_{n-1} = \{n-1, n\}$, $U_r = \{r+1\}$, $r = 1, \ldots, n-2$, $U_{n-1} = \emptyset$. Therefore, $L_{n-1} = I$, and L_r, $r = 1, \ldots, n-2$ are given by

$$[L_r]_{ij} = \begin{cases} 1 & i = j \\ \bar{H}_{r+1,r}/\bar{H}_{(r+1),(r+1)} & (i,j) = (r+1, r) \\ 0 & \text{otherwise.} \end{cases}$$

Furthermore, D_r is given by

$$D_r = \begin{cases} \bar{H}_{r,r} - \left(\bar{H}_{r,r+1}\right)^2 / \bar{H}_{(r+1),(r+1)} & r \leq n-2 \\ \bar{H}_{S_r,S_r} & r = n-1. \end{cases}$$

Consequently, if $\bar{H}_{i,i}$, $i = 1, \ldots, n$ and $\bar{H}_{i+1,i}$, $i = 1, \ldots, n-1$ are stored, each L_r and D_r are obtained in $O(1)$. Therefore, the time complexity is $O(n)$. Note that the matrix H_k is used only when $d_k = -H_k \nabla f(x_k)$ and $H_k y_k$ are calculated. Therefore, we do not need to store H_k explicitly. We only need to store $(H_k)_{ij}, \forall (i,j) \in \bar{F}$, and update these elements in each iteration, where $\bar{F} = \{(i,j) \mid i < j,\ |i-j| < 1\}$. In practical, $H_k\omega$, which is a product of H_k and some vector $\omega \in R^n$, is computed as follows, when $(H_k)_{ij}, \forall (i,j) \in \bar{F}$ is stored.

First note that the product $H_k\omega$ is written as

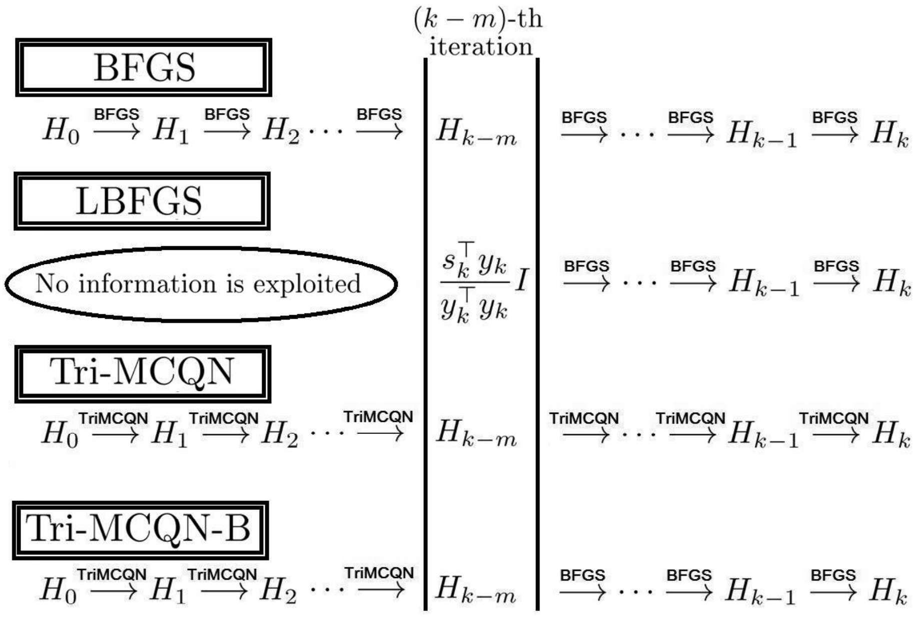
$$H_k\omega = \left[L_1^{(k)}\right]^\top \left[L_2^{(k)}\right]^\top \cdots \left[L_{l-1}^{(k)}\right]^\top D L_{l-1}^{(k)} \cdots L_2^{(k)} L_1^{(k)} \omega.$$

Then the vector $H_k\omega = \omega_{2l-1}$ is obtained by calculating this equation from right side as $\omega_1 = L_1^{(k)} P\omega$, $\omega_2 = L_2^{(k)}\omega_1$, $\cdots$, $\omega_{l-1} = L_{l-1}^{(k)}\omega_{l-2}$, $\omega_l = D^{(k)}\omega_{l-1}$, $\omega_{l+1} = \left[L_{l-1}^{(k)}\right]^\top \omega_l$, $\cdots$, $\omega_{2l-1} = P^\top \left[L_1^{(k)}\right]^\top \omega_{2l-2}$.

Consequently, the time complexity of the Tri-MCQN method is $O(n)$.

(2)Applying the pure BFGS update from the $(k-m)$-th iteration

When the approximate Hessian by the Tri-MCQN update is far from the true Hessian, the MCQN method may converge slowly. Then, the k-th approximate Hessian H_k is updated by the Tri-MCQN update until $(k - m)$-th iterate, and by the pure BFGS update after the reminder m iterates, that is, from $(k - m + 1)$-th to k-th iterate. The last m times BFGS updates are implemented by the LBFGS update with m vector pairs. We will give an appropriate m by means of some numerical experiments in Section 4. The update idea of some method is described in Figure 2. Tri-MCQN-B means a method containing the idea (1) and (2).



$(k-m)$-th iteration
BFGS
$H_0 \xrightarrow{\text{BFGS}} H_1 \xrightarrow{\text{BFGS}} H_2 \cdots \xrightarrow{\text{BFGS}} H_{k-m} \xrightarrow{\text{BFGS}} \cdots \xrightarrow{\text{BFGS}} H_{k-1} \xrightarrow{\text{BFGS}} H_k$
LBFGS
No information is exploited
$\frac{s_k^\top y_k}{y_k^\top y_k} I \xrightarrow{\text{BFGS}} \cdots \xrightarrow{\text{BFGS}} H_{k-1} \xrightarrow{\text{BFGS}} H_k$
Tri-MCQN
$H_0 \xrightarrow{\text{TriMCQN}} H_1 \xrightarrow{\text{TriMCQN}} H_2 \cdots \xrightarrow{\text{TriMCQN}} H_{k-m} \xrightarrow{\text{TriMCQN}} \cdots \xrightarrow{\text{TriMCQN}} H_{k-1} \xrightarrow{\text{TriMCQN}} H_k$
Tri-MCQN-B
$H_0 \xrightarrow{\text{TriMCQN}} H_1 \xrightarrow{\text{TriMCQN}} H_2 \cdots \xrightarrow{\text{TriMCQN}} H_{k-m} \xrightarrow{\text{BFGS}} \cdots \xrightarrow{\text{BFGS}} H_{k-1} \xrightarrow{\text{BFGS}} H_k$

Figure 2: The differences between the proposed method and the existing quasi-Newton methods

(3)Restart of the initial approximate matrix

The Tri-MCQN-B method converges fast, if the true Hessian of the problem is tridiagonal and the approximate Hessian is close to the true Hessian. However, if the approximate Hessian is far from the true Hessian, the convergence may be slow because of the reason described in Section 3. Therefore, to overcome this difficulty, if a search direction d_k is not suitable, we perform a restart, that is, we discard the current matrix H_{k-m}^{Tri} and set an initial matrix again.

