Asia-Pacific Journal of Operational Research
Vol. 27, No. 1 (2010) 71–84

DOI: 10.1142/S0217595910002594

IMPROVED PROJECTED GRADIENT ALGORITHMS FOR SINGLY LINEARLY CONSTRAINED QUADRATIC PROGRAMS SUBJECT TO LOWER AND UPPER BOUNDS

YUN-SHAN FU and YU-HONG DAI*

LSEC, ICMSEC
Academy of Mathematics & Systems Science
Chinese Academy of Sciences, P.O. Box 2719
Beijing 100080, China
* *dyh@lsec.cc.ac.cn*

In this paper, we consider the projected gradient algorithms for solving the quadratic program with bound constraints and a single linear equality constraint (SLBQP). We establish the relationship between the Lagrangian multiplier in the projection subproblem and the Lagrangian multiplier in the original optimization problem. Then we give an improved initial estimate of the Lagrangian multiplier in the subproblem based on this relationship. It appears that this initial estimate is very close to the optimal Lagrangian multiplier after several iterations of the outer loop. This will reduce at most 40% of the computing time in the projection subproblem. This initial guess can also be used in all kinds of projected gradient methods for solving the SLBQP problem. The numerical results show that it brings much more improvement in monotone algorithms than in nonmonotone algorithms. We also apply the adaptive steepest descent step-size and the Dai-Yuan step-size which are two monotone step-sizes to the projected gradient method of this SLBQP problem. Our numerical experiments showed that their performance can be better than some other monotone projected gradient methods.

Keywords: Monotone projected gradient algorithm; Support Vector Machine (SVM); adaptive steepest descent step-size; Dai-Yuan step-size; projection subproblem.

1. Introduction

In this paper, we consider the following quadratic problem

$$\begin{aligned} \min_{x\in\mathbf{R}^n} \quad & f(x) = \frac{1}{2}x^T Ax - c^T x, \\ s.t. \quad & l \le x \le u, \\ & a^T x = b. \end{aligned} \tag{1.1}$$

Here $A \in \mathbf{R}^{n\times n}$ is symmetric but may be indefinite, a, c, l and u (with $l < u$) are vectors in $\mathbf{R}^n$, and b is a scalar. There is a single linear equality constraint in

*Corresponding author.

addition to simple bounds on the variables in this problem. We refer to this as the general SLBQP problem as abbreviated by Dai and Fletcher (2005).

The SLBQP problem appears in a wide range of applications. For example, some problems in multicommodity network and logistics have the form of (1.1) in which the matrix A is diagonal (Held *et al.*, 1974; Meyer, 1984; Pardalos and Rosen, 1987). The SLBQP problem also appears in the training methodology known as Support Vector Machine (SVM) (see Vapnik, 1982). In support vector machine problems, they formulate the problem as an optimization problem whose dual problem is an SLBQP problem. This dual problem is easier to solve compared to the primal problem. The matrix A in SVM is a symmetric and semi-positive definite matrix.

Projected gradient methods are appealing approaches for problem (1.1). In each iteration, it needs to decide the scaled projected gradient by projecting the point onto the feasible region (the feasible region is necessarily to be a closed convex set). The projection can be non-expensive if the feasible region is special enough. It can switch from one active set to another rapidly by taking a projected gradient step.

Recent projected gradient algorithms (Dai and Fletcher, 2005; Zanni, 2006) for solving this special problem made use of some extensions of the BB step-size (see Barzilai and Borwein, 1988). The GVPM algorithm presented by Zanni (2006) and Serafini *et al.* (2005) is a monotone algorithm using the two BB step-sizes alternatively to determine the scaled projected gradient d_k. Dai and Fletcher's algorithm uses a BB-type step-size to determine the scaled projected gradient d_k. They incorporated an adaptive non-monotone line search to allow the increase of the function value on some iteration in order for the method to work well. It seems that BB-type step-sizes are quite a good choice for solving the SLBQP problem by projected gradient method. This conclusion may also base on the observation of the good performance of a series of BB-type non-monotone projected gradient method for solving Bound Constrained Quadratic Problem (BQP) (Birgin *et al.*, 2000, 2003; Dai and Fletcher, 2005). A natural thought is to construct a projected gradient method with some other step-sizes (not necessarily to be non-monotone step-sizes) besides BB-type step-sizes. This is our motivation of this paper. Recent years, some monotone projected gradient methods are developed to solve the BQP problem (Zhou *et al.*, 2006b). Here we introduce the adaptive steepest descent (ASD) step-size (Zhou *et al.*, 2006a) and the Dai-Yuan (DY) step-size. The projected gradient methods which are based on these two step-sizes perform quite well in solving the BQP problem according to their experiment results.

