Available online at www.sciencedirect.com

ScienceDirect

Computers and Mathematics with Applications 56 (2008) 1001–1009

An International Journal
computers & mathematics with applications

www.elsevier.com/locate/camwa

# A limited memory BFGS-type method for large-scale unconstrained optimization

Yunhai Xiao[a,*], Zengxin Wei[b], Zhiguo Wang[c]

[a] Institute of Applied Mathematics, School of Mathematics and Information Science, Henan University, Kaifeng, 475004, PR China
[b] College of Mathematics and Information Science, Guangxi University, Nanning, 530004, PR China
[c] School of Mathematics and Information Science, Henan University, Kaifeng, 475004, PR China

Received 29 January 2007; received in revised form 15 December 2007; accepted 7 January 2008

**Abstract**

In this paper, a new numerical method for solving large-scale unconstrained optimization problems is presented. It is derived from a modified BFGS-type update formula by Wei, Li, and Qi. It is observed that the update formula can be extended to the framework of limited memory scheme with hardly more storage or arithmetic operations. Under some suitable conditions, the global convergence property is established. The implementations of the method on a set of CUTE problems indicate that this extension is beneficial to the performance of the algorithm.
© 2008 Elsevier Ltd. All rights reserved.

*Keywords:* Unconstrained optimization; Large-scale optimization; Quasi-Newton method; Limited memory BFGS method

## 1. Introduction

In this paper we consider the unconstrained optimization problem

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

where $f : \mathbf{R}^n \to \mathbf{R}$ is a nonlinear function whose gradient at point $x_k$ is $g(x_k)$, or $g_k$, for the sake of simplicity. We assume that $f$ is continuously differentiable and $n$ is large.

One of the most effective methods for solving the unconstrained problem (1.1) is the Newton method. It normally requires the fewest number of function evaluations, and is very good at handling ill-conditioning. However, its efficiency largely depends on the possibility of solving efficiently a linear system which arises when computing the search direction $d_k$ at each iteration

$$G(x_k)d_k = -g(x_k), \tag{1.2}$$

where $G(x_k)$ is the Hessian matrix of $f$ in the current iteration. Moreover, the exact solution of the linear system (1.2) could be too burdensome, or is not necessary when $x_k$ is far from the solution of $f$ (see [1]).

* Corresponding author.
*E-mail addresses:* yunhai816@163.com, yhxiao@henu.edu.cn (Y. Xiao), zxwei@gxu.edu.cn (Z. Wei), wangzg@henu.edu.cn (Z. Wang).

0898-1221/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.camwa.2008.01.028

Inexact Newton methods (see [1,2]) represent the basic approach underlying most of the Newton-type large-scale unconstrained algorithms. At each step, the current estimate of the solution is updated by approximately solving the linear system (1.2) using an iterative algorithm. The inner iteration is typically "truncated" before the solution of the linear system is obtained. The limited memory BFGS (L-BFGS) method (see [3,4]) is an adaptation of the BFGS method for large-scale problems. The implementation is almost identical to that of the standard BFGS method, the only difference is that the inverse Hessian approximation is not formed explicitly, but defined by a small number of BFGS updates. It often provides a fast rate of linear convergence, and requires minimal storage.

Another class of methods which can be successfully adopted to solve large-scale unconstrained optimization problems is the nonlinear conjugate gradient methods (e.g. [5–8]). They are very popular due to their simplicity and low storage requirements. The search direction in all nonlinear conjugate gradient methods is given by $d_k = -g_k + \beta_k d_{k-1}$, with $d_0 = -g_0$ and $\beta_k$ being a scalar. Most of the recent work on nonlinear conjugate gradient methods is focused on the design of a new $\beta_k$ or a new line search strategy. We refer to paper [9] for a good review. The large-scale unconstrained optimization problems have received much attention in recent decades. We refer to [10,11] for a good survey.

