Available online at www.sciencedirect.com
ScienceDirect
Applied Mathematics and Computation 185 (2007) 350–359
APPLIED MATHEMATICS AND COMPUTATION
www.elsevier.com/locate/amc

A new subspace limited memory BFGS algorithm for large-scale bound constrained optimization

Yunhai Xiao [a,*], Zengxin Wei [b]

[a] College of Mathematics and Econometrics, Hunan University, Changsha 410082, PR China
[b] College of Mathematics and Information Science, Guangxi University, Nanning 53004, PR China

Abstract

An active set limited memory BFGS algorithm for large-scale bound constrained optimization is introduced. The active sets are based on guessing technique to be identified at each iteration, the search direction in free subspace is determined by limited memory BFGS (L-BFGS) algorithm, which provides an efficient means for attacking large-scale optimization problems. The implementations of the method on CUTE test problems are described.

Keywords: Nonlinear optimization; Bound constrained problem; Limited memory method; Stationary point; Gradient projection method

1. Introduction

The nonlinear programming problem with simple bounds on variables to be considered is

$$\min\ f(x) \quad \text{s.t. } l \leqslant x \leqslant u, \tag{1.1}$$

where $f: R^n \to R$ is a nonlinear function whose gradient $\nabla f(x)$ is available, and denote $\nabla f(x^k)$, or simply g^k, the vectors l and u represent lower and upper bounds on the variables, respectively, n is the number of variables, which is assumed to be large.

Early methods for this problem tended to be of the active set variety (see [15,19]). Methods of this type are quite efficient for problems of relatively small dimension, but are typically unattractive for large-scale problems. The main reason is that typically at most one constraint can be added to or dropped from the active set at each iteration, and the potential worst-case complexity in which each of the possible 3^n active sets is visited before discovering the optimal one. Several authors have alluded to design active set methods that are capable of making rapid changes to incorrect predictions (see [1,12,13,22,23]), but it is now more common to use gradient projection methods.

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

0096-3003/$ - see front matter
doi:10.1016/j.amc.2006.06.119

The gradient project method (see [1]) is a constructive method, which bending the search direction along the constraint boundary to add to or drop from the current estimated active set many constraints at each iteration, and yet find the active set in a finite number of steps. This work has motivated further studies on projection techniques both for the general linearly constrained case and for the bound constrained case (see [3,7,9,17,18,22]).

In 1980's, many authors have considered extending the trust region concepts and algorithms to the constrained minimization case, Conn et al. [9] have proposed a class of trust region algorithms for simple bound constrained nonlinear optimization. When the strict complementarity condition holds, they show that these iterative processes detect the set of active bounds at the solution after a finite number of iterations. It follows that their generalization of the Cauchy point concept allows the calculation to be reduced to an unconstrained case and the rate of convergence analysis for unconstrained optimization to be applied without any modification. Moreover, Lescrenier [18] established the superlinear convergence rate without requiring strict complementary condition under appropriate assumptions. Furthermore, preliminary numerical results show that these methods are effective and suggest that they are well suited to solve large-scale problems (see [11,17]).

There have been a number of comparative studies of algorithms for bound constrained optimization (see [4,13,17,26]), but none of these makes a compelling case as to the best approach for large-scale case [14]. In practice, both gradient projection and interior point methods [8] appear to require a modest number of iterations [14].

In this paper, we present a new algorithm that combines an active set strategy with the gradient projection method. As in Facchinei and Lucidi [12], we avoid the necessity of finding an exact minimizer of quadratic subproblem with bound constraints. The main difference between the approach of this paper and the one given in Ni and Yuan [23] is that we use the identified technique in [12], as a sequence, the direction of the inactive variables was defined more simply. Some desirable features of the algorithm include: All iterates are feasible and the sequence of objective functions is decreasing; rapid changes in the active set are allowed; a global convergence theory is established. Moreover, it reserves the advantage of the effective active set identified technique in [12], and use the superiority of the subspace limited memory BFGS method (see [23]) which has been proved much suit for solving large-scale problems.

The paper is organized as follows: we first discuss the construction of the algorithm, and list some properties of the search direction in the next section. In Section 3, the global convergence of the algorithm is proved. The implementation of the method is described in Section 4; Furthermore, we give some conclusions in the last section.

Notation. The number of variables in the optimization problem is n, and the number of correction pairs used in the limited memory methods is m. The Hessian inverse approximation is denoted by H^k. The set of the free variables at iteration k is F^k and $\overline{H}^k$ denotes the reduced matrix of H^k with the index of rows and columns are all in F^k, the $|F^k|$ denote the length of the set F^k. $\nabla f_i(x)$ denotes the ith component of vector $\nabla f(x)$, i.e. $\nabla f_i(x) = \frac{\partial f(x)}{\partial x_i}$. Thoughtout the paper, $\|\cdot\|$ denote the Eucilidean norm.