Another consideration is the projection sub-problem. It appears that the framework of the sub-problem put forward by Dai and Fletcher is quite efficient and widely-used. They consider the Lagrange function with regard to the single linear equality constraint as the objective function and establish the delicate relationship between the Karush-Kuhn-Tucker condition of the original sub-problem and the modified one. Finally, this sub-problem was formulated as a problem of finding the zero point of a piecewise linear non-decrease continuous function with scalar variable

λ which is the lagrangian multiplier in the modified problem. In this paper we give a trustful initialization of λ_0. This initialization can also be used in other similar projection sub-problems. The numerical results showed that this adaptive initial guess of λ_0 is very close to the optimal Lagrangian multiplier after several iterations. Our numerical experiments demonstrate that this modification saves quite a lot of the computational cost compared to the original algorithm.

This paper is outlined as follows. In Sec. 2, we give an improved initial guess of λ_0 in projection sub-problem and its theoretical property. In Sec. 3, we introduce the adaptive steepest descent (ASD) step-size and the projected gradient method induced by it. In Sec. 4, we show the numerical experiments and the results. The paper is concluded in Sec. 5.

2. The Projection Subproblem

2.1. *The projection subproblem*

We denote P as the projection operator onto Ω,

$$P(x) = \arg\min_{y\in\Omega} \|x - y\|_2. \tag{2.1}$$

Here $\Omega = \{x : l \le x \le u, a^T x = b\}$ is the feasible region of this problem. To project a point $\bar{x}$ onto the feasible set Ω is equivalent to solve the optimization sub-problem:

$$\begin{aligned} \min_x \quad & \frac{1}{2}\|x - \bar{x}\|_2^2, \\ s.t. \quad & l \le x \le u, \\ & a^T x = b. \end{aligned} \tag{2.2}$$

This is also an SLBQP problem in which A is an identity matrix. Helgason *et al.* (1980) proposed an $O(n \log n)$ algorithm for solving the special problem based on appropriate manipulation of the corresponding Kuhn-Tucker conditions. Brucker (1984), Calamai and Moré (1987) proposed algorithms based on binary search. Pardalos and Kovoor (1990) proposed a randomized algorithm that runs in expected linear time. A more efficient algorithm based on the secant approximation is proposed by Dai and Fletcher (2005). We use their framework in this paper.

They first construct the Lagrange penalty function,

$$\Phi(x, \lambda) := \frac{1}{2}x^T x - \bar{x}^T x - \lambda(a^T x - b), \tag{2.3}$$

Fixing the Lagrangian multiplier λ, we can easily get the optimal solution of the problem

$$\begin{aligned} \min_x \quad & \Phi(x, \lambda), \\ s.t. \quad & l \le x \le u. \end{aligned} \tag{2.4}$$

We use $x(\bar{x}, \lambda)$ to denote the optimal solution of (2.4)

$$x(\bar{x}, \lambda) = \arg \min_{l \le x \le u} \Phi(x, \lambda), \tag{2.5}$$

and $r(\bar{x}, \lambda)$ to denote the residue of the equality constraint

$$r(\bar{x}, \lambda) = a^T x(\bar{x}, \lambda) - b. \tag{2.6}$$

If some λ^* satisfies $r(\bar{x}, \lambda^*) = 0$, then this λ^* is the optimal Lagrangian multiplier of the Karush-Kuhn-Tucker system of problem (2.1) and $x(\bar{x}, \lambda^*)$ is the projection point of $\bar{x}$. We denote the set of Lagrangian multiplier in the projection subproblem as $\Lambda(\bar{x})$.

The function $r(\bar{x}, \lambda)$ in (2.6) is a piecewise linear non-decrease continuous function of λ with breaking points $(l_i - c_i)/a_i$ and $(u_i - c_i)/a_i$. Dai and Fletcher constructed the algorithm based on this property of $r(\bar{x}, \lambda)$ (Calamai and Moré, 1987).