As the restart condition, we propose the following conditions with the stepsize α_k and the search direction d_k.

The restart condition

Let δ, $\alpha_{\min}$, $\alpha_{\max}$, c_1, c_2 be positive constants satisfying $\delta \in (0, 1)$, $\alpha_{\min} \leq \alpha_{\max}$ and $c_2 \leq c_1$. If all the following conditions (4.1), (4.2) and (4.3) are not satisfied, then we replace H_{k-m} with $\frac{y_k^\top y_k}{s_k^\top y_k} I$.

$$\alpha_{\min} \leq \alpha \leq \alpha_{\max} \tag{4.1}$$

$$c_2 \frac{s_k^\top y_k}{y_k^\top y_k} \|\nabla f(x_k)\| \leq \|d_k\| \leq c_1 \frac{s_k^\top y_k}{y_k^\top y_k} \|\nabla f(x_k)\| \tag{4.2}$$

$$\frac{-d_k^\top \nabla f(x_k)}{\|d_k\| \|\nabla f(x_k)\|} > \delta \tag{4.3}$$

We explain *the restart conditions* (4.1), (4.2) and (4.3). The condition (4.3) guarantees that an angle between the search direction and the gradient of the objective function is always larger than some constant δ. If this angle is sufficiently large, the global convergence is guaranteed. The conditions (4.1) and (4.2) guarantees that the search direction d_k is scaled well. In general, when the search direction is appropriate for the Newton-type methods, the stepsize α_k must be 1. For this reason, if the condition (4.1) is not satisfied with $\alpha_{\min}, \alpha_{\max} \approx 1$, d_k would be bad. Additionally, even if the stepsize is 1, the convergence is slow when $\|d_k\|$ is very small. The condition (4.2) guarantees that $\|d_k\|$ is within the appropriate magnitude. Since, $\|\nabla f(x_k)\|$ depends on the problem scale, we compare $\|d_k\|$ with a product of $\|\nabla f(x_k)\|$ and $s_k^\top y_k / y_k^\top y_k$. Note that this is equivalent to compare $\|d_k\|$ with a search direction $-s_k^\top y_k / y_k^\top y_k \nabla f(x_k)$ of the Barzilai-Borwein (BB) method [5]. Consequently, the global convergence of the proposed method with the restart is guaranteed under the same conditions as the BB method. Moreover, if conditions (4.1), (4.2) are satisfied, the Tri-MCQN method is adopted, so the faster convergence than the BB method and LBFGS method is expected.

We present the concrete update algorithm consisting of these three ideas below. Note that the proposed update corresponds to **Step 4** in the quasi-Newton algorithm in Section 2.

The proposed update

Let α_k and d_k be the stepsize and the search direction at the k-th iteration, respectively. Let p be a counter related to the restart.

Step 4-1: If $k = 0$ or the restart condition does not hold, then set $p = 0$, $H^{\mathrm{Tri}}_{k-m} = \frac{s_0^\top y_0}{y_0^\top y_0} I$.

Step 4-2: Set the $(k - m)$-th approximate matrix as follows.

- If $p > m$, set $H_{k-m} = H^{\mathrm{Tri}}_{k-m}$.
- If $k > m, p \leq m$, set $H_{k-m} = \frac{s_k^\top y_k}{y_k^\top y_k} I$.
- If $k \leq m, p \leq m$, set $H_0 = \frac{s_k^\top y_k}{y_k^\top y_k} I$.

Step 4-3: Obtain H_{k+1} from H_{k-m} by the BFGS update as follows.

- If $k > m$ and $p > m$, obtain H_{k+1} by H_{k-m} and $(s_i, y_i), i = k - m, \ldots, k$.
- If $k \leq m$ and $p \leq m$, obtain H_{k+1} by H_0 and $(s_i, y_i), i = 1, \ldots, k$.
- If $k > m$ and $p \leq m$, obtain H_{k+1} by H_{k-m} and $(s_i, y_i), i = 1, \ldots, p$.

Step 4-4: If $p \geq m$, obtain H^{Tri}_{k-m+1} by the MCQN update with H^{Tri}_{k-m} and s_{k-m}, y_{k-m}. Set $p = p + 1$.

We explain this update algorithm. **Step 4-1** corresponds to the idea (3). If *the restart condition* is not satisfied, i.e., any (4.1), (4.2) and (4.3) are not satisfied, then we set $p = 0$ and perform a restart, that is, we discard the current matrix H_{k-m} and set an initial matrix as $\frac{s_k^\top y_k}{y_k^\top y_k} I$. **Steps 4-2** and **4-3** correspond to the idea (2). The Tri-MCQN update is used until $(k - m)$-th iterate, and the BFGS update is used from $(k - m + 1)$-th to k-th iterate to construct the k-th approximate Hessian H_k. **Step 4-4** corresponds to the idea (1). The updates of Hessian by the MCQN are limited to only the tridiagonal elements.

5 Numerical experiments

In this section, numerical results are reported for the proposed method, as well as for the LBFGS method. All algorithms were coded in MATLAB 7.4, and run on a machine with 3.2GHz Pentium 4 CPU and 3.2GB memory. In each experiment, benchmark problems were chosen from CUTEr [8]. In the Tri-MCQN method, the partial matrix $\overline{H}_{ij}, \forall (i, j) \in F$ is updated by the BFGS formula. We employ $\|\nabla f(x_k)\| < n10^{-9}$ as the termination criterion. If the number of outer iterations exceeds 50000, then we terminated all methods as failing. The Wolfe conditions (2.2), (2.3) are used to obtain the stepsize. Note that CUTEr contains problems for which the LBFGS method can obtain a solution within the small number of iteration. The proposed method is ineffective for such problems. Therefore, in all the experiments, the LBFGS method has been executed until the first 20 iterations, and after that, the proposed method starts. Moreover, the LBFGS method stores 5 vector pairs. The Numerical results are shown in Tables 2-6. In the tables, ”n” means the dimension number. ”nf” means the value of $\|\nabla f\|$ at the last iteration. ”ite” means the number of the iteration. ”eval” means the number of the function evaluation.

We will compare algorithms by using the distribution function proposed in [3]. We denote a set of solvers as S, and a set of problems that can be solved by all methods in S as P_S. We also denote a measure for evaluation required to solve a problem p by a solver s as $t_{p,s}$, and the best $t_{p,s}$ for each p as

t_p^*, i.e., $t_p^* := min\left\{t_{p,s} \mid s \in S\right\}$. The distribution function $F_s^S(\tau)$ for a method s is defined by

$$F_s^S(\tau) = \frac{\left|\left\{p \in P_S \mid t_{p,s} \le \tau t_p^*\right\}\right|}{|P_S|},\ \tau \ge 1. \tag{5.1}$$

The algorithm whose $F_s^S(\tau)$ is close to 1 is considered to be superior to the other algorithms in S. In addition, the algorithm whose $F_s^S(\tau)$ reach to 1 rapidly with increased value of τ is considered to be more stable than the other algorithms. We regard $t_{p,s}$ as the number of iterations and the number of the function evaluations. We call the graph whose vertical and abscissa axes are $F_s^S(\tau)$ and τ, respectively, as performance profile.