2. Algorithm

In this section we show how to define at each feasible point x^k a search direction d^k which can be used in connection with the projected search. We first discuss the determination of search directions based on guessing technique in [12]. We set the feasible region $K = \{x \in R^n : l_i \leqslant x_i \leqslant u_i, i = 1, \ldots, n\}$, a vector $\bar{x} \in K$ is said to be a stationary point for problem (1.1) if it satisfies

$$\begin{cases} l_i = \bar{x}_i & \Rightarrow \quad \nabla f_i(\bar{x}) \geqslant 0, \\ l_i < \bar{x}_i < u_i & \Rightarrow \quad \nabla f_i(\bar{x}) = 0, \\ \bar{x}_i = u_i & \Rightarrow \quad \nabla f_i(\bar{x}) \leqslant 0. \end{cases} \tag{2.1}$$

Strict complementarity is said to hold at $\bar{x}$ if the strict inequality hold in the first and the third implication of (2.1).

In order to introduce the procedure that estimate the active bounds, let $\bar{x} \in R^n$ be a stationary point of problem (1.1), and consider the associated active constraint set

$$\overline{L} = \{i : l_i = \bar{x}_i\},\ \overline{U} = \{i : \bar{x}_i = u_i\}. \tag{2.2}$$

Furthermore let

$$\overline{F} = \{1, \ldots, n\} \setminus (\overline{L} \cup \overline{U})$$

be the set of the free variables. By using this notation, the condition (2.1) can be stated in the form:

$$\begin{cases} \nabla f_i(\bar{x}) \geqslant 0 & \forall i \in \overline{L}, \\ \nabla f_i(\bar{x}) = 0 & \forall i \in \overline{F}, \\ \nabla f_i(\bar{x}) \leqslant 0 & \forall i \in \overline{U}. \end{cases} \tag{2.3}$$

Then it seems fairly natural to define the following approximations $L(x)$, $F(x)$ and $U(x)$ to $\overline{L}$, $\overline{F}$ and $\overline{U}$ respectively:

$$\begin{aligned} L(x) &= \{i : x_i \leqslant l_i + a_i(x)\nabla f_i(x)\}, \\ U(x) &= \{i : x_i \geqslant u_i + b_i(x)\nabla f_i(x)\}, \\ F(x) &= \{1, \ldots, n\} \setminus (L \cup U), \end{aligned} \tag{2.4}$$

where $a_i(x)$ and $b_i(x)$ are nonnegative continuous bounded from above on K, such that if $x_i = l_i$ or $x_i = u_i$ then $a_i(x) > 0$ or $b_i(x) > 0$ respectively. Other identified technique can be consult to [4,23]. The following results comes from [12, Theorem 3] shows that $L(x)$, $F(x)$ and $U(x)$ are indeed "good" estimate of $\overline{L}$, $\overline{F}$ and $\overline{U}$ respectively.

Theorem 2.1. *For any feasible x, $L(x) \cap U(x) = \emptyset$. Furthermore, if $\bar{x}$ is a stationary point of problem (1.1) where strict complementarity holds, then there exists a neighborhood of $\bar{x}$ such that for every feasible point x in this neighborhood we have*

$$L(x) = \overline{L}, \quad F(x) = \overline{F}, \quad U(x) = \overline{U}.$$

Now let $x^k \in K$ be the current point at iteration k. Consider the sets $L^k = L(x^k)$, $U^k = U(x^k)$ and $F^k = F(x^k)$, the subspace direction $d^k_{F^k}$ is chosen as the search direction for the inactive variables. Let Z be the matrix whose columns are $\{e_i | i \in F^k\}$, where e_i is the ith column of the identity matrix in $R^{n\times n}$, and H^k be an approximation of the full space inverse Hessian matrix. Let $\overline{H}^k \in R^{|F^k|\times|F^k|}$ be an approximation of the reduced inverse Hessian matrix, then $\overline{H}^k = Z^{\mathrm{T}} H^k Z$. The search direction $d^k = (d^k_{L^k}, d^k_{F^k}, d^k_{U^k})$ chosen as

$$d^k_i = l_i - x^k_i, \quad i \in L^k; \tag{2.5}$$

$$d^k_i = u_i - x^k_i, \quad i \in U^k; \tag{2.6}$$

$$d^k_i = -(Z\overline{H}^k Z^{\mathrm{T}} g^k)_i, \quad i \in F^k. \tag{2.7}$$

The projected search has been used by several authors for solving quadratic and nonlinear programming problems with bound bounds on the variables (see [1,2,20,23]). The projected search requires that a steplength, $\alpha_k > 0$, which produces a sufficient decrease in the function $\phi^k : R \to R$ defined by

$$\phi^k(\alpha) = f([x^k + \alpha d^k]^+),$$

where $[\cdot]^+$ is the projection into K defined by

$$[x]^+ = \begin{cases} x_i & \text{if } l_i \leqslant x_i \leqslant u_i, \\ l_i & \text{if } x_i < l_i, \\ u_i & \text{if } x_i > u_i. \end{cases} \tag{2.8}$$

The sufficient decrease condition requires that $\alpha_k > 0$ satisfy

$$\phi^k(\alpha) \leqslant \phi^k(0) + \sigma\nabla\phi^k(0)\alpha \tag{2.9}$$

for some constant $\sigma \in (0, \frac{1}{2})$.

The following result shows that whenever $d^k \neq 0$, it is at least a descent direction for objective function $f(x)$ at current point x^k, the property is very important to establish our global convergence.