Their algorithm comprises two phases — Bracketing Phase and Secant Phase. The Bracketing Phase is to determine an interval $[\lambda_l, \lambda_u]$ which can contain a solution of the equation $r(\bar{x}, \lambda) = 0$. Since $r(\bar{x}, \lambda)$ is a non-decrease continuous function of λ, we set $\lambda_l := \lambda$ and search for the interval in the positive λ direction in the step-size of $\triangle\lambda$ if $r(\bar{x}, \lambda) < 0$; we set $\lambda_u := \lambda$ and search for the interval in the negative λ direction in the step-size of $\Delta\lambda$ if $r(\bar{x}, \lambda) > 0$. The Bracketing Phase terminates when we find the λ_l and λ_u that satisfies $r(\bar{x}, \lambda_l) < 0$ and $r(\bar{x}, \lambda_u) > 0$ or find a λ satisfies that $|r(\bar{x}, \lambda)|$ is sufficient small.

Once the interval is determined, the algorithm turns to the Secant Phase. The Secant phase is to find the solution of the equation $r(\bar{x}, \lambda) = 0$ by the repeated use of the secant method. If a value of $r(\lambda)$ is sufficiently close to zero, then the process terminates.

Initially values of λ_l and λ_u are available with $r(\lambda_l) < 0$ and $r(\lambda_u) > 0$, and a secant step to a new point λ is taken. Set $s =$ If $r(\lambda) > 0$ then the iteration proceeds as follows. If λ lies in the left half of the interval $[\lambda_l, \lambda_u]$, then a secant step based on λ_l and λ is taken on the next iteration. If λ lies in the right half of the interval, then either a secant step based on λ and λ_u, or a step to the point $\frac{3}{4}\lambda_l + \frac{1}{4}\lambda$ is taken, whichever is the smaller step. This ensures that the interval length is reduced by a factor of $\frac{3}{4}$ or less. In both cases λ_u is replaced by λ to give a new bracket. Similar decisions are taken if $r(\lambda) < 0$ at the start of the iteration.We terminate the secant phase if preset tolerances on either $r(\lambda)$ or the length of the interval $\triangle\lambda$ are met. The more detailed pseudo-code is shown in their paper (Dai and Fletcher, 2005).

2.2. *An improved subproblem initialization*

For deriving our improved initial estimate of λ, we first give the following proposition. Proposition 2.1 is an obvious deduction.

Proposition 2.1. *$\hat{x}$ is the projection point of $\bar{x}$ onto the feasible set of problem* (1.1) *if and only if there exists a $\bar{\lambda} \in \mathbf{R}$ such that*

$$\hat{x} = mid(l, u, \bar{x} + \bar{\lambda} a) \quad \textit{and} \quad a^T \hat{x} = b, \tag{2.7}$$

where $mid(l, u, h)$ is the componentwise operation that supplies the median of its three arguments.

Now we write the Karush-Kuhn-Tucker system of problem (1.1)

$$\begin{aligned} Ax - c - \alpha + \beta - \lambda a &= 0, \\ \alpha_i(x_i - l_i) &= 0, \quad i = 1, \ldots, n \\ \beta_i(u_i - x_i) &= 0, \quad i = 1, \ldots, n \\ \alpha_i, \beta_i &\geq 0, \quad i = 1, \ldots, n \\ l_i \leq x_i &\leq u_i, \quad i = 1, \ldots, n \\ a^T x &= b. \end{aligned} \tag{2.8}$$

Proposition 2.2. *If the feasible set Ω is a nonempty, closed and convex set and the first-order constraint qualification holds at x^*, then x^* is a first-order critical point of the problem if and only if there exist x^*, α^*, β^* and λ^* such that the Karush-Kuhn-Tucker system* (2.8) *is satisfied.*

Proposition 2.3. *If the feasible set Ω is a nonempty, closed and convex set and the first-order constraint qualification holds at x^*, then x^* is a first-order critical point of the problem if and only if*

$$P(x^* - tg(x^*)) = x^* \quad \text{for any } t \geq 0, \tag{2.9}$$

where $g(x)$ is the gradient of f at x.

Then we can get Proposition 2.4 based on the above statements.

Proposition 2.4. *Suppose that x^* is the first-order critical point of problem* (1.1), *$\bar{\lambda} \in \Lambda(x^* - tg(x^*))$ and $t > 0$, then $\bar{\lambda}/t$ is the lagrange multiplier of the KKT system of problem* (1.1).