5.1 Experiments for the parameters in the restart condition

We investigate behaviors of the proposed method by varying the parameter $\alpha_{\min}$, $\alpha_{\max}$, c_1, c_2 in *the restart condition*. We set $\delta = 10^{-8}$ and examine 16 combinations of parameters listed in Table 1. The "p 0" is benchmark in Table 1. We vary the each parameter on the basis of the "p 0". The results are given in Figures 3 - 18. In the figures, the abscissa axis represents the rate of the Tri-MCQN update over iterations, that is,

$$r_1 = \frac{\text{The number of the Tri-MCQN update}}{\text{The number of the total iterations} - 20}.$$

Note that we subtract 20 from the denominator, because we have used the LBFGS method for the first iterations. The vertical axis represents how fast the proposed method is. We use

$$r_2 = \frac{\text{The number of iterates of the proposed method}}{\text{The number of iterates of the LBFGS method}}.$$

as the measure. The each point in the figure denotes the results for each problem. Note that the results of the problems solved within 20 iterations are plotted at (0,1). From graphs, we can observe the effectiveness of the each parameter combinations. When the higher Tri-MCQN rate implies that the proposed method is faster than the LBFGS method, *the restart condition* is effective.

Overall, if the parameters are chosen appropriately, *the restart condition* appears effective. First, we investigate $\alpha_{\min}$. If $\alpha_{\min}$ is small, the results are not good from Figures 4-6. The adoption of the small stepsize may mean the inappropriate construction of the approximate Hessian. Second, the results are rarely different among some varieties of $\alpha_{\max}$. Third, when $c_1 = 1.5$ or 2, *the restart condition* has been rarely satisfied. Therefore, we cannot regard this parameter as the best candidate. In the case of $c_1 = 5$ or 10, although the restart condition is sometimes satisfied, the results are not good. This means that even if $\|d_k\|$ generated by the MCQN update is much larger than $\frac{s_k^\top y_k}{y_k^\top y_k}\|\nabla f(x_k)\|$, d_k is not bad. Finally, we observe c_2. If c_2 is small, the results are not good compared to $c_2 = 1$. This means that even if the stepsize is 1, the proposed method converges slowly for the case $\|d_k\|$ is small. Next, we choose the characteristic combinations of parameters from Table 1, and observe the performance profile in Figures 19 and 20. From this figure, we can see that if *the restart condition* is not used, the convergence becomes slow depending on a problem. Consequently, we conclude that the parameter $\alpha_{\min} = 1$, $\alpha_{\max} = \infty$, $c_1 = \infty$, $c_2 = 0.7$ are the best, and we conduct the next experiments with this parameter.

Table 1: The list of parameters

	$\alpha_{\min}$	$\alpha_{\max}$	c_1	c_2
p 0	1	∞	∞	1
p 1	0	∞	∞	1
p 2	0.5	∞	∞	1
p 3	0.7	∞	∞	1
p 4	1	1	∞	1
p 5	1	2	∞	1
p 6	1	5	∞	1
p 7	1	10	∞	1
p 8	1	∞	1.5	1
p 9	1	∞	2	1
p 10	1	∞	5	1
p 11	1	∞	10	1
p 12	1	∞	∞	0
p 13	1	∞	∞	0.1
p 14	1	∞	∞	0.5
p 15	1	∞	∞	0.7

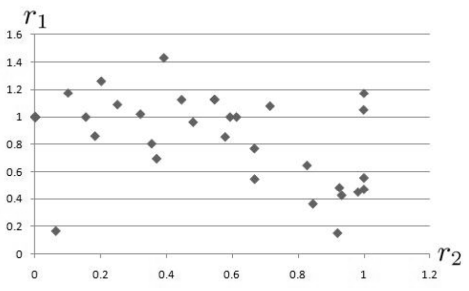


Figure 3: p 0, $(\alpha_{\min}, \alpha_{\max}, c_1, c_2) = (1, \infty, \infty, 1)$

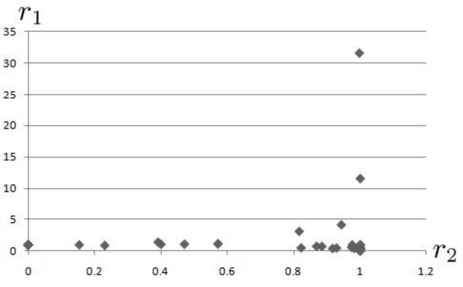


Figure 4: p 1, $(\alpha_{\min}, \alpha_{\max}, c_1, c_2) = (0, \infty, \infty, 1)$

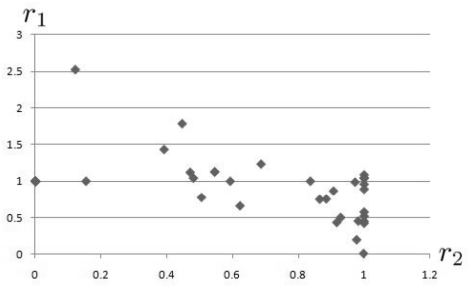


Figure 5: p 2, $(\alpha_{\min}, \alpha_{\max}, c_1, c_2) = (0.5, \infty, \infty, 1)$

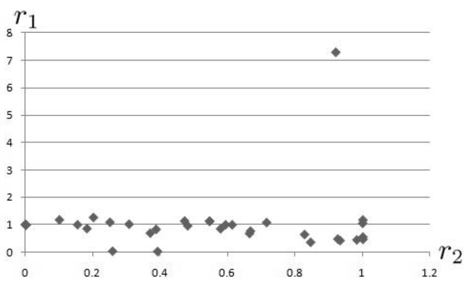


Figure 6: p 3, $(\alpha_{\min}, \alpha_{\max}, c_1, c_2) = (0.7, \infty, \infty, 1)$