Lemma 2.1. *If H^k is positive definite, then d^k defined by (2.5)–(2.7) satisfies*

$$(d^k)^{\mathrm{T}} g^k \leqslant 0. \tag{2.10}$$

Proof. From the definition of search direction d^k, (2.5)–(2.7), we have

$$(d^k)^{\mathrm{T}} g^k = \sum_{i\in L^k}(l_i - x_i^k)g_i^k + \sum_{i\in U^k}(u_i - x_i^k)g_i^k + \sum_{i\in F^k} -g_i^k(Z\overline{H}^k Z^{\mathrm{T}} g^k)_i \leqslant 0.$$

The above relation comes from the positive definite of H^k (so $\overline{H}^k$) and the definition of the active set (2.4), which also indicate $(d^k)^{\mathrm{T}} g^k = 0$ if and only if $d^k = 0$. □

The limited memory BFGS method (see [5,6]) was is an adaptation of the BFGS method to large-scale problems, the only difference is in the matrix update, for getting Hessian inverse approximate H^{k+1}, instead of storing the matrices H^k, at every iteration x^k the method stores a small number, say m, of correction pairs $\{s^i, y^i\}$, $i = k-1, \ldots, k-m$, where

$$s^k = x^{k+1} - x^k, \quad y^k = g^{k+1} - g^k.$$

The standard BFGS correction with H^k have the following form:

$$H^{k+1} = (V^k)^{\mathrm{T}} H^k V^k + \rho^k s^k (s^k)^{\mathrm{T}},$$

where $\rho^k = \frac{1}{(y^k)^{\mathrm{T}} s^k}$ and $V^k = I - \rho^k y^k (s^k)^{\mathrm{T}}$. If we use the stored correction pairs, we get

$$\begin{aligned} H^{k+1} &= (V^k)^{\mathrm{T}}[(V^{k-1})^{\mathrm{T}} H^{k-1} V^{k-1} + \rho^{k-1} s^{k-1}(s^{k-1})^{\mathrm{T}}]V^k + \rho^k s^k (s^k)^{\mathrm{T}} \\ &= (V^k)^{\mathrm{T}}(V^{k-1})^{\mathrm{T}} H^{k-1} V^{k-1} + (V^k)^{\mathrm{T}} \rho^{k-1} s^{k-1}(s^{k-1})^{\mathrm{T}} V^k + \rho^k s^k (s^k)^{\mathrm{T}} = \cdots \\ &= \left[(V^k)^{\mathrm{T}} \ldots (V^{k-m+1})^{\mathrm{T}}\right] H^{k-m+1} [V^{k-m+1} \ldots V^{k-1}] \\ &\quad + \rho^{k-m+1}\left[(V^{k-1})^{\mathrm{T}} \ldots (V^{k-m+2})^{\mathrm{T}}\right] s^{k-m+1}(s^{k-m+1})^{\mathrm{T}}[V^{k-m+2} \ldots V^{k-1}] + \cdots + \rho^k s^k (s^k)^{\mathrm{T}}. \end{aligned} \tag{2.11}$$

These correction pairs contain information about the curvature of the function and, in conjunction with the BFGS formula, define the limited memory iteration matrix.

To maintain the positive definiteness of the limited memory BFGS matrix, some researchers discard a correction pair $\{s^k, y^k\}$ if the curvature $(s^k)^{\mathrm{T}} y^k > 0$ is not satisfied (see [5]). This mean that the m directions in S^k and Y^k may actually include some with indices less than $k-m$. Another approach was proposed by Powell [24,23], they replace s^k with a new $(s^k)'$ by means of some relations.

When used the limited memory update in the bounded constrained optimization problems, the set of active constraints should be changes at first finite steps. One approach is to store $\{s^k, y^k\}_{k-m+1}^k$, and update it as a full matrix, and reduced in the free subspace. This is very costly for even moderately large problems. Another approach of updating H^k is to use only reduced gradient and projected steps, but too much information may be lost. In our numerical experiments, we used the approach based on the recursive BFGS update that does that discard information corresponding to that part of the inactive set that is not changed. At each iteration, we stores the sequence $\{\overline{s}^k\}$ and $\{\overline{y}^k\}$ according to the reduced gradient and projected steps. The algorithm list as follows:

Algorithm 2.1. `UPDATE` $(\mathtt{ns}, \{\overline{s}^k\}, \{\overline{y}^k\}, H^0, d, Z)$

step 1. $d = Z'd$;
step 2. if $ns = 0$, $d = H^0 d$; return;
step 3. $\alpha = \overline{s}_{ns-1}^{\mathrm{T}} d / \overline{y}_{ns-1}^{\mathrm{T}} \overline{s}_{ns-1}^k$; $d = d - \alpha \overline{y}_{ns-1}^k$;
step 4. *call* `UPDATE` $(\mathtt{ns}-1, \{\overline{s}^k\}, \{\overline{y}^k\}, H^0, d, Z)$;
step 5. $d = d + (\alpha - (d^{\mathrm{T}} \overline{y}_{ns-1}^k / \overline{y}_{ns-1}^{kT} \overline{s}_{ns-1}^k))\overline{s}_{n-1}^k$;
step 6. $d = Z'd$.

where $ns \leqslant m$ is the number of the correction pairs. Note that we reinitialize ns to zero, when $(\overline{y}^k)^{\mathrm{T}}\overline{s}^k \leqslant 0$. We found experimentally that this was better than skipping the update.