Since the standard BFGS method is widely used to solve general minimization problems, most of the studies concerning limited memory methods concentrate on the L-BFGS method. As we know, the BFGS update only exploits the gradient information, while the function values available are neglected. Hence, many efficient attempts have been made to modify the usual quasi-Newton methods using both the gradient and function value information (e.g. [12, 13]). Lately, in order to get a higher order accuracy in approximating the Hessian matrix of objective function, Wei, Li, and Qi (see [14]) proposed a modified BFGS-type method for the solution of (1.1), and the reported numerical results in [14] show that the average performance is better than that of the standard BFGS method. This motivates us to propose a limited memory BFGS-type method on the basis of Wei et al. (see [14]), which is suitable for solving large-scale unconstrained optimization problems. The major contribution of this paper is an extension of the BFGS-type method in [14] to limited memory scheme. Unlike the standard L-BFGS method, a distinguishing feature of our proposed method is that a triple

$$\{s_i, y_i, \lambda_i\}, \quad i = k - \widehat{m} + 1, \ldots, k$$

is stored at each iteration, where $s_i = x_{i+1} - x_i$, $y_i = g_{i+1} - g_i$, $\widehat{m} > 0$, and $\lambda_i$ is a scalar related to function values. Compared with the standard BFGS method, at each iteration, the proposed method requires no more function or derivative evaluations, and hardly more storage or arithmetic operations. Under appropriate conditions, we establish the global convergence of the method. The numerical experiments of the proposed method on a set of large problems indicate that it is promising.

We organize this paper as follows. In the next section, we briefly review the BFGS-type method of Wei et al. In Section 3, we describe the modified limited memory BFGS-type algorithm. We discuss the global convergence and convergence rate with weak Wolfe–Powell line search in Section 4. In the last section we describe the comparative testing for an implementation based on the modified method versus an implementation based on other well-known codes. Throughout the paper, $\|\cdot\|$ denotes the Euclidean norm of vectors.

## 2. Modified BFGS update

Quasi-Newton methods are iterative methods of the form

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

where $\alpha_k$ is a steplength, and $d_k$ is a search direction with the form

$$B_k d_k + g_k = 0, \tag{2.1}$$

where $B_k$ is an approximation of $\nabla^2 f(x_k)$. By tradition, $\{B_k\}$ satisfies the following quasi-Newton equation

$$B_{k+1} s_k = y_k. \tag{2.2}$$

The very famous update $B_k$ is the standard BFGS formula

$$B_{k+1} = B_k - \frac{B_k s_k s_k^{\mathrm{T}} B_k}{s_k^{\mathrm{T}} B_k s_k} + \frac{y_k y_k^{\mathrm{T}}}{s_k^{\mathrm{T}} y_k}. \tag{2.3}$$

It has been shown that the standard BFGS method is the most effective one in quasi-Newton methods from the computation point of view. When $f$ is convex, the global convergence of the BFGS method has been widely studied (e.g. [15–17]). For general function $f$, Dai (see [18]) has constructed an example to show that the standard BFGS method may fail for nonconvex functions with inexact line search. Mascarenhas (see [19]) showed the nonconvergence of the standard BFGS method even with exact line search. Li and Fukushima (see [20,21]) made a slight modification to the standard BFGS method and developed a modified BFGS method and a cautious BFGS method. Under appropriate conditions, these two methods are globally and superlinearly convergent for nonconvex minimization problems. In what follows, we simply describe the work of Li and Fukushima [20].

When solving the unconstrained optimization problem (1.1), in order to obtain a global convergence of BFGS method without convexity assumption on the objective function, Li and Fukushima (see [20]) formulated a new quasi-Newton equation with the following form

$$B_{k+1}s_k = y_k^*, \tag{2.4}$$

where $y_k^* = y_k + t_k\|g_k\|s_k$, with $t_k > 0$ is determined by $t_k = 1 + \max\{-\frac{s_k^{\mathrm{T}}y_k}{\|s_k\|^2}, 0\}$. Then it is easy to see that $s_k^{\mathrm{T}}y_k^* \geq 0$ always holds.

In order to get a better approximation of the objective function Hessian matrix, Wei, Li, and Qi (see [14]) also proposed a similar quasi-Newton equation based on (2.4):

$$B_{k+1}s_k = y_k^* = y_k + \lambda_k s_k, \tag{2.5}$$

where

$$\lambda_k = \frac{2[f(x_k) - f(x_{k+1})] + (g_{k+1} + g_k)^{\mathrm{T}}s_k}{\|s_k\|^2}. \tag{2.6}$$

Note that this quasi-Newton equation (2.5) contains both gradient and function value information at the current and the previous step.

Wei, Li, and Qi replaced all the $y_k$ in (2.3), and obtained the following modified BFGS-type update formula

$$B_{k+1} = B_k - \frac{B_k s_k s_k^{\mathrm{T}} B_k}{s_k^{\mathrm{T}} B_k s_k} + \frac{y_k^*(y_k^*)^{\mathrm{T}}}{s_k^{\mathrm{T}} y_k^*}, \tag{2.7}$$

where $y_k^* = y_k + \lambda_k s_k$. The modified BFGS-type update formula differs from the standard BFGS update, and a higher order approximation of $\nabla^2 f(x)$ can be obtained (see [14]).