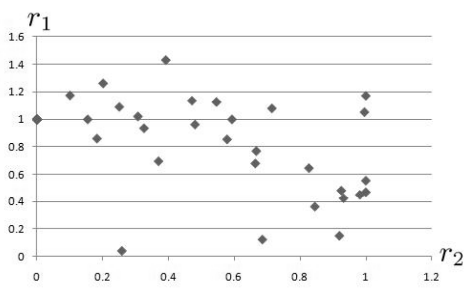


Figure 7: p 4, $(\alpha_{\min}, \alpha_{\max}, c_1, c_2) = (1, 1, \infty, 1)$

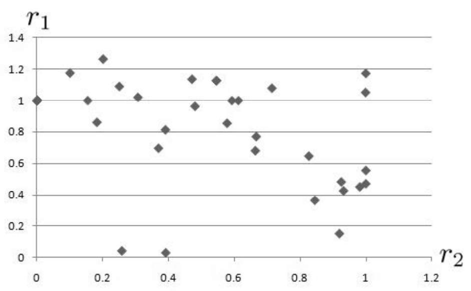


Figure 8: p 5, $(\alpha_{\min}, \alpha_{\max}, c_1, c_2) = (1, 2, \infty, 1)$

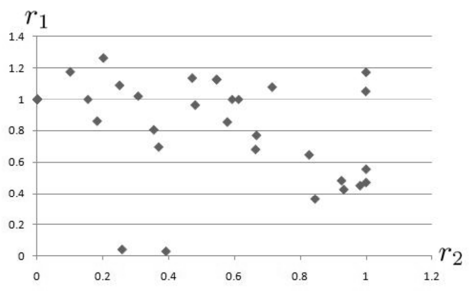


Figure 9: p 6, $(\alpha_{\min}, \alpha_{\max}, c_1, c_2) = (1, 5, \infty, 1)$

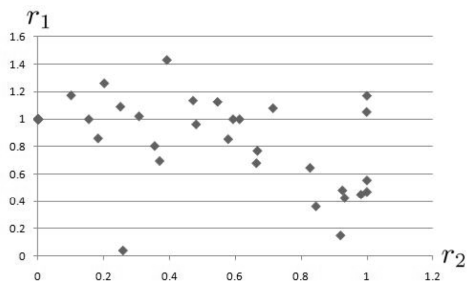


Figure 10: p 7, $(\alpha_{\min}, \alpha_{\max}, c_1, c_2) = (1, 10, \infty, 1)$

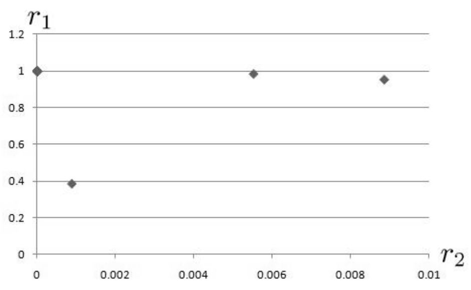


Figure 11: p 8, $(\alpha_{\min}, \alpha_{\max}, c_1, c_2) = (1, \infty, 1.5, 1)$

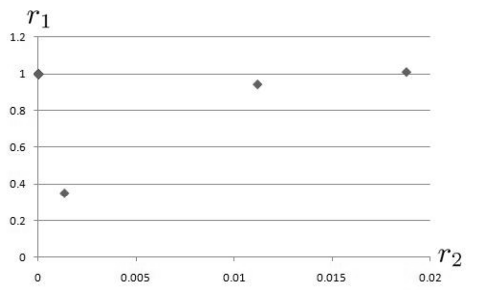


Figure 12: p 9, $(\alpha_{\min}, \alpha_{\max}, c_1, c_2) = (1, \infty, 2, 1)$

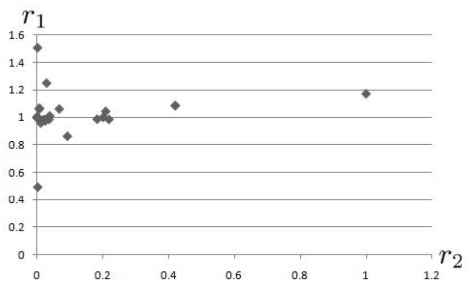


Figure 13: p 10, $(\alpha_{\min}, \alpha_{\max}, c_1, c_2) = (1, \infty, 5, 1)$

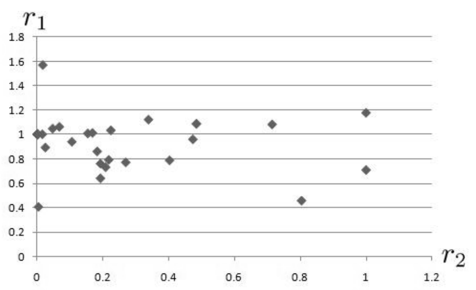


Figure 14: p 11, $(\alpha_{\min}, \alpha_{\max}, c_1, c_2) = (1, \infty, 10, 1)$

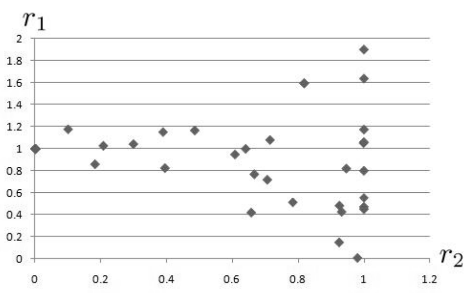


Figure 15: p 12, $(\alpha_{\min}, \alpha_{\max}, c_1, c_2) = (1, \infty, \infty, 0)$

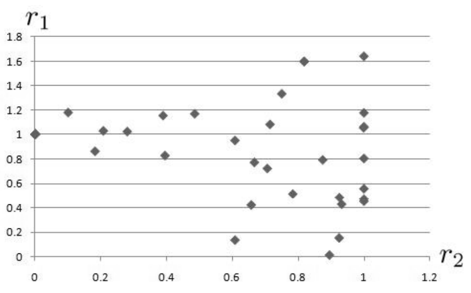


Figure 16: p 13,$(\alpha_{\min}, \alpha_{\max}, c_1, c_2) = (1, \infty, \infty, 0.1)$

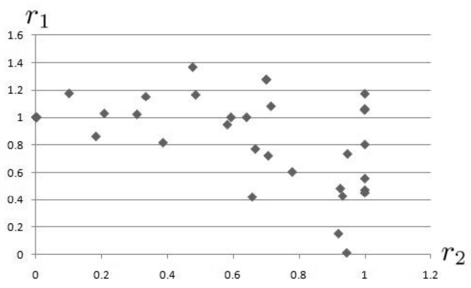


Figure 17: p 14, $(\alpha_{\min}, \alpha_{\max}, c_1, c_2) = (1, \infty, \infty, 0.5)$

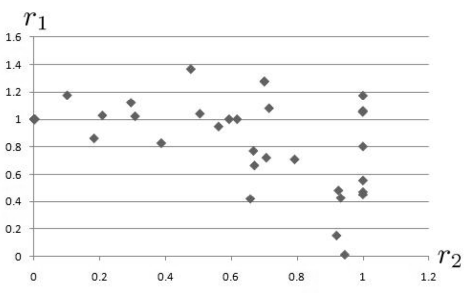


Figure 18: p 15, $(\alpha_{\min}, \alpha_{\max}, c_1, c_2) = (1, \infty, \infty, 0.7)$