We now ready to formally state the overall algorithm for solving the bound constrained optimization problems (1.1), which we called the projected active set limited memory BFGS (PAL-BFGS) algorithm.

Algorithm 2.2. (**PAL-BFGS Algorithm**)

Step 0. Given starting point $x^0 \in K$, constant $\sigma \in \left(0, \frac{1}{2}\right)$ and $m \in (3, 20)$, the "basic matrix" θI, nonnegtive continuous $a_i(x)$ and $b_i(x)$; compute $f(x^0)$, $\nabla f(x^0)$ and set $k = 0$.
Step 1. Initialize. *Determine* $L^k = L(x^k)$, $U^k = U(x^k)$, *and* $F^k = F(x^k)$ *according to* (2.4).
Step 2. Determine the search direction. *Compute* d^k *from* (2.5)–(2.7).
Step 3. Stopping test. *If* $d^k = 0$, *stop*; *otherwise*, *continue*.
Step 4. Backtracking line search. *Using the projected line search rule which find* α_k *satisfy* (2.9).
Step 5. Accept the new point. *Set* $x^{k+1} = [x^k + \alpha_k d^k]^+$. *Compute* $f(x^{k+1})$ *and* $\nabla f(x^{k+1})$.
Step 6. Update. *Update* H^k *by means of* (2.11).
Step 7. Continue with the next iteration. *Increase the iteration counter* $k = k + 1$ *and go back to Step* 1.

Remark

(1) Step 4 indicate that an initial trial value of α_k is chosen as 1. For $j = 1, 2, \ldots$, let $\alpha_k = \beta^* \alpha_k$, in our numerical experiments, we choose $\beta = 0.1$ and the number of line searches not past 10 at each iteration.
(2) In our numerical experiments, we did not use (2.11) directly, and we use the Algorithm 2.1 to generate the search direction instead, for more detail can be seen in [6,16].

3. Convergence analysis

In the section we analyze the global convergence of Algorithm 2.2, first, we show that the standard assumption below is fulfilled.

Assumption 3.1. There exists positive scalars ρ_1, ρ_2 such that any matrix $\overline{H}^k$, $k = 1, 2, \ldots$, satisfies

$$\rho_1 \|z\|^2 \leqslant z^{\mathrm{T}} \overline{H}^k z \leqslant \rho_2 \|z\|^2, \text{for all nonzero } z \in R^{|F^k|}.$$

Moreover, for the global convergence analysis of Algorithm 2.2, we list some useful lemmas.

Lemma 3.1. *Let* d^k *be the search direction from* (2.5)–(2.7) *and assume that* $d^k \neq 0$, *then*

$$\min\left\{1, \frac{\|u - l\|_\infty}{\|d^k\|_\infty}\right\} \geqslant \beta^k \geqslant \min\left\{1, \frac{\epsilon_k}{\|d^k\|_\infty}\right\}, \tag{3.1}$$

where $\beta^k = sup_{0 \leqslant \gamma \leqslant 1}\{\gamma | l \leqslant x^k + \gamma d^k \leqslant u\}$, *and* $\epsilon_k = \min\{|a_i(x^k) g_i(x^k)|, |b_i(x^k) g_i(x^k)|, i \in F^k, g_i(x^k) \neq 0\}$.

Proof. By the definition of β^k, x^k and $x^k + \beta^k d^k$ are feasible points of (1.1), which gives

$$\|\beta^k d^k\|_\infty \leqslant \|u - l\|_\infty.$$

Thus the first part of (3.1) is true.

Now we show the second part of (3.1). It is sufficient to prove that

$$x_i^k + \overline{\beta} d_i^k \in [l_i, u_i] \tag{3.2}$$

for all $i = 1, \ldots, n$, where $\overline{\beta} = \min\left\{1, \frac{\epsilon_k}{\|d^k\|_\infty}\right\}$. If $i \in L(x^k)$, it follows from definition (2.5) that $x_i^k + d_i^k = l_i$, similarly for $i \in U(x^k)$. If $i \in F(x^k)$, we have

$$x_i^k > l_i + a_i(x^k) \nabla f_i(x^k),$$
$$x_i^k < u_i + b_i(x^k) \nabla f_i(x^k).$$

Suppose that there exists an $i \in F^k$ such that $\nabla f_i(x^k) < 0$, from the definition (2.7), we have $d_i^k > 0$, then

$$u_i > x_i^k + (-b_i(x^k)\nabla f_i(x^k)) \geqslant x_i^k + \epsilon_k \frac{d_i^k}{\|d^k\|_\infty} \geqslant x_i^k + \overline{\beta} d_i^k.$$

Similarly for $\nabla f_i(x^k) > 0$, we have $x_i^k + \overline{\beta} d_i^k \geqslant l_i$. When $i \in F^k$ and $g_i(x^k) = 0$, the conclusion is obvious. Therefore we have shown that (3.2) holds for all $i = 1, \ldots, n$. □

Lemma 3.2. *Let $x^k \in K$ and d^k is the direction defined by (2.5)–(2.7), then we have*

$$\|d^k\|^2 \leqslant -\gamma \nabla f(x^k)^T d^k, \tag{3.3}$$

for some positive scalar γ.