Let $H_k$ be the inverse Hessian approximation of $f$. Then the inverse update formula of (2.7) can be written as

$$\begin{aligned} H_{k+1} &= H_k + \frac{(s_k - H_k y_k^*)s_k^{\mathrm{T}} + s_k(s_k - H_k y_k^*)^{\mathrm{T}}}{s_k^{\mathrm{T}} y_k^*} - \frac{(y_k^*)^{\mathrm{T}}(s_k - H_k y_k^*)}{(s_k^{\mathrm{T}} y_k^*)^2} s_k s_k^{\mathrm{T}} \\ &= \left(I - \frac{s_k (y_k^*)^{\mathrm{T}}}{s_k^{\mathrm{T}} y_k^*}\right) H_k \left(I - \frac{y_k^* s_k^{\mathrm{T}}}{s_k^{\mathrm{T}} y_k^*}\right) + \frac{s_k s_k^{\mathrm{T}}}{s_k^{\mathrm{T}} y_k^*}. \end{aligned} \tag{2.8}$$

The properties of the modified BFGS-type method, especially the global and the superlinear convergence properties, are analyzed in [14,12], respectively. Numerical experiments in [14] also show that the modified BFGS-type method is competitive with the standard BFGS method.

## 3. Limited memory BFGS-type method

In this section, we propose a new algorithm to solve (1.1). This method generates a sequence of points $\{x_k\}$ by

$$x_{k+1} = x_k + \alpha_k d_k, \quad k = 0, 1, 2, \ldots$$

where $d_k$ is a descent direction of $f$ at $x^k$, and $\alpha_k$ is the steplength which is determined by a line search. In the following, we describe the method in details.

The L-BFGS method is an adaptation of the BFGS method to large-scale problems. In the L-BFGS method, matrix $H_k$ is obtained by updating the basic matrix $H_0 \widehat{m}$ times using BFGS formula with the previous $\widehat{m}$ iterations. The standard BFGS correction with $H_k$ has the following form

$$H_{k+1} = V_k^{\mathrm{T}} H_k V_k + \rho_k s_k s_k^{\mathrm{T}}, \tag{3.1}$$

where $\rho_k = \frac{1}{y_k^{\mathrm{T}} s_k}$, and $V_k = I - \rho_k y_k s_k^{\mathrm{T}}$. Therefore, $H_{k+1}$ in the L-BFGS method has the following form:

$$\begin{aligned} H_{k+1} &= V_k^{\mathrm{T}} H_k V_k + \rho_k s_k s_k^{\mathrm{T}} \\ &= V_k^{\mathrm{T}} [V_{k-1}^{\mathrm{T}} H_{k-1} V_{k-1} + \rho_{k-1} s_{k-1} s_{k-1}^{\mathrm{T}}] V_k + \rho_k s_k s_k^{\mathrm{T}} \\ &= \cdots \\ &= [V_k^{\mathrm{T}} \cdots V_{k-\widehat{m}+1}] H_{k-\widehat{m}+1} [V_{k-\widehat{m}+1} \cdots V_k] \\ &\quad + \rho_{k-\widehat{m}+1} [V_{k-1}^{\mathrm{T}} \cdots V_{k-\widehat{m}+2}^{\mathrm{T}}] s_{k-\widehat{m}+1} s_{k-\widehat{m}+1}^{\mathrm{T}} [V_{k-\widehat{m}+2} \cdots V_{k-1}] + \cdots + \rho_k s_k s_k^{\mathrm{T}}. \end{aligned} \tag{3.2}$$

To improve the performance of the standard limited memory BFGS algorithm, it is better to use the modified BFGS-type update instead of the standard BFGS update. If we replace all the $y_k$ with $y_k^*$ in (3.2), the new limited memory BFGS-type update can be obtained

$$\begin{aligned} H_{k+1} &= [V_k^{*\mathrm{T}} \cdots V_{k-m+1}^*] H_{k-\widehat{m}+1} [V_{k-\widehat{m}+1}^* \cdots V_k^*] \\ &\quad + \rho_{k-\widehat{m}+1}^* [V_{k-1}^{*\ \mathrm{T}} \cdots V_{k-\widehat{m}+2}^{*\ \mathrm{T}}] s_{k-\widehat{m}+1} s_{k-\widehat{m}+1}^{\mathrm{T}} [V_{k-\widehat{m}+2}^* \cdots V_{k-1}^*] + \cdots + \rho_k^* s_k s_k^{\mathrm{T}}, \end{aligned} \tag{3.3}$$

where $y_k^* = y_k + \lambda_k s_k$, $\rho_k^* = \frac{1}{y_k^{*\mathrm{T}} s_k}$ and $V_k^* = I - \rho_k^* y_k^* s_k^{\mathrm{T}}$.