On the other hand, suppose that (x^*, λ^*) satisfies the KKT system of problem (1.1) *and $t > 0$, then we have that $t\lambda^* \in \Lambda(x^* - tg(x^*))$.*

Proof. If x^* is the first-order critical point of the problem, then there exist x^*, α^*, β^* and λ^* such that the Karush-Kuhn-Tucker system (2.8) is satisfied, then

$$g(x^*) = \lambda^* a + \alpha^* - \beta^*, \tag{2.10}$$

$$x^* - tg(x^*) = x^* - t\lambda^* a + t\alpha^* - t\beta^*. \tag{2.11}$$

According to the complementary condition in the Karush-Kuhn-Tucker system and Proposition 2.1, we have

$$t\lambda^* \in \Lambda(x^* - tg(x^*)). \tag{2.12}$$

On the other hand, if x^* is the first-order critical point of the problem $\bar{\lambda} \in \Lambda(x^* - tg(x^*))$, then x^* must satisfy (2.9). We obtain that

$$x^* = \text{mid}(l, u, x^* - tg(x^*) + \bar{\lambda} a), \tag{2.13}$$

for any $t > 0$. Thus if $x_i^* = l_i$, we have

$$-tg_i(x^*) + \bar{\lambda}a_i \leq 0; \tag{2.14}$$

if $l_i < x_i^* < u_i$, we have

$$-tg_i(x^*) + \bar{\lambda}a_i = 0; \tag{2.15}$$

if $x_i^* = u_i$, we have

$$-tg_i(x^*) + \bar{\lambda}a_i \geq 0. \tag{2.16}$$

Consequently, if $x_i^* = l_i$, we can set

$$\alpha_i^* = g_i(x^*) - \frac{\bar{\lambda}}{t}a_i \geq 0,\ \beta_i^* = 0; \tag{2.17}$$

if $l_i < x_i^* < u_i$, we can set

$$\alpha_i^* = 0, \beta_i^* = 0; \tag{2.18}$$

if $x_i^* = u_i$, we can set

$$\alpha_i^* = 0,\ \beta_i^* = -g_i(x^*) + \frac{\bar{\lambda}}{t}a_i \geq 0. \tag{2.19}$$

Therefore $\bar{\lambda}/t$ is the lagrange multiplier of the KKT system of problem (1.1), which completes the proof. □

The numerical results show that the value of $\Lambda(x_k - \bar{\alpha}_k g(x_k))/\bar{\alpha}_k$ becomes a constant after several iterations and keeps unchanged until the algorithm terminates. We have a numerical example which is shown in Fig. 1. This is a problem of 1000 dimension and the condition number of matrix A is 100. The X-axis is the number of iteration and the Y-axis is the value of the ratio. We can observe that the ratio is in the neighborhood of the optimal lagrangian multiplier after several iterations and stays unchange until the algorithm terminates.

Since the algorithm can always converge to a first-order critical point of the problem (which will be discussed later), this constant must be the value of the optimal Lagrangian multiplier of problem (1.1).

Furthermore, after several iterations, the value of $\frac{\Lambda(x_k - \bar{\alpha}_k g(x_k))}{\bar{\alpha}_k}$ can be equal to the value of $\frac{\Lambda(x_{k+1} - \bar{\alpha}_{k+1} g(x_{k+1}))}{\bar{\alpha}_{k+1}}$ with quite a high probability. Based on the observation of the numerical results and Proposition 2.4, we can improve the initial estimate of the Lagrangian multiplier in the projection subproblem. Here we have a good initial guess of $\Lambda(x_{k+1} - \bar{\alpha}_{k+1} g(x_{k+1}))$:

$$\Lambda(x_k - \bar{\alpha}_k g(x_k))\bar{\alpha}_{k+1}/\bar{\alpha}_k.$$

This improved initial guess can also be used in other similar projected gradient methods subproblems. The numerical results showed that when using this initialization in GVPM and Dai-Fletcher methods, the computing time on projection subproblem can also be reduced quite a lot.

In Sec. 4, we will give the numerical results and show that how much computing time can be actually saved.