Proof. Since $\overline{H}^k$ is a symmetric positive definite matrix, from (2.7), we have

$$d_{F^k}^k = -(Z\overline{H}^k Z^{\mathrm{T}} g^k)_{F^k}.$$

From Assumption 3.1 yields

$$\rho_1 \|d_{F^k}^k\|^2 \leqslant -\nabla f_{F^k}(x^k)^{\mathrm{T}} d_{F^k}^k \leqslant \rho_2 \|d_{F^k}^k\|^2,$$

hence

$$\nabla f_{F^k}(x^k)^{\mathrm{T}} d_{F^k}^k \leqslant -\rho_1 \|d_{F^k}^k\|^2. \tag{3.4}$$

Now we prove that there exists a positive scalar γ_i such that

$$\nabla f_i(x^k)^{\mathrm{T}} d_i^k \leqslant -\gamma_i (d_i^k)^2, \tag{3.5}$$

for each $i \in L^k \cup U^k$. If $d_i^k = 0$ the inequality holds trivially. So suppose that $d_i^k \neq 0$. We only show the inequality for $i \in L^k$, since the case of $i \in U^k$ is analogous.

Since $x^k \in K$ and $d_i^k = l_i - x_i^k$ for each $i \in L^k$, then all nonzero d_i^k must be negative. Then it follows from the definition of the set L^k that

$$a_i(x^k)\nabla f_i(x^k) \geqslant -d_i^k. \tag{3.6}$$

Now $a_i(x^k) > 0$, because if $a_i(x^k) = 0$ then $x_i^k = l_i$ and this implies $d_i^k = 0$. So $a_i(x^k) > 0$, $d_i^k < 0$ and hence, from (3.6), we get

$$\nabla f_i(x^k)^{\mathrm{T}} d_i^k \leqslant -\frac{1}{a_i(x^k)} (d_i^k)^2.$$

but $a_i(x^k)$ is bounded from above on K, whence there exists

$$\xi_i \geqslant \sup_{l \leqslant x \leqslant u} a_i(x) > 0$$

and (3.5) holds with $\gamma_i = 1/\xi_i$. So some positive constant exist shows our claims. □

Lemma 3.3. *Let x^k, d^k be given iterates of Algorithm 2.2. Then x^k is a KKT point of (1.1) if and only if $d^k = 0$.*

Proof. Let $d^k = 0$. If $i \in L^k$ then, by (2.4) and (2.5), we have

$$0 = d_i^k = l_i - x_i^k \geqslant -a_i(x^k)\nabla f_i(x^k).$$

But, since $x_i^k = l_i$, $a_i(x^k) > 0$, whence $\nabla f_i(x^k) \geqslant 0$. On the other hand for each $i \in U^k$ we have

$$0 = d_i^k = u_i - x_i^k \leqslant -b_i(x^k)\nabla f_i(x^k),$$

whence $\nabla f_i(x^k) \leqslant 0$. If $d_{F^k}^k = 0$ then, by

$$d_i^k = -(Z\overline{H}^k Z^{\mathrm{T}} g^k)_i, i \in F^k.$$

For $\overline{H}^k$ is a positive definite matrix, we must have $\nabla f_i(x^k) = 0$.

Now suppose that x_k is a stationary point of f on k. Hence it follows from (2.1) and (2.3) that

$$L^k = \{i : x_i^k = l_i\},\ F^k = \{i : l_i < x_i^k < u_i\},\ U^k = \{i : x_i^k = u_i\}.$$

Therefore $d_{L^k} = d_{U^k} = 0$ by (2.5) and (2.6). On the other hand since $\nabla f_{F^k}(x^k) = 0$, $\overline{H}^k$ is a positive definite matrix, and (2.7), hence $d_{F^k} = 0$. Therefore $d^k = 0$. □

It follows from Lemmas 2.1 and 3.3 that d^k is a descent direction if x^k is not a KKT point. Now we prove the global convergence theorem for the Algorithm 2.2.

Theorem 3.1. *Suppose that Assumption 3.1 holds. Let x^k, d^k, and $\overline{H}^k$ be computed by the Algorithm 2.2 for solving the problem (1.1) and assume that $f(x)$ is twice continuously differentiable in K and, there exists a positive constant γ_1 such that $\|Z^T\overline{H}^kZ\| \leqslant \gamma_1$ for all k. Then every accumulation point of $\{x^k\}$ is a KKT point of the problem (1.1).*