Now, we state the steps of the new limited memory BFGS-type (L-BFGS-T) algorithm with weak Wolfe–Powell line search as follows.

**Algorithm 3.1** (*L-BFGS-T*). **Step 0.** Choose an initial point $x_0 \in \mathbf{R}^n$ and a symmetric positive definite matrix $H_0$. Let $0 < \delta < \frac{1}{2}$, $\sigma \in (\delta, 1)$, and a positive integer $m$ exist. Set $k = 0$.

**Step 1.** If $\|g_k\| = 0$ then stop.

**Step 2.** Determine $d_k$ by $d_k = -H_k g_k$.

**Step 3.** Find a steplength $\alpha_k$ satisfying the weak Wolfe–Powell conditions

$$f(x_k + \alpha_k d_k) \le f(x_k) + \delta \alpha_k g_k^{\mathrm{T}} d_k, \tag{3.4}$$

$$g_{k+1}^{\mathrm{T}} d_k \ge \sigma g_k^{\mathrm{T}} d_k. \tag{3.5}$$

Moreover, if $\alpha_k = 1$ satisfies (3.4) and (3.5), we take $\alpha_k = 1$.

**Step 4.** Set $x_{k+1} = x_k + \alpha_k d_k$.

**Step 5.** Let $\widehat{m} = \min\{k+1, m\}$. Update $H_0$ for $\widehat{m}$ times to get $H_{k+1}$ by (3.3).

**Step 6.** Set $k = k + 1$. Go to Step 1.

Clearly, we note that the above algorithm is as simple as the L-BFGS method from storage and cost points of view at each iteration.

## 4. Convergence analysis

This section is devoted to show that Algorithm 3.1 is convergent on twice continuously differentiable and uniformly convex function, and its convergence rate is R-linear. In the following, we assume that the algorithm updates $B_k$—the inverse of $H_k$. We also assume that the basic matrix $B_0$, and its inverse $H_0$, are bounded. The Algorithm 3.1 with Hessian approximation $B_k$ can be stated as follows.

**Algorithm 4.1** (*L-BFGS-T2*). **Step 2.** Determine $d_k$ by $d_k = -B_k^{-1} g_k$.

**Step 5.** Let $\widehat{m} = \min\{k+1, m\}$. Update $B_0$ for $\widehat{m}$ times with the triples $\{s_i, y_i, \lambda_i\}_{i=k-\widehat{m}+1}^{k}$, i.e. for $l = k-\widehat{m}+1, \ldots, k$ compute

$$B_k^{(l+1)} = B_k^{(l)} - \frac{B_k^{(l)} s_l s_l B_k^{(l)}}{s_l^{\mathrm{T}} B_k^{(l)} s_l} + \frac{y_l^* y_l^{*\mathrm{T}}}{y_l^{*\mathrm{T}} s_l}, \tag{4.1}$$

where $s_l = x_{l+1} - x_l$, $y_l^* = y_l + \lambda_l s_l$ and $B_k^{(k-\widehat{m}+1)} = B_0$ for all $k$.

Note that Algorithms 3.1 and 4.1 are mathematically equivalent. In our numerical experiments we implement Algorithm 3.1 and Algorithm 4.1 is given only for the purpose of analysis.

In order to establish the global convergence of Algorithm 4.1, we need some assumptions first.

**Assumption 4.1.** The level set $\Omega = \{x \mid f(x) \le f(x_0)\}$ is bounded.

**Assumption 4.2.** The function $f$ is twice continuously differentiable on $\Omega$.

**Assumption 4.3.** The function $f$ is uniformly convex, i.e., there exist two positive constants $M_1$ and $M_2$ such that

$$M_1\|z\|^2 \le z^{\mathrm{T}} G(x) z \le M_2\|z\|^2 \tag{4.2}$$

holds for all $z \in \mathbf{R}^n$ and $x \in \Omega$.