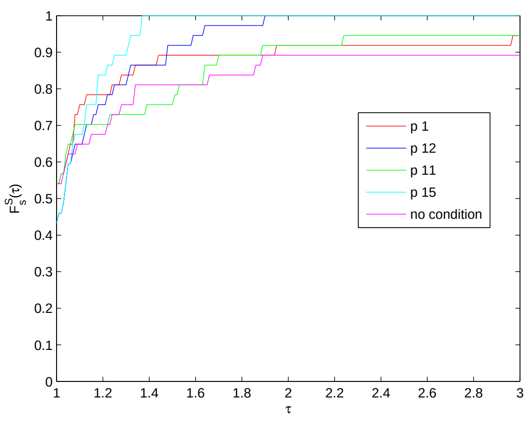


Figure 19: The performance profile of the iteration for the parameters in the restart condition

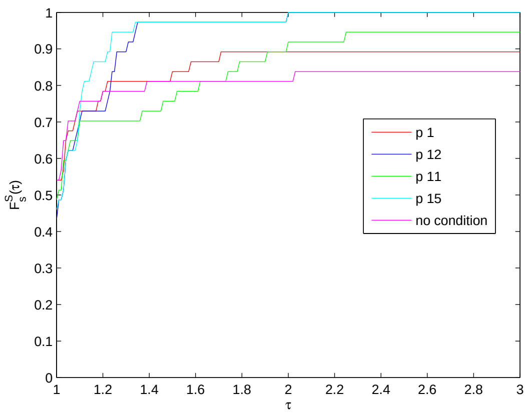


Figure 20: The performance profile of the function evaluation for the parameters in the restart condition

5.2 Experiment for the parameter m

The results of the proposed method with several $m = 0, 2, 5, 10$ are shown. The distribution function (5.1) for the numerical results is shown in Figures 21 and 22. $m = 0$ means that the only Tri-MCQN update is used from 20th iterate to termination. In the case of $m = 0$, if the approximate Hessian is not good, the convergence may become slow. In fact, even if τ becomes larger than 3, $m = 0$ doesn't reach top. The case of $m = 5$ seems to be appropriate for the benchmark problems. From Figures 21 and 22, we see that the case of $m = 2$ is unstable. This is because this case exploits the small number of the stored vector pairs to construct the approximate matrix. This means the lack of the information.

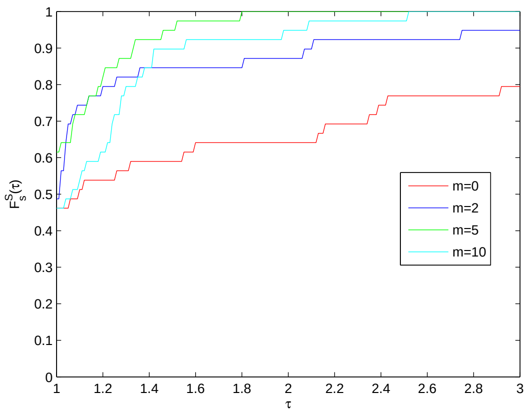


Figure 21: The performance profile of the number of iterations for the parameter m

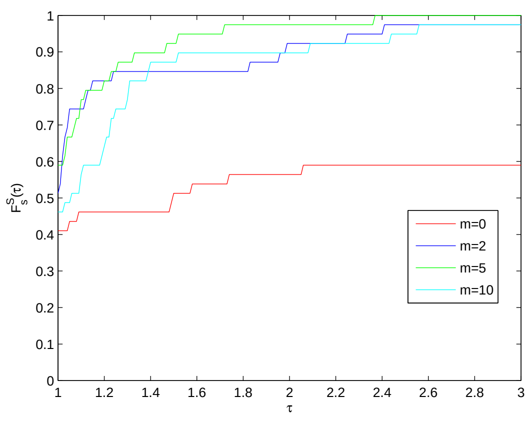


Figure 22: The performance profile of the number of function evaluations for the parameter *m*

5.3 Experiment comparing the proposed method with the LBFGS method

In this section, the proposed method is compared with the LBFGS method by using (5.1). We adopt p 15 as the parameter of *the restart condition*. The result is shown in Figures 23 and 24. Despite the value of τ, the proposed method is always upper than the LBFGS method and reaches top rapidly. Therefore, even if this method is inferior to the LBFGS method for some problem, the degree of loss is within 1.4 times roughly. Furthermore, if the proposed method solves the problems for which the LBFGS method requires a lot of iterates, the optimal solution is obtained faster. In the Table 2, the LBFGS requires a lot of iterates for TRIDIA. TRIDIA is a problem defined as follows [8].

$$f(x) = (x_1 - 1)^2 + \sum_{i=2}^{n} i(x_{i-1} - 2x_i)^2, \ x^0 = (1, \ldots, 1)^\top.$$

It is known that TRIDIA is ill-conditioned, and the LBFGS requires lots of iterates to solve it. From this experiment, it is shown that the proposed method is effective for such problems.

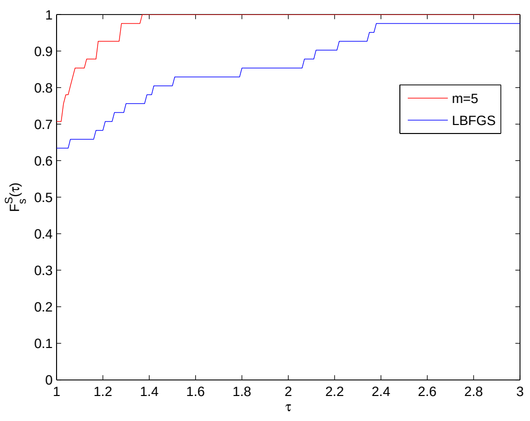


Figure 23: The performance profile between the LBFGS and the proposed method with the number of iterations

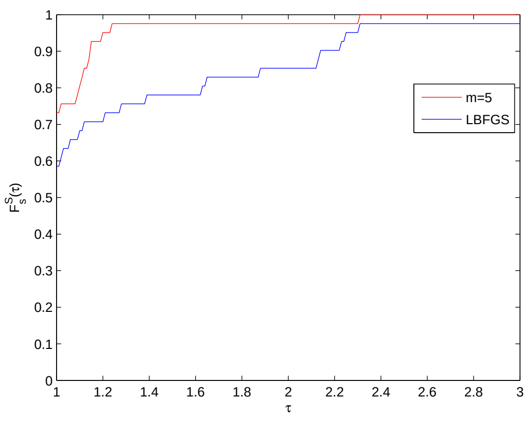


Figure 24: The performance profile between the LBFGS and the proposed method with the number of the function evaluations

6 Concluding remarks

In this paper, we proposed the method which combines the tridiagonal MCQN method and the LBFGS method. The numerical results suggest that the proposed method is very promising.

Only unconstrained minimization problems were considered in the paper. The extension of the LBFGS method to problems with bound constrained is proposed in [2]. Using the similar idea, we may extend the proposed method to such problems.

7 Acknowledgements

I would like to express my deep appreciation to Associate Professor Nobuo Yamashita for his continual guidance and invaluable suggestions to accomplish this paper. He carefully read this paper and commented in detail on the whole work. Without his considerable help, none of the work could be completed. I am deeply indebted to Professor Masao Fukushima who has given me careful advice and guidance. I am also grateful to Assistant Professor Shunsuke Hayashi for his significant advice. I would like to thank all members in Professor Fukushima's Laboratory for their warm friendship during the course.