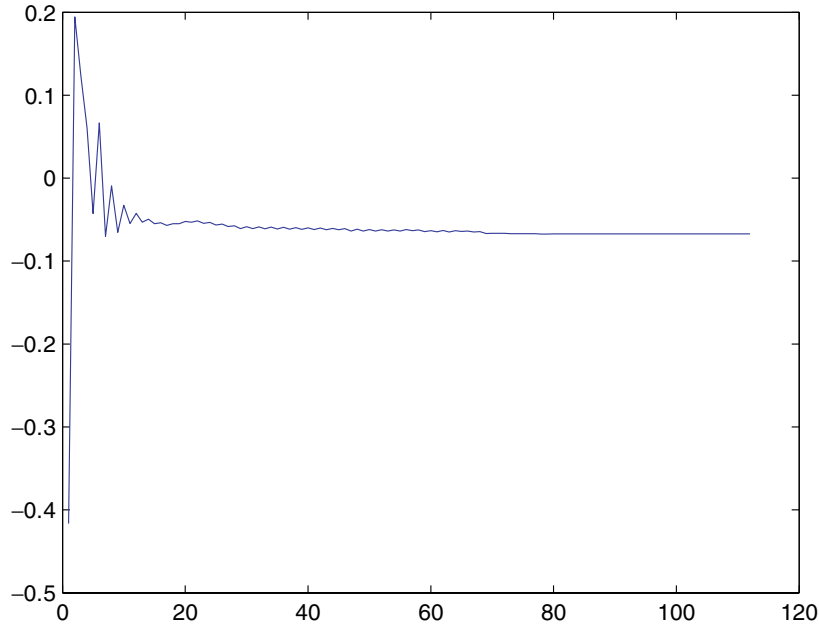

Fig. 1. An example.

3. Two Monotone Projected Gradient Methods

3.1. *Projected adaptive steepest descent method*

Here we introduce Adaptive Steepest Descent (ASD) step-size which is first presented by Zhou *et al.* (2006b). This method is a combination of steepest descent (SD) step-size and minimal gradient (MG) step-size. This step-size also is a monotone step-size. Let x_k be the current iterate point, and $g_k = g(x_k) = \nabla f(x_k)$ be the gradient at x_k.

$$\alpha_k^{ASD} = \begin{cases} \alpha_k^{MG}, & \text{if } \dfrac{\alpha_k^{MG}}{\alpha_k^{SD}} > \kappa. \\ \alpha_k^{SD} - \delta\alpha_k^{MG}, & \text{if } \dfrac{\alpha_k^{MG}}{\alpha_k^{SD}} \leq \kappa. \end{cases} \tag{3.1}$$

Here

$$\alpha_k^{MG} = \frac{g_k^T A g_k}{g_k^T A^2 g_k}, \quad \alpha_k^{SD} = \frac{g_k^T g_k}{g_k^T A g_k}. \tag{3.2}$$

It adaptively chooses either the MG step-size or the shortened SD step-size at each iteration. The MG step-size is used when the descent direction for the next iteration might be bad. The shortened SD step-size is used to guarantee a sufficient reduction.

They proved that the steepest descent method reaches its lowest convergent rate when $\frac{\alpha_k^{MG}}{\alpha_k^{SD}} > 0.5$. In this instance, using the MG step-size can obtain a g_{k+1} which is more inclined to some eigenvector than the choice of α_k^{SD}. On the other hand, the ratio $\frac{\alpha_k^{MG}}{\alpha_k^{SD}}$ reaches its minimal value in the event that MG step-size reaches its lowest convergent rate. This adaptive strategy may avoid the worst numerical performance of both of the step-sizes.

This method can get a very good performance when combined to the frame of the projected gradient method for solving Bound-constrained Quadratic Programming problem (BQP). In another paper (Zhou *et al.*, 2006b), they use this step-size in projected gradient method and name this method as the PASD method.

Algorithm 3.1.

Step 0. Initialization.

Step 1. If $\| P(x_k - g_k) - x_k \| < \varepsilon$, *then stop, else go to Step 2.*

Step 2. Set $u_k = P(x_k - g_k) - x_k$, *then compute* $\bar{\alpha}_k^{MG'} = \frac{u_k^T A u_k}{u_k^T A^2 u_k}$ *and* $\bar{\alpha}_k^{SD'} = -\frac{g_k^T u_k}{u_k^T A u_k}$

Step 3. Set

$$\bar{\alpha}_k^{ASD'} = \begin{cases} \bar{\alpha}_k^{MG'}, & \text{if } \dfrac{\bar{\alpha}_k^{MG'}}{\bar{\alpha}_k^{SD'}} > \kappa. \\ \bar{\alpha}_k^{SD'} - \delta\bar{\alpha}_k^{MG'}, & \text{if } \dfrac{\bar{\alpha}_k^{MG'}}{\bar{\alpha}_k^{SD'}} \le \kappa. \end{cases} \tag{3.3}$$

$$\bar{\alpha}_k = \max\{\alpha_{\min}, \min\{\alpha_{\max}, \bar{\alpha}_k^{ASD'}\}\}, \tag{3.4}$$