Assumption 4.1 and the inequality (3.4) indicate that $\{f(x_k)\}$ is a nonincreasing sequence, which ensures that $\{x_k\} \subset \Omega$ and there exists a local minimizer $x^*$ such that

$$\lim_{k\to\infty} f(x_k) = f(x^*).$$

An immediate consequence of Assumption 4.2 is that there exists a constant $L \ge 0$ such that

$$\|g(x) - g(y)\| \le L\|x - y\|, \quad \forall\, x, y \in \Omega. \tag{4.3}$$

For the sake of simplicity, from now on, we abbreviate $f(x_k)$, $f(x^*)$ as $f_k$, $f^*$, respectively.

**Theorem 4.1.** *Suppose that Assumptions 4.1–4.3 hold. Then for any positive definite matrix $B_0$, the sequence $\{x_k\}$ generated by Algorithm 4.1 is convergent to $x^*$, and the convergence rate is R-linear, that is, there is a constant $0 \le \gamma < 1$ such that*

$$f_k - f^* \le \gamma^k (f_0 - f^*). \tag{4.4}$$

**Proof.** Let $\lambda_k$ be chosen as in (2.6), following the definition of $y_k^*$, we have

$$\begin{aligned} s_k^{\mathrm{T}} y_k^* &= s_k^{\mathrm{T}} y_k + s_k^{\mathrm{T}} \lambda_k s_k = 2[f_k - f_{k+1}] + 2g_{k+1}^{\mathrm{T}} s_k \\ &= 2[-g_{k+1}^{\mathrm{T}} s_k + \frac{1}{2} s_k^{\mathrm{T}} G(x_k + \theta(x_{k+1} - x_k)) s_k] + 2g_{k+1}^{\mathrm{T}} s_k \\ &= s_k^{\mathrm{T}} G(x_k + \theta(x_{k+1} - x_k)) s_k, \end{aligned}$$

where $\theta \in (0, 1)$. Combining with Assumption 4.3, it is easy to see that

$$M_1\|s_k\|^2 \le s_k y_k^* \le M_2\|s_k\|^2. \tag{4.5}$$

According to the definition of $y_k^*$, we have

$$\begin{aligned} \|y_k^*\| &= \|y_k + \frac{2[f_k - f_{k+1}] + (g_{k+1} + g_k)^{\mathrm{T}} s_k}{\|s_k\|^2} s_k\| \\ &\le \|y_k\| + \frac{|2[f_k - f_{k+1}] + (g_{k+1} + g_k)^{\mathrm{T}} s_k|}{\|s_k\|} \end{aligned}$$

$$
\begin{aligned}
&\le 2\|y_k\| + \frac{|s_k^{\mathrm{T}} G(x_k + \theta(x_{k+1} - x_k))s_k|}{\|s_k\|} \\
&\le 2L\|s_k\| + M_2\|s_k\| \\
&= (2L + M_2)\|s_k\|,
\end{aligned}
\tag{4.6}
$$

where $\theta \in (0, 1)$. From (4.5) and (4.6), we have

$$
\frac{\|y_k^*\|^2}{s_k^{\mathrm{T}} y_k^*} \le \frac{(2L + M_2)^2\|s_k\|^2}{M_1\|s_k\|^2} = \frac{(2L + M_2)^2}{M_1} = M. \tag{4.7}
$$

Let $\mathrm{tr}(B)$ be the trace of $B$. Then from (4.1) and (4.7), and the boundedness of $\|B_k^{(k-\widehat{m}+1)}\|$, we obtain

$$
\begin{aligned}
\mathrm{tr}(B_{k+1}) &\le \mathrm{tr}(B_k^{(k-\widehat{m}+1)}) + \sum_{l=k-\widehat{m}+1}^{k} \frac{\|y_l^*\|^2}{s_l^{\mathrm{T}} y_l^*} \\
&\le \mathrm{tr}(B_k^{(k-\widehat{m}+1)}) + \widehat{m}M \\
&\le M_3,
\end{aligned}
\tag{4.8}
$$

for some positive constant $M_3$. There is also a simple expression for the determinant

$$
\begin{aligned}
\det(B_{k+1}) &= \det(B_k^{(k-\widehat{m}+1)}) \prod_{l=k-\widehat{m}+1}^{k} \frac{y_l^{*\mathrm{T}} s_l}{s_l^{\mathrm{T}} B_k^{(l)} s_l} \\
&= \det(B_k^{(k-\widehat{m}+1)}) \prod_{l=k-\widehat{m}+1}^{k} \frac{y_l^{*\mathrm{T}} s_l}{s_l^{\mathrm{T}} s_l} \frac{s_l^{\mathrm{T}} s_l}{s_l^{\mathrm{T}} B_k^{(l)} s_l}.
\end{aligned}
\tag{4.9}
$$