Proof. From Lemma 3.2 we have

$$\|d^k\|^2 \leqslant -\gamma \nabla f(x^k)^{\mathrm{T}} d^k.$$

Further,

$$\begin{aligned}\|d^k\|^2 &= \|Z\overline{H}^k Z^{\mathrm{T}} g^k\|^2 + \sum_{i\in L^k}(l_i - x_i^k)^2 + \sum_{i\in U^k}(u_i - x_i^k)^2 \\ &\leqslant \gamma_1\|g^k\|^2 + \sum_{i\in L^k}(a_i(x^k)\nabla f(x^k))^2 + \sum_{i\in U^k}(b_i(x^k)\nabla f(x^k))^2 = (\gamma_1 + \mu_k)\|g^k\|^2 \leqslant (\gamma_1 + \mu_k)\eta_1 \end{aligned} \tag{3.7}$$

where $\mu_k = \sum_{i\in L^k} a_i(x^k)^2 + \sum_{i\in U^k} b_i(x^k)^2$ and $\eta_1 = \max_{x\in K}\|g^k\|^2$. Thus from (3.1) and (3.7), there exists a constant $\overline{\beta} \in (0,1)$ such that

$$\beta_k \geqslant \overline{\beta} \quad \text{for all } k. \tag{3.8}$$

If $\alpha_k < 0.1\overline{\beta}$, by the definition of α_k there exists $j \geqslant 0$ such that $\alpha_{k,j} \leqslant 10\alpha_k$ and $\alpha_{k,j}$ is an unacceptable steplength, which implies that

$$f(x^k) + \sigma\alpha_{k,j}(g_k)^{\mathrm{T}} d^k \leqslant f(x^k + \alpha_k d^k) \leqslant f(x^k) + \alpha_{k,j}(g^k)^{\mathrm{T}} d^k + \frac{1}{2}\eta_2\alpha_{k,j}^2\|d^k\|^2, \tag{3.9}$$

where $\eta_2 = \max_{x\in K}\|\nabla^2 f(x)\|$. The above inequality and (3.3) imply that

$$\alpha_{k,j} \geqslant \frac{-2(1-\sigma)(g^k)^{\mathrm{T}} d^k}{\eta_2\|d^k\|^2} \geqslant \frac{2(1-\sigma)}{\eta_2\gamma}. \tag{3.10}$$

Hence the above inequality and $\alpha_k \geqslant 0.1\alpha_{k,j}$ yield

$$\alpha_k \geqslant \min\left\{\frac{-(1-\sigma)}{5\eta_2\gamma}, 0.1\overline{\beta}\right\} > 0 \tag{3.11}$$

for all k. Because k is a bounded set,

$$\infty > \sum_{k=1}^{\infty}(f(x^k) - f(x^{k+1})) \geqslant \sum_{k=1}^{\infty} -\sigma\alpha_k(g^k)^{\mathrm{T}} d^k. \tag{3.12}$$

(3.11) and (3.12) show that

$$\sum_{k=1}^{\infty} -(g^k)^{\mathrm{T}} d^k < \infty \tag{3.13}$$

which implies

$$\lim_{k\to\infty}(g^k)^{\mathrm{T}} d^k = 0. \tag{3.14}$$

It follows from (3.14) and

$$(d^k)^{\mathrm{T}} g^k = -(g^k)^{\mathrm{T}} Z \overline{H}^k Z^{\mathrm{T}} g^k + \sum_{i \in L^k} (l_i - x_i^k) g_i^k + \sum_{i \in U^k} (u_i - x_i^k) g_i^k$$

that

$$\lim_{k \to \infty} \|Z^{\mathrm{T}} g^k\| = 0, \tag{3.15}$$

$$\lim_{k \to \infty} \sum_{i \in L^k} (l_i - x_i^k) g_i^k = 0, \tag{3.16}$$

$$\lim_{k \to \infty} \sum_{i \in U^k} (u_i - x_i^k) g_i^k = 0. \tag{3.17}$$

Let $\bar{x}$ be any accumulation point of $\{x^i\}$, there exists a subspace $\{x^{k_i}\}$ $(i = 1, 2, \ldots)$ such that

$$\lim_{k \to \infty} x^{k_i} = \bar{x}. \tag{3.18}$$

From (2.2) and (2.3), if $\bar{x}$ is not a KKT point, there exists $j \in \overline{L}$ (or $j \in \overline{U}$) such that

$$g_j(\bar{x}) < 0 \quad (\text{or } g_j(\bar{x}) > 0) \tag{3.19}$$

or there exists $j \in \overline{F}$ such that

$$g_j(\bar{x}) \neq 0. \tag{3.20}$$

If (3.20) holds for some $j \in \overline{F}$, from ((3.15)–(3.17)), imply that for all sufficiently large i, we have $j \notin L(x^{k_i}) \cup U(x^{k_i}) \cup F(x^{k_i})$, it is impossible. □

4. Numerical experiments with the CUTE collection

In this section we examine the numerical behavior of the algorithm on a set of test problems. The code was written in MATLAB and in double precision arithmetic. All runs were performed on PC (CPU P4 2.6 GHz, 256M memory). For each test problem, the termination condition is the projected gradient of the objective function below 10^{-5}, namely

$$\|[x^k - \nabla f(x^k)]^+ - x^k\| \leqslant 10^{-5}. \tag{4.1}$$

Our experiments have been performed on the set of the nonlinear bound constrained problems from [10] collection that have second derivatives available and the dimension can be provided by the the user. We selected the problems both have upper and lower bound. Some test problem which the number of variables can be provided by the user: `BDEXP`, `EXPLIN`, `EXPLIN2`, `PROBPENL`, `EDENSCH`, `MCCORMCK`, `PENALTY1` and `NONSCOMP`, and we chose the dimension between 1000 and 10000. Moreover, we also tested some small problems with fixed dimension in CUTE, namely: `HATFLDA`, `HATFADC` `HS110`. The bounds of problems `EDENSCH` and `PLENTY1` can be refer to [5]. For the starting point, the lower and upper bounds, and some parameters on each test problems can be found in CUTE.