References

[1] Blair, J.R.S., Peyton, B.: An introduction to chordal graphs and clique trees. In: George, A., Gilbert, J.R., Liu, J.W.H. (eds.), Graph Theory and Sparse Matrix Computation, pp. 1–29, Springer, NewYork (1993).

[2] Byrd, H, R., Lu, P., Nocedal, J., Zhu, C.: A limited memory algorithm for bound constrained optimization, ACM transactions on mathematical software, **23**(4), 550–560 (1997).

[3] Dolan, D.E., More, J.J.: Benchmarking optimization software with performance profiles, Mathematical Programming, **91**, pp. 201–213 (2002).

[4] Fletcher, R.: A new approach to variable metric algorithms, Computer Journal, **13**(3), 317–322 (1970).

[5] Fletcher, R.: On the Barzilai-Borwein method, Applied Optimization, **96**(II), 235–256 (2005).

[6] Fletcher, R., Powell, M. J. D.: A rapidly convergent descent method for minimization, Computer Journal, **6**, pp. 163–168 (1963).

[7] Fukuda, M., Kojima, M., Murota, K., Nakata, K.: Exploiting sparsity in semidefinite programming via matrix completion I: general framework, SIAM Journal on Optimization, **11**(3), 647–674 (2000).

[8] Gould, N.I.M., Orban, D., Toint, Ph.L.: CUTEr, a constrained and unconstrained testing environment: revisited, ACM transactions on mathematical software, **29**, 373–394 (2003).

[9] LIU, D.C., Nocedal, J.: On the limited memory BFGS method for large scale optimization, Mathematical Programming, **45**, 503–528 (1989).

[10] Nocedal, J.: Updating quasi-Newton matrices with limited storage, Mathematics of Computation **35**, 773–782 (1980).

[11] Nocedal, J., Wright, S.J.: Numerical optimization, Springer, New York (1999).

[12] Yamashita, N.: Sparce quasi-newton updates with positive difinite matrix completion, Mathematical Programming, Ser. A **115**, 1–30 (2008).

[13] Zhang, H., William, W.H.: A nonmonotone line search technique and its application to unconstrained optimization, SIAM Journal on Optimization, **14**(4), 1043–1056 (2004).

Table 2: The LBFGS method

Problem	n	method	nf	ite	eval	time
ARWHEAD	5000	LBFGS	2.01E-05	9	28	5.000E-02
BROYDN7D	5000	LBFGS	4.29E-05	7080	7609	7.937E+01
BRYBND	5000	LBFGS	4.61E-05	23	42	1.200E-01
CHAINWOO	4000	LBFGS	3.34E-05	388	547	1.190E+00
COSINE	10000	LBFGS	2.06E-05	9	21	9.000E-02
CRAGGLVY	5000	LBFGS	4.92E-05	76	115	4.300E-01
DIXMAANA	3000	LBFGS	2.17E-06	9	14	2.000E-02
DIXMAANB	3000	LBFGS	2.02E-06	9	14	2.000E-02
DIXMAANC	3000	LBFGS	1.51E-05	9	15	2.000E-02
DIXMAAND	3000	LBFGS	1.06E-05	9	17	2.000E-02
DIXMAANE	3000	LBFGS	2.10E-05	180	198	3.300E-01
DIXMAANF	3000	LBFGS	2.21E-05	159	173	2.800E-01
DIXMAANG	3000	LBFGS	2.99E-05	185	205	3.300E-01
DIXMAANH	3000	LBFGS	2.41E-05	122	139	2.500E-01
DIXMAANI	3000	LBFGS	2.88E-05	277	298	5.700E-01
DIXMAANJ	3000	LBFGS	2.42E-05	214	229	3.500E-01
DIXMAANL	3000	LBFGS	2.96E-05	166	191	3.300E-01
DIXON3DQ	10000	LBFGS	9.15E-05	9627	10338	3.024E+01
DQDRTIC	5000	LBFGS	1.69E-05	18	34	6.000E-02
DQRTIC	5000	LBFGS	3.86E-05	47	97	1.200E-01
EDENSCH	2000	LBFGS	1.06E-05	25	41	6.000E-02
EG2	1000	LBFGS	7.29E-07	4	14	1.000E-02
ENGVAL1	5000	LBFGS	9.59E-06	25	39	8.000E-02
EXTROSNB	1000	LBFGS	9.56E-06	33	55	5.000E-02
FLETCHCR	1000	LBFGS	3.68E-06	61	83	8.000E-02
FMINSRF2	5625	LBFGS	5.60E-05	277	298	9.000E-01
GENHUMPS	5000	LBFGS	2.47E-05	6837	11473	4.102E+01
LIARWHD	5000	LBFGS	3.77E-07	17	40	7.000E-02
MOREBV	5000	LBFGS	4.92E-05	15	22	4.000E-02
NONCVXU2	5000	LBFGS	4.60E-05	9650	10343	3.013E+01
NONDIA	5000	LBFGS	2.58E-06	34	69	1.500E-01
NONDQUAR	5000	LBFGS	4.42E-05	850	1120	1.670E+00
PENALTY1	1000	LBFGS	6.98E-06	93	164	9.000E-02
POWELLSG	5000	LBFGS	4.41E-06	47	76	1.000E-01
QUARTC	5000	LBFGS	3.86E-05	47	97	1.200E-01
SCHMVETT	5000	LBFGS	4.81E-05	36	49	2.400E-01
SPARSINE	5000	LBFGS	8.13E-02	50000	56240	2.956E+02
SPARSQUR	10000	LBFGS	9.96E-05	33	68	3.500E-01
SROSENBR	5000	LBFGS	1.31E-06	16	34	3.000E-02
TESTQUAD	5000	LBFGS	1.07E-04	50000	316815	2.792E+02
TQUARTIC	5000	LBFGS	3.76E-05	19	36	5.000E-02
TRIDIA	5000	LBFGS	4.94E-05	1441	1839	3.070E+00
WOODS	4000	LBFGS	3.01E-05	30	60	8.000E-02

Table 3: The proposed method ($m = 0$)