$$d_k = P(x_k - \bar{\alpha}_k g_k) - x_k. \tag{3.5}$$

Step 4. Do line search by quadratic iterpolation along the direction d_k *and determine* α_k *such that*

$$f(x_k + \alpha_k d_k) \le f(x_k) + \sigma\alpha_k g_k^T d_k, \tag{3.6}$$

then set $x_{k+1} = x_k + \alpha_k d_k$ *,* $k = k + 1$ *, go to Step 1.*

Here they use the direction u_k to calculate the step-size in the steepest descent direction $-g_k$. The reason why not to use $-g_k$ to compute the step-size is that the numerical performance can be quite bad in practice when using the step-size calculated by $-g_k$. And the performance improves quite a lot when using u_k to calculate $\bar{\alpha}_k^{ASD'}$. A reasonable explanation is that the angle between $-g_k$ and the generated d_k can be almost a right angle in most of the iterations, and the angle between u_k and d_k is closed to zero.

In their paper (Zhou *et al.*, 2006b), they compared the numerical results of PASD, projected Barzilai-Borwein (PBB) method without line search and projected Dai-Yuan (PDY) method (which will be discussed later). The numerical results

showed that the numerical performance of PASD method is comparable when solving the BQP problem. That is why we choose to solve SLBQP problem with this Adaptive Steepest Descent method.

3.2. *Projected Dai-Yuan method*

Projected Dai-Yuan method (PDY) is a projected gradient method with Dai-Yuan step-size.

$$\alpha_k^{DY} = \begin{cases} \alpha_k^{SD}, & \text{if } \mathrm{mod}(k,4) = 1 \text{ or } 2. \\ \alpha_k^{YV}, & \text{if } \mathrm{mod}(k,4) = 3 \text{ or } 4. \end{cases} \tag{3.7}$$

Here

$$\alpha_k^{SD} = \frac{g_k^T g_k}{g_k^T A g_k}, \alpha_k^{YV} = \frac{2}{((\frac{1}{\alpha_{k-1}^{SD}} - \frac{1}{\alpha_k^{SD}})^2 + \frac{4\|g_k\|_2^2}{(\alpha_{k-1}^{SD}\|g_{k-1}\|_2)^2})^{\frac{1}{2}} + \frac{1}{\alpha_{k-1}^{SD}} + \frac{1}{\alpha_k^{SD}}}. \tag{3.8}$$

Here the Dai-Yuan step-size is a modified version of the step-size which is first presented by Yuan (2006). The original method takes α_k^{SD} when k is odd and takes α_k^{YV} when k is even. It can find the exact solution within 3 iterations for two dimension convex problems.

Many modifications of this algorithm are discussed in Dai and Yuan (2005). They are all the different combinations of α_k^{SD}, α_k^{YV} and a modified step-size of α_k^{YV}. It is found that the choice above produces better numerical results than BB method. The algorithm that uses this modified version for solving BQP problem is first considered in Zhou *et al.* (2006b). The original method and its modified version are all monotone gradient methods.

Algorithm 3.2.

Step 0. Initialization.

Step 1. If $\| P(x_k - g_k) - x_k \| < \varepsilon$, *then stop, else go to Step 2.*

Step 2. Set $u_k = P(x_k - g_k) - x_k$, *then compute* $\alpha_k^{SD'} = \frac{-g_k^T u_k}{g_k^T A g_k}$ *and* $\alpha_k^{YV'} = \frac{2}{((\frac{1}{\alpha_{k-1}^{SD'}} - \frac{1}{\alpha_k^{SD'}})^2 + \frac{4\|g_k\|_2^2}{(\alpha_{k-1}^{SD'}\|g_{k-1}\|_2)^2})^{\frac{1}{2}} + \frac{1}{\alpha_{k-1}^{SD'}} + \frac{1}{\alpha_k^{SD'}}}$.