Since by (4.8) the largest eigenvalue of $B_k^{(l)}$ is also less than $M_3$, using (4.5) and the boundedness of $\|B_k^{(k-\widehat{m}+1)^{-1}}\|$, we have,

$$
\begin{aligned}
\det(B_{k+1}) &\ge \det(B_k^{(k-\widehat{m}+1)}) \left(\frac{M_1}{M_3}\right)^{\widehat{m}} \\
&\ge M_4,
\end{aligned}
\tag{4.10}
$$

for some positive constant $M_4$. Therefore from (4.8) and (4.10) we conclude that there is a constant $\xi > 0$ such that

$$
\cos\theta_k = \frac{s_k^{\mathrm{T}} B_k s_k}{\|s_k\| \cdot \|B_k s_k\|} \ge \xi. \tag{4.11}
$$

One can see that the line search conditions (3.4) and (3.5) and Assumptions 4.1–4.3 imply that there is a constant $c > 0$ such that

$$
f(x_{k+1}) - f(x^*) \le (1 - c\cos^2\theta_k)(f(x_k) - f(x^*)).
$$

Using (4.11) we obtain (4.4).

From (4.2), we know the following inequality holds

$$
\frac{1}{2}M_1\|x_k - x^*\|^2 \le f_k - f^*,
$$

which together with (4.4) implies $\|x_k - x^*\| \le \gamma^{\frac{k}{2}}[2(f_0 - f^*)/M_1]^{1/2}$, so that the sequence $\{x_k\}$ is R-linearly convergent, too. $\square$

## 5. Numerical experiments

The main aim of this section is to report the performance of Algorithm 3.1, then to compare with the methods of the standard L-BFGS and prp+ methods. Our experiments are performed on a set of 49 nonlinear unconstrained problems

Table 1
Test problem and its dimension

| Problem | Dimension |
|---|---|
| bdqrtic, tridia, arwhead, nondia, nondquar, dqdrtic, eg2, dixmaana, dixmaanb, liaruhd, dixmaanc, dixmaane, edensch, vardim, liarwhd, dixon3dq, engval1, fletchcr, honcvxu2, dixmaanf, dixmaang, dixmaanh, dixmaani, dixmaanj, dixmaank, dixmaanl, broyden7d, cosine, e-denschnb, e-denschnf, sinquad, biggsb1, genrose, nondia, penalty1, brownal, power, freuroth, srosenbr, woods, fletchbv, dqrtic, fletchbv3, bdexp, genhvmps, indef | 1000, . . . , 10000 |
| arglinb, penalty2, bratuld | 100, . . . , 1000 |

that have second derivatives available. These test problems are all from the CUTE (see [22]) collection. For each test problem we consider 10 numerical experiments with number of variables ranging from 1000 to 10000 (or 100 to 1000). That is to say, there are 490 test problems which we used. The name of the problems and their dimensions are listed in Table 1.

The codes are written in Fortran77 and in double precision arithmetic. All runs are performed on a PC (`CPU P4 2.6 GHz, 256M memory`) with `RedHat Linux` operation system. For each test problem, the termination condition is that

$$\|g(x_k)\| \leq 10^{-5}. \tag{5.1}$$

For each problem, we choose the initial matrix $H_0 = I$, i.e., the identity matrix. We will test the following three category of methods:

• `l-bfgs-t`: Algorithm 3.1 with $\delta = 10^{-4}$, $\sigma = 0.1$ in weak Wolfe–Powell conditions. The number of correction pairs which were used is $m = 5$.

• `l-bfgs`: The L-BFGS method in [3,4], and the Fortran code are authored by J. Nocedal.

• `prp+`: The conjugate gradient method in [6], and the Fortran code are coauthored by G. Liu, J. Nocedal, and R. Waltz.

The l-bfgs and prp+ codes are obtained from J. Nocedal's web page at

http://www.ece.northwestern.edu/˜nocedal/software.html.

When running the above two programs, default values are used for all parameters, and the stopping criterion is (5.1). We also force the routine to stop if the number of function evaluations exceed 2000. Since a large set of problems are used, we describe the results completely on the author's home page at the following web site:

http://maths.henu.edu.cn/szdw/teachers/xyh.htm.