Problem	n	nf	ite	eval	time	LBFGS	Tri-MCQN
ARWHEAD	5000	2.00E-05	9	28	7.40E-01	9	0
BROYDN7D	5000	4.90E-05	4594	26187	1.14E+03	20	4574
BRYBND	5000	4.90E-05	54	209	7.26E+00	20	34
CHAINWOO	4000	3.80E-05	666	7019	1.08E+02	20	646
COSINE	10000	2.10E-05	9	21	1.48E+00	9	0
CRAGGLVY	5000	1.55E-04	50000	2245301	1.77E+04	20	49980
DIXMAANA	3000	2.00E-06	9	14	4.40E-01	9	0
DIXMAANB	3000	2.00E-06	9	14	4.50E-01	9	0
DIXMAANC	3000	1.50E-05	9	15	4.30E-01	9	0
DIXMAAND	3000	1.10E-05	9	17	4.40E-01	9	0
DIXMAANE	3000	1.50E-05	69	74	7.16E+00	20	49
DIXMAANF	3000	2.50E-05	68	74	5.46E+00	20	48
DIXMAANG	3000	3.00E-05	70	80	5.65E+00	20	50
DIXMAANH	3000	2.70E-05	58	66	4.53E+00	20	38
DIXMAANI	3000	3.00E-05	184	189	2.18E+01	20	164
DIXMAANJ	3000	2.80E-05	116	124	1.24E+01	20	96
DIXMAANL	3000	3.00E-05	79	90	8.64E+00	20	59
DIXON3DQ	10000	2.80E-05	10011	15010	3.53E+03	20	9991
DQDRTIC	5000	1.70E-05	18	34	1.45E+00	18	0
DQRTIC	5000	5.00E-05	288	405	5.74E+01	20	268
EDENSCH	2000	1.80E-05	28	71	1.18E+00	20	8
EG2	1000	1.00E-06	4	14	8.00E-02	4	0
ENGVAL1	5000	4.80E-05	33	172	4.70E+00	20	13
EXTROSNB	1000	9.00E-06	71	555	2.83E+00	20	51
FLETCHCR	1000	7.00E-06	100	843	3.14E+00	20	80
FMINSRF2	5625	5.60E-05	7078	14250	1.51E+03	20	7058
GENHUMPS	5000	2.08E+01	50000	844598	1.27E+04	20	49980
LIARWHD	5000	0.00E+00	17	40	1.38E+00	17	0
MOREBV	5000	4.90E-05	15	22	1.20E+00	15	0
NONCVXU2	5000	9.08E-02	50000	257934	9.84E+03	20	49980
NONDIA	5000	4.10E-05	7029	190694	1.76E+03	20	7009
NONDQUAR	5000	4.90E-05	315	397	4.82E+01	20	295
PENALTY1	1000	1.00E-05	2808	8314	1.05E+02	20	2788
POWELLSG	5000	4.80E-05	4275	22006	8.09E+02	20	4255
QUARTC	5000	5.00E-05	288	405	5.68E+01	20	268
SCHMVETT	5000	3.80E-05	39	136	5.05E+00	20	19
SPARSINE	5000	1.45E+00	50000	807685	1.28E+04	20	49980
SPARSQUR	10000	9.80E-05	108	150	4.08E+01	20	88
SROSENBR	5000	1.00E-06	16	34	1.28E+00	16	0
TESTQUAD	5000	4.90E-05	9711	166893	1.96E+03	20	9691
TQUARTIC	5000	3.80E-05	19	36	1.52E+00	19	0
TRIDIA	5000	4.00E-05	491	7.38E+03	5.54E+01	20	471
WOODS	4000	3.80E-05	1477	13389	2.49E+02	20	1457

Table 4: The proposed method ($m = 2$)

Problem	n	nf	ite	eval	time	LBFGS	Tri-MCQN
ARWHEAD	5000	2.00E-05	9	28	5.10E-01	9	0
BROYDN7D	5000	4.20E-05	6227	7195	9.53E+02	1389	3409
BRYBND	5000	2.10E-05	25	44	2.11E+00	20	5
CHAINWOO	4000	3.50E-05	288	348	3.65E+01	48	240
COSINE	10000	2.10E-05	9	21	5.60E-01	9	0
CRAGGLVY	5000	NaN	62	6400	FFFFF	27	35
DIXMAANA	3000	2.00E-06	9	14	3.00E-01	9	0
DIXMAANB	3000	2.00E-06	9	14	2.00E-01	9	0
DIXMAANC	3000	1.50E-05	9	15	2.60E-01	9	0
DIXMAAND	3000	1.10E-05	9	17	3.50E-01	9	0
DIXMAANE	3000	2.70E-05	78	83	8.15E+00	20	58
DIXMAANF	3000	2.00E-05	62	68	6.31E+00	20	42
DIXMAANG	3000	2.70E-05	88	99	8.62E+00	25	63
DIXMAANH	3000	2.50E-05	59	67	4.34E+00	20	39
DIXMAANI	3000	3.00E-05	115	120	9.77E+00	20	95
DIXMAANJ	3000	2.60E-05	75	83	5.90E+00	21	54
DIXMAANL	3000	2.00E-05	76	88	5.86E+00	23	53
DIXON3DQ	10000	2.80E-05	10189	10194	3.63E+03	20	10169
DQDRTIC	5000	1.70E-05	18	34	1.07E+00	18	0
DQRTIC	5000	4.90E-05	62	113	6.56E+00	29	33
EDENSCH	2000	8.00E-06	26	42	8.10E-01	22	4
EG2	1000	1.00E-06	4	14	4.00E-02	4	0
ENGVAL1	5000	8.00E-06	25	38	1.55E+00	20	5
EXTROSNB	1000	1.00E-05	33	54	8.00E-01	21	12
FLETCHCR	1000	8.00E-06	50	72	1.24E+00	24	26
FMINSRF2	5625	5.20E-05	606	642	1.25E+02	80	526
GENHUMPS	5000	2.10E-05	8694	11873	1.33E+03	2313	6381
LIARWHD	5000	0.00E+00	17	40	1.08E+00	17	0
MOREBV	5000	4.90E-05	15	22	8.70E-01	15	0
NONCVXU2	5000	5.76E-04	50000	52123	9.12E+03	3244	46756
NONDIA	5000	4.70E-05	243	1100	3.51E+01	87	156
NONDQUAR	5000	4.80E-05	651	726	1.16E+02	92	559
PENALTY1	1000	8.00E-06	99	207	1.66E+00	58	41
POWELLSG	5000	4.20E-05	129	162	1.58E+01	35	94
QUARTC	5000	4.90E-05	62	113	8.74E+00	29	33
SCHMVETT	5000	3.60E-05	37	47	3.64E+00	22	15
SPARSINE	5000	4.92E-01	50000	51111	9.31E+03	2150	47850
SPARSQUR	10000	7.90E-05	37	73	6.26E+00	24	13
SROSENBR	5000	1.00E-06	16	34	8.20E-01	16	0
TESTQUAD	5000	4.90E-05	836	988	1.45E+02	50	786
TQUARTIC	5000	3.80E-05	19	36	1.42E+00	19	0
TRIDIA	5000	4.30E-05	206	239	3.92E+01	24	182
WOODS	4000	3.00E-05	113	147	1.41E+01	37	76

Table 5: The proposed method ($m = 5$)