We choose $\sigma = 10^{-1}$ in the Armijo line search, and choose $a_i(x) = 10^{-5}$ and $b_i(x) = 10^{-5}$ in (2.4), this choose is very small and can approximate the active set correctly. In the limited memory method, set $\theta = 1$ then the "basic matrix" is the identity matrix I. In order to assess the reliability of our algorithm, we first tested this method against the well known alternative algorithm MINOS [21] and LOQO [25] using the bound problems with more than 120 variables from the CUTE [10] collection. The stopping criteria of MINOS and LOQO were inhibited, the number of correction pairs used in limited memory method is $m = 5$. The numerical results of these algorithms are listed in Table 1. The columns have the following meanings:

Problem: the name of the test problem in CUTE [10];
Dim: the dimension of the problem;
NI: the number of iterations;

Table 1
Test results for PAL-BFGS and MINOS method for small problems

No.	Problems	Dim	ALBFGS			MINOS			LOQO		
			NI	NF	$f(x)$	NI	NF	$f(x)$	NI	NF	$f(x)$
1	NONSCOMP	100	62	288	8.10e−13	164	424	4.22e+03	27	56	2.19e−09
2	EXPLIN	120	200	874	−7.24e+05	111	144	−5.03e+05	21	41	−5.03e+05
3	EXPLIN2	120	2	3	−7.26e+05	106	122	−5.03e+05	24	47	−5.03e+05
5	MCCORMCK	100	31	43	−9.18e+01	113	224	−9.36e+01	12	23	−9.18e+01
6	PROBPENL	100	3	13	1.98e−06	642	2363	−3.65e+05	500	1001	−1.62e+32
7	BDEXP	100	147	283	−3.61e+89	1	7	−2.69e+44	263	573	−2.87e+152
8	HATFLDC	25	48	142	2.53e−12	34	68	4.55e−16	9	17	4.41e−20
9	HS110	50	2	3	−9.99e+09	50	53	−9.99e+09	14	103	−9.99e+09
10	HATFLDA	4	47	119	3.36e−10	20	42	1.71e−16	60	131	3.35e−17

NF: the number of function evaluations;
$f(x)$: the final function value;
TIME: the CPU time in seconds;
PGNORM: the norm of the final projected gradient;
$^*F_1$: Termination because the number of iterations reached 1000.

We tested this method against some well-known alternative algorithms using the same bounded constrained problems from the CUTE collection. The algorithm that we used for comparing PAL-BFGS is the projected BFGS method PROJBFGS. The PROJBFGS effective for small to very large problems that have appropriate compactness properties (see [16] and the references therein). The MATLAB code was contributed by C.T. Kelley, which is available at `http://www4.ncsu.edu/~ctk/` and, we did not make any modified, the parameters are default. Table 2 tell the numerical results for the method, for each test problem, the termination condition is (4.1).

From Table 2, we observe that convergence was obtained for all the test problems with the two category methods, with the exception of the PAL-BFGS that did not solve `EXPLIN` after more than one thousand iterations. The PAL-BFGS that, in most case, its behavior in the problems where it works quite efficient, which require less iterations, less function evaluations, and little time consuming. In our experiments, we did not compare with other famous methods, such as, the L-BFGS-B (see [5,26]) and the TRON (see [17]), and we

Table 2
Test results for PAL-BFGS method and PROJBFGS method

No.	Problems	Dim	PAL-BFGS				PROJBFGS			
			NI	NF	TIME	PGNORM	NI	NF	TIME	PGNORM
1	NONSCOMP	5000	63	289	5.75	5.745e−06	108	412	7.78	6.723e−06
1	NONSCOMP	10000	63	341	31.45	8.854e−06	121	456	59.55	8.815e−06
2	EXPLIN	1200	$^*F_1$	4158	8.00	3.779e−05	65	228	0.41	9.068e−07
3	EXPLIN2	1200	2	3	0.02	0.000e+00	2	3	0.02	0.000e+00
4	MCCORMCK	1000	33	92	0.24	8.970e−06	69	202	0.84	9.919e−06
5	PROBPENL	5000	3	16	0.25	2.671e−06	2	9	0.09	1.712e−06
6	PROBPENL	10000	3	7	0.61	3.172e−07	78	687	47.86	1.527e−08
8	HATFLDC	25	48	142	0.06	8.567e−06	48	142	0.05	8.567e−06
9	HS110	50	2	3	0.01	0.000e+00	2	3	0.00	0.000e+00
10	HATFLDA	4	47	119	0.04	5.758e−06	72	190	0.06	7.175e−06
11	EDENSCH	1000	73	279	0.61	8.703e−06	73	222	0.59	8.509e−06
12	EDENSCH	1000	66	248	0.56	9.550e−06	77	261	0.64	9.456e−06
14	EDENSCH	1000	13	36	0.13	4.952e−07	30	83	0.23	4.496e−07
15	PENALTY1	1000	68	222	0.45	6.261e−06	94	250	0.57	3.796e−06
16	PENALTY1	1000	123	314	0.73	3.404e−06	159	388	1.03	1.025e−06
17	PENALTY1	1000	21	103	0.14	7.574e−13	25	92	0.17	1.105e−06
18	PENALTY1	1000	19	82	0.14	4.917e−06	39	141	0.26	3.059e−09

leave a further task for us, but we assure that our proposed method represents an efficient way to solve large large-scale bound constrained minimization problems.