The performance of the three algorithms, relative to CPU time, is evaluated using the profiles of Dolan and Moré (see [23]). That is, for the subset of the methods being analyzed, we plot the fraction $P$ of problems for which any given method is within a factor $\tau$ of the best time. The ideas are used with other two measures. In Figs. 1–3, we display the performance profiles of Dolan and Moré for l-bfgs, l-bfgs-t, and prp+, referring to the number of iterations, number of function and gradient evaluations, and CPU time, respectively.

In this series of experiments, l-bfgs-t and prp+ can solve over 90% of the test problems, and l-bfgs solves 80%. This can be observed in Fig. 3 where the parameter $\tau$ is large enough ($\tau = 50$ here). Observing Figs. 1 and 2 respectively, it can be concluded that l-bfgs-t is always the top performer for all values of $\tau$. The two figures indicate that l-bfgs-t seems to be the best by comparison with l-bfgs and prp+. This is not surprising when we consider that the modified BFGS update always get a better approximation of the Hessian matrix at each iteration.

Fig. 3 shows the implementation of the l-bfgs-t method with l-bfgs and prp+ methods using the total CPU time as a measure. This figure shows that the prp+ method is faster than the others. We observe that the iterative form of prp+ is very simple and requires low storage, which explains why the computing times of prp+ are good in spite of its large number of function and gradient evaluations. At each iteration, l-bfgs-t does not require more storage or arithmetic operations as l-bfgs. Moreover, a higher order accuracy in approximating the Hessian matrix of the objective function makes l-bfgs-t need less iterations, less function and gradient evaluations. It is not surprising that l-bfgs-t can be faster than l-bfgs. Fig. 3 seems to indicate that l-bfgs-t is comparable with the others in efficiency, and it is about 10% faster than l-bfgs.

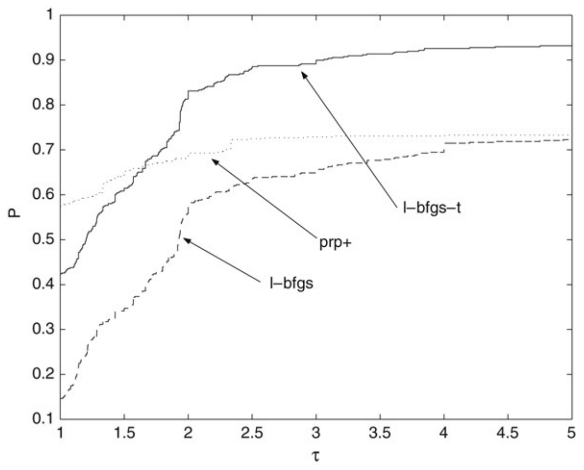

Fig. 1. Performance profiles based on iterations.

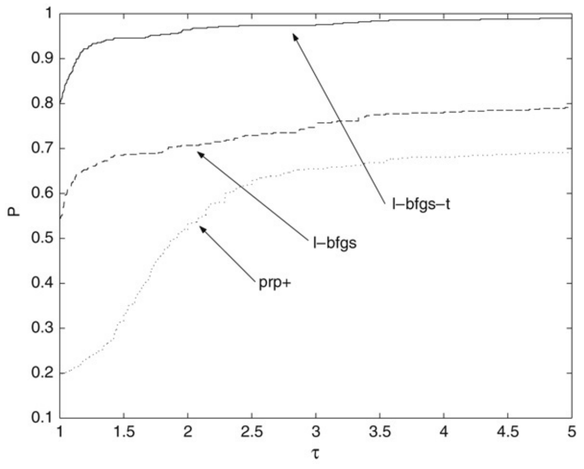

Fig. 2. Performance profiles based on function and gradient evaluations.

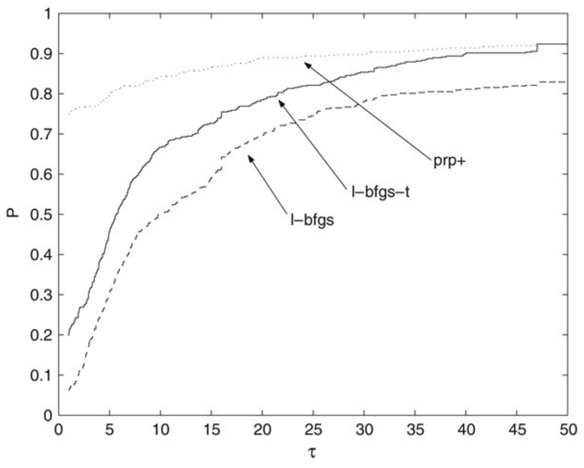

Fig. 3. Performance profiles based on CUP time.