Problem	n	nf	ite	eval	time	LBFGS	Tri-MCQN
ARWHEAD	5000	2.00E-05	9	28	3.00E-01	9	0
BROYDN7D	5000	4.50E-05	6704	7443	8.47E+02	2957	3747
BRYBND	5000	4.90E-05	27	46	2.10E+00	20	7
CHAINWOO	4000	3.80E-05	275	337	3.73E+01	73	202
COSINE	10000	2.10E-05	9	21	1.00E-01	9	0
CRAGGLVY	5000	4.80E-05	61	90	7.38E+00	20	41
DIXMAANA	3000	2.00E-06	9	14	1.80E-01	9	0
DIXMAANB	3000	2.00E-06	9	14	7.00E-02	9	0
DIXMAANC	3000	1.50E-05	9	15	9.00E-02	9	0
DIXMAAND	3000	1.10E-05	9	17	1.80E-01	9	0
DIXMAANE	3000	2.00E-05	87	93	6.89E+00	25	62
DIXMAANF	3000	2.90E-05	75	81	6.90E+00	20	55
DIXMAANG	3000	2.90E-05	79	89	6.33E+00	24	55
DIXMAANH	3000	2.50E-05	88	100	5.32E+00	40	48
DIXMAANI	3000	2.80E-05	154	159	1.81E+01	20	134
DIXMAANJ	3000	2.30E-05	90	102	5.45E+00	44	46
DIXMAANL	3000	2.60E-05	75	86	8.46E+00	20	55
DIXON3DQ	10000	9.40E-05	10133	10140	4.07E+03	29	10104
DQDRTIC	5000	1.70E-05	18	34	6.60E-01	18	0
DQRTIC	5000	4.80E-05	60	111	7.04E+00	32	28
EDENSCH	2000	1.10E-05	25	41	4.00E-01	25	0
EG2	1000	1.00E-06	4	14	1.00E-02	4	0
ENGVAL1	5000	4.50E-05	27	90	1.25E+00	22	5
EXTROSNB	1000	8.00E-06	35	56	7.40E-01	20	15
FLETCHCR	1000	9.00E-06	47	69	8.30E-01	29	18
FMINSRF2	5625	5.60E-05	288	323	3.18E+01	153	135
GENHUMPS	5000	2.70E-05	5653	10498	5.01E+02	3477	2176
LIARWHD	5000	0.00E+00	17	40	9.10E-01	17	0
MOREBV	5000	4.90E-05	15	22	6.30E-01	15	0
NONCVXU2	5000	4.50E-05	9916	12348	4.80E+02	7866	2050
NONDIA	5000	5.00E-05	40	85	1.19E+00	38	2
NONDQUAR	5000	4.90E-05	565	680	7.09E+01	200	365
PENALTY1	1000	2.00E-06	95	186	1.48E+00	72	23
POWELLSG	5000	2.50E-05	47	72	3.47E+00	31	16
QUARTC	5000	4.80E-05	60	111	5.96E+00	32	28
SCHMVETT	5000	3.60E-05	31	44	1.35E+00	29	2
SPARSINE	5000	8.98E-02	50000	55103	6.54E+03	19163	30837
SPARSQUR	10000	3.80E-05	37	75	2.74E+00	32	5
SROSENBR	5000	1.00E-06	16	34	5.50E-01	16	0
TESTQUAD	5000	3.20E-05	616	742	1.28E+02	53	563
TQUARTIC	5000	3.80E-05	19	36	6.50E-01	19	0
TRIDIA	5000	4.90E-05	219	255	2.94E+01	36	183
WOODS	4000	1.80E-05	41	67	2.04E+00	31	10

Table 6: The proposed method ($m = 10$)

Problem	n	nf	ite	eval	time	LBFGS	Tri-MCQN
ARWHEAD	5000	2.00E-05	9	28	5.00E-02	9	0
BROYDN7D	5000	4.90E-05	9577	18370	1.89E+02	9538	39
BRYBND	5000	4.20E-05	23	42	3.70E-01	23	0
CHAINWOO	4000	4.00E-05	343	435	2.62E+01	194	149
COSINE	10000	2.10E-05	9	21	9.00E-02	9	0
CRAGGLVY	5000	4.60E-05	68	101	3.07E+00	52	16
DIXMAANA	3000	2.00E-06	9	14	2.00E-02	9	0
DIXMAANB	3000	2.00E-06	9	14	2.00E-02	9	0
DIXMAANC	3000	1.50E-05	9	15	2.00E-02	9	0
DIXMAAND	3000	1.10E-05	9	17	3.00E-02	9	0
DIXMAANE	3000	2.50E-05	95	103	5.86E+00	44	51
DIXMAANF	3000	2.40E-05	79	85	7.24E+00	25	54
DIXMAANG	3000	2.70E-05	94	111	4.10E+00	64	30
DIXMAANH	3000	2.10E-05	90	100	5.22E+00	37	53
DIXMAANI	3000	2.90E-05	127	132	1.01E+01	24	103
DIXMAANJ	3000	2.90E-05	97	108	5.56E+00	44	53
DIXMAANL	3000	2.90E-05	75	86	7.62E+00	20	55
DIXON3DQ	10000	9.80E-05	10386	10398	3.73E+03	57	10329
DQDRTIC	5000	1.70E-05	18	34	7.00E-02	18	0
DQRTIC	5000	3.80E-05	85	136	1.69E+00	80	5
EDENSCH	2000	8.00E-06	25	41	1.90E-01	25	0
EG2	1000	1.00E-06	4	14	1.00E-02	4	0
ENGVAL1	5000	1.90E-05	31	115	2.10E-01	31	0
EXTROSNB	1000	9.00E-06	35	57	2.50E-01	30	5
FLETCHCR	1000	9.00E-06	58	90	5.30E-01	48	10
FMINSRF2	5625	5.40E-05	351	387	2.22E+01	233	118
GENHUMPS	5000	4.60E-05	11140	33890	4.89E+02	9412	1728
LIARWHD	5000	0.00E+00	17	40	4.70E-01	17	0
MOREBV	5000	4.90E-05	15	22	2.10E-01	15	0
NONCVXU2	5000	4.20E-05	13946	15994	4.17E+02	12095	1851
NONDIA	5000	2.00E-06	51	101	4.50E-01	51	0
NONDQUAR	5000	4.30E-05	623	826	2.93E+01	449	174
PENALTY1	1000	2.00E-06	239	452	7.00E-01	235	4
POWELLSG	5000	5.00E-06	53	87	3.40E-01	52	1
QUARTC	5000	3.80E-05	85	136	1.69E+00	80	5
SCHMVETT	5000	3.60E-05	31	41	2.00E+00	20	11
SPARSINE	5000	5.87E-02	50000	56902	1.84E+03	42196	7804
SPARSQUR	10000	2.40E-05	44	81	5.20E-01	44	0
SROSENBR	5000	1.00E-06	16	34	1.30E-01	16	0
TESTQUAD	5000	4.60E-05	464	506	9.15E+01	26	438
TQUARTIC	5000	3.80E-05	19	36	7.00E-02	19	0
TRIDIA	5000	4.30E-05	225	262	2.77E+01	51	174
WOODS	4000	3.00E-05	36	61	2.70E-01	35	1