Step 3. Set

$$\bar{\alpha}_k^{DY'} = \begin{cases} \alpha_k^{SD'}, & \text{if } mod(k,4) = 1 \text{ or } 2. \\ \alpha_k^{YV'}, & \text{if } mod(k,4) = 3 \text{ or } 4. \end{cases} \tag{3.9}$$

$$\bar{\alpha}_k = \max\{\alpha_{\min}, \min\{\alpha_{\max}, \bar{\alpha}_k^{DY'}\}\}, \tag{3.10}$$

$$d_k = P(x_k - \bar{\alpha}_k g_k) - x_k. \tag{3.11}$$

Step 4. Do a line search by quadratic iterpolation along the direction d_k *and determine* α_k *such that*

$$f(x_k + \alpha_k d_k) \le f(x_k) + \sigma \alpha_k g_k^T d_k, \tag{3.12}$$

then set $x_{k+1} = x_k + \alpha_k d_k$, $k = k + 1$, *go to Step 1.*

3.3. *Convergence analysis*

In this subsection, we give the convergence analysis of the PASD method and the PDY method. Denote the scaled projected gradient

$$g_t(x) = P(x - tg(x)) - x, \tag{3.13}$$

where $x \in \Omega$, $g(x) = Ax - b$ and $t > 0$. Here we have that $d_k = g_{\bar{\alpha}_k}(x)$.

For deriving the convergence result of Algorithm 3.1, we have the following lemma about the property of the scaled projected gradient.

Lemma 3.1. *For all $x \in \Omega$ and $t \in (0, \alpha_{\max}]$, it holds that*

$$\langle g(x), g_t(x)\rangle \leq -\frac{\|g_t(x)\|_2^2}{t} \leq -\frac{\|g_t(x)\|_2^2}{\alpha_{\max}}. \tag{3.14}$$

Theorem 3.1. *Algorithm* 3.1 *and Algorithm* 3.2 *are well defined for the SLBQP problem, and any accumulation point of the sequence $\{x_k\}$ generated by it is a first-order critical point of the problem.*

Proof. By contradiction. If the conclusion does not hold, we denote $\hat{x}$ an accumulation point of $\{x_k\}$, then there must be a sub-sequence of $\{x_k\}$ converge to $\hat{x}$. For any $\alpha \in (0, \alpha_{\max}]$ we have

$$\|g_\alpha(\hat{x})\|_2 > 0. \tag{3.15}$$

On the other hand, if $\alpha_k = 1$ can not satisfy the Armijo Condition (3.6), it follows that

$$\alpha_k = -\frac{\langle g_k, d_k\rangle}{\langle d_k, Ad_k\rangle} \geq \frac{\|d_k\|_2^2}{\alpha_{\max}\langle d_k, Ad_k\rangle} \geq \frac{1}{\alpha_{\max} \parallel A \parallel_2}. \tag{3.16}$$

Set

$$\tau = \min\left\{1, \frac{1}{\alpha_{\max} \parallel A \parallel_2}\right\}. \tag{3.17}$$

Then for any $k \geq 1$, we have

$$\alpha_k \geq \tau. \tag{3.18}$$

From the continuity of $\|g_\alpha(\hat{x})\|_2$, there must be some $\gamma > 0$ such that for any $\alpha \in [\tau, \alpha_{\max}]$, we have

$$\|g_\alpha(\hat{x})\|_2 > \gamma. \tag{3.19}$$

Consequently, if k is big enough we have

$$\|g_{\alpha_k}(x_k)\|_2 > \gamma/2. \tag{3.20}$$

Combining (3.6) and Lemma 3.1 we have

$$f(x_{k+1}) \leq f(x_k) - \frac{\sigma\alpha_k\|g_{\alpha_{\max}}(x)\|_2^2}{\alpha_{\max}}. \tag{3.21}$$

Set $k \to \infty$, then $f(x_k) \to -\infty$ which contradicts that $f(x)$ is bounded below, which completes the proof. □

4. Numerical Experiments

In this section, the numerical experiments were done by using Matlab v 7.6 on Dell OptiPlex 755 (2.66 GHz, 1.96 GB of RAM), Windows XP.

Here we consider the Random SPD test problems. The generation of these problems is based on the generation of random SPD BQP test problems in Moré and Toraldo (1989) and Dai and Fletcher (2005a, 2005b). Here we generate the problem using five parameters – n, $ncond$, $ndeg$, $na(\bar{x})$, and $na(\bar{x}_1)$.

The generated A is a dense matrix. n is the size of the problem. 10^{ncond} is the condition number of matrix A. $ndeg$, $na(\bar{x})$, and $na(\bar{x}_1)$ determine the degenerate degree, the active set of the optimal point and the active set of the starting point respectively. The starting point can be generated by the generation of the problem. The tolerance ϵ in outer iteration is 10^{-5} as used in Calamai and Moré (1987). In projection subproblem, the accuracy required at the k-th iteration is $|r(\bar{x}, \lambda)| \leq 10^{-5}$ when $k = 1$ and $|r(\bar{x}, \lambda)| \leq 10^{-10}$ when $k \geq 2$.