Acknowledgement

We are very thankful Professor D.H. Li for providing us the CUTE test problems and his constructive criticisms on this paper.

References

[1] D.P. Bertsekas, Projected Newton methods for optimization problems with simple constrains, SIAM J. Contr. Opt. 20 (1982) 221–246.
[2] E.G. Birgin, J.M. Martínez, Large−scale active-set box-constrained optimization method with spectral projected gradients, Comput. Opt. Appl. 23 (2002) 101–125.
[3] J.V. Burke, J.J. Moré, On the identification of active constraints, SIAM J. Numer. Anal. 25 (1988) 1197–1211.
[4] J.V. Burke, J.J. Moré, Exposing constrints, SIAM J. Opt. 4 (1994) 573–595.
[5] R.H. Ryrd, P.H. Lu, J. Nocedal, A limited memory algorithm for bound constrained optimization, SIAM J. Statist. Sci. Comput. 16 (1995) 1190–1208.
[6] R.H. Ryrd, J. Nocedal, R.B. Schnabel, Representations of quasi-Newton matrices and their use in limited memory methods, Math. Program. 63 (1994) 129–156.
[7] P. Calamai, J.J. Moré, Projected gradient for linearly constrained programs, Math. Program. 39 (1987) 93–116.
[8] T.F. Ccoleman, Y. Li, An interior trust region approach for nonlinear minimization subject to bounds, SIAM J. Opt. 6 (1996) 418–445.
[9] A.R. Conn, N.I.M. Gould, Ph. L. Toint, Global convergence of a class of trust region algorithm for optimization with simple bounds, SIAM J. Numer. Anal. 25 (1988) 433–460.
[10] A.R. Conn, N.I.M. Gould, Ph. L. Toint, CUTE: constrained and unconstrained testing environment, ACM Trans. Math. Softw. 21 (1995) 123–160.
[11] A.R. Conn, N.I.M. Gould, Ph. L. Toint, LANCELOT: A Fortran package for large-scale nonlinear optimization (release A), Springer Series in Computational Mathematics, Springer Verlag, Heidelberg, Berlin, New York, 1992.
[12] F. Facchinei, J. Júdice, J. Soares, An active set Newton algorithm for large-scale nonlinear programs with box canstranits, SIAM J. Opt. 8 (1998) 158–186.
[13] F. Facchinei, S. Lucidi, L. Palagi, A truncated Newton algorithm for large scale box constrained optimization, SIAM J. Opt. 12 (2002) 1100–1125.
[14] N.I.M. Gould, D. Orban, Ph.L. Toint, Numerical methods for large-scale nonlinear optimization, Acta Numer. 14 (2005) 299–361.
[15] D. Goldfarb, Extension of Davidon's variable metric algorithm to maximization under linear inequality and constraints, SIAM J. Appl. Math. 17 (1969) 739–764.
[16] C.T. Kelley, Iterative methods for optimization, Philadelphia, PA, 1999.
[17] C.J. Lin, J.J. Moré, Nowton's method for large bound-constrained optimization problems, SIAM J. Opt. 9 (1999) 1100–1127.
[18] M. Lescrenier, Convergence of trust region algorithm for optimization with bounds when strict complementarity does not hold, SIAM J. Numer. Anal. 28 (1991) 467–695.
[19] D.G. Luenberger, Introduction to Linear and Nonlinear Programming, Addison-Wesley, Reading, MA, 1973, Ch. 11.
[20] J.J. Moré, G. Toraldo, On the solution of large quadratic programming problems with bound constraints, SIAM J. Opt. 1 (1991) 93–113.
[21] B.A. Murtagh, M.A. Saunders, family MINOS 5.5 User's Guide, Report SOL 83-20R, Systems Optimization Laboratory, Stanford University (revised July 1998).
[22] Q. Ni, A subspace projected conjuagte gradient algorithm for large bound constrained quadratic programming, Numer. Math. (a Journal of Chinese Universities) 7 (1998) 51–60.
[23] Q. Ni, Y.X. Yuan, A subspace limited memory quasi-Newton algorithm for large-scale nonlinear bound constrained optimization, Math. Comp. 66 (1997) 1509–1520.
[24] M.J.D. Powell, A fast algorithm for nonlinearly constrained optimization calculations, Numer. Anal. (1978) 155–157.
[25] R.J. Vanderbei, family LOQO: An interior point code for quadratic programming, Technical Report, Statistics and Operation Research, Princeton University, SOR-94-15, 1994.
[26] C.Y. Zhu, R.H. Byrd, P.H. Lu, J. Nocedal, L-BFGS-B Fortran subroutines for large-scale bound constrained optimization, ACM Trans. Math. Softw. 23 (1997) 550–560.