From the three figures, we conclude that, l-bfgs-t performs better than the l-bfgs method does, which requires less iterations, less function and gradient evaluations, and little time. Moreover, preliminary experimental comparisons also indicate that our extension is very beneficial to the performance.

## Acknowledgments

We would like to thank Professor D.H. Li for several helpful conversations. We are also very grateful to two anonymous referees for their useful suggestions and comments on the previous version of this paper. The second author's work was supported by Chinese NSF grant 10761001.

## References

[1] S.G. Nash, A survey of truncated-Newton methods, J. Comput. Appl. Math. 124 (2000) 45–59.
[2] R. Dembo, T. Steihaug, Truncated-Newton algorithms for large-scale unconstrained optimization, Math. Program. 26 (1983) 190–212.
[3] D. Liu, J. Nocedal, On the limited memory BFGS method for large-scale optimization, Math. Program. 45 (1989) 503–528.
[4] J. Nocedal, Updating quasi-Newton matrices with limited storage, Math. Comput. 35 (1980) 773–782.
[5] W.W. Hager, H. Zhang, A new conjugate gradient method with guaranteed descent and an efficient line search, SIAM J. Optim. 16 (2005) 170–192.
[6] J.C. Gilbert, J. Nocedal, Global convergence properties of conjugate gradient methods for optimization, SIAM J. Optim. 2 (1992) 21–42.
[7] L. Zhang, W. Zhou, D.H. Li, Global convergence of a modified Fletcher–Reeves conjugate gradient method with Armijo-type line search, Numer. Math. 104 (2006) 561–572.
[8] L. Zhang, W. Zhou, D.H. Li, A descent modified Polak–Ribière–Polyak conjugate gradient method and its global convergence, IMA J. Numer. Anal. 26 (2006) 629–640.
[9] W.W. Hager, H. Zhang, A survey of nonlinear conjugate gradient methods, Pacific J. Optim. 2 (2006) 35–58.
[10] N.I.M. Gould, D. Orban, Ph.L. Toint, Numerical methods for large-scale nonlinear optimization, Acta Numer. 14 (2005) 299–361.
[11] J. Nocedal, Large Scale Unconstrained Optimization, Duff and Watson, 1997, pp. 311–338.
[12] Z. Wei, G. Yu, G. Yuan, Z. Lian, The superlinear convergence of a modified BFGS-type method for unconstrained optimization, Comput. Optim. Appl. 29 (2004) 315–332.
[13] J.Z. Zhang, N.Y. Deng, L.H. Chen, New quasi-Newton equation and related methods for unconstrained optimization, J. Optim. Theory Appl. 102 (1999) 147–167.
[14] Z. Wei, G. Li, L. Qi, New quasi-Newton methods for unconstrained optimization problems, Appl. Math. Comput. 175 (2006) 1156–1188.
[15] C.G. Broyden, J.E. Dennis, J.J. Moré, On the local and superlinear convergence of quasi-Newton methods, J. Inst. Math. Appl. 12 (1973) 223–246.
[16] R.H. Byrd, J. Nocedal, A tool for the analysis of quasi-Newton methods with application to unconstrained minimization, SIAM J. Numer. Anal. 26 (1989) 727–739.
[17] R.H. Byrd, J. Nocedal, Y. Yuan, Global convergence of a class of quasi-Newton methods on convex problems, SIAM J. Numer. Anal. 24 (1987) 1171–1189.
[18] Y.H. Dai, Convergence properties of the BFGS algorithm, SIAM J. Optim. 13 (2002) 693–701.
[19] W.F. Mascarenhas, The BFGS method with exact line search fails for non-convex objective functions, Math. Program. 99 (2004) 49–61.
[20] D.H. Li, M. Fukushima, A modified BFGS method and its global convergence in nonconvex minimization, J. Comput. Appl. Math. 129 (2001) 15–35.
[21] D.H. Li, M. Fukushima, On the global convergence of the BFGS methods for nonconvex unconstrained optimization problems, SIAM J. Optim. 11 (2001) 1054-164.
[22] A.R. Conn, N.I.M. Gouldc, Ph.L. Toint, CUTE: Constrained and unconstrained testing environment, ACM Trans. Math. Softw. 21 (1995) 123–160.
[23] E.D. Dolan, J.J. Moré, Benchmarking optimization software with performance profiles, Math. Program. 91 (2002) 201–213.