First, we compare the numerical performance of projection subproblem algorithm with the improved initial guess of λ and the subproblem algorithm with the default initial guess of λ. The main calculation of the projection subproblem is to compute the residue $r(\overline{x}, \lambda)$. We compute a $r(\overline{x}, \lambda)$ when we change to a new λ.

Table 1 is the average number of how many λ is computed in 20 different random SLBQP problems when using three different projected gradient algorithms. After using the improved initial λ, the computational cost decreased at most 40% compared to the original projection sub-problem algorithm. It makes much more improvement when using this initialization in monotone algorithms. One possible reason is that the monotonicity of the step-sizes restricts the length of the step-sizes. This will cause fewer changes of the active set, hence make it easy to predict the new initial Lagrangian multiplier.

Table 1. Comparison in subproblem.

	n	$\kappa(A)$	Default λ_0	Improved λ_0	Saved (%)
Algorithm 3.1	1000	10	456.2	367.7	19.40
		100	1921.5	1617.7	15.81
	3000	10	469.9	374.6	20.28
		100	1810.9	1500.4	17.15
GVPM	1000	10	465.2	265.8	42.9
		100	1611.8	960.3	40.4
	3000	10	477.6	266.4	44.2
		100	1571.4	962.6	38.7
Dai-Fletcher	1000	10	427.9	382.2	10.7
		100	1639.6	1563.5	4.6
	3000	10	504.7	467.8	7.3
		100	1471.0	1387.2	5.7

Table 2. Comparison of iteration (computing time (s)).

		GVPM		Algorithm 3.1		Algorithm 3.2		Dai-Fletcher	
n	$\kappa(A)$	It.	Time	It.	Time	It.	Time	It.	Time
1000	10	19.9	0.086	22.5	0.164	20.6	0.127	18.1	0.061
	100	74.1	0.278	97.6	0.484	146.2	0.645	71.1	0.178
	1000	307.0	0.800	379.3	2.692	1203.1	4.708	276.0	0.627
3000	10	20.7	0.453	23.9	0.811	22.0	0.748	16.7	0.336
	100	75.4	1.466	108.9	3.869	178.8	6.342	70.9	1.331
	1000	304.6	5.255	874.5	29.077	1457.7	48.053	269.7	4.688
5000	10	20.6	1.116	24.1	2.228	25.4	2.348	16.0	0.828
	100	71.3	3.350	105.1	9.295	178.9	15.758	70.4	3.234
	1000	292.9	14.119	941.5	89.903	1562.8	148.808	265.4	12.902

Second, we compare the numerical performance of GVPM, Algorithm 3.1 and Algorithm 3.2 and Dai-Flether algorithm. Only the last algorithm is a non-monotone algorithm among the four. The tolerance ϵ in outer iteration is 10^{-3}.

The main computational cost in these four methods is the multiplication between matrix and vector. The matrix-vector multiplication appears at the calculation of the step-size, line-search and function evaluation. GVPM, Algorithm 3.1 and Algorithm 3.2 computes 1, 2 and 2 matrix-vector multiplications at each iteration respectively. Dai-Fletcher algorithm computes 1 matrix-vector multiplications at each iteration no matter whether the line search is done at the iteration. Table 2 compares the average value of iteration and computing time computed in 10 different random SLBQP problems.

Our numerical experiments demonstrate that the computing time is almost proportional to the product of the number of matrix-vector multiplication computed at each iteration and the number of iterations. GVPM and Dai-Fletcher algorithm have almost the same computational cost at each iteration. The numerical performance of Dai-Fletcher algorithm defeats the numerical performance of GVPM at all the experiment cases especially when the condition number is quite large. Algorithms 3.1 and Algorithms 3.2 have almost the same computational cost at each iteration. The numerical performance of Algorithms 3.1 defeats the numerical performance of Algorithms 3.2 at all the experiment cases except for the cases when the condition number is 10. Basically, Dai-Fletcher algorithm has the best numerical performance among the four algorithms. GVPM has the best numerical performance among the three monotone projected gradient algorithms. Both of Algorithm 3.1 and Algorithm 3.2 do not use BB-type step-size. They has more computational cost at each iterations than the BB-type methods. Another disadvantage maybe the number of iterations when the condition number is large.

5. Conclusion

In this paper, we present two monotone projected gradient algorithms for SLBQP problem and give their convergence analysis. We establish the relationship between

the Lagrangian multiplier in the projection subproblem and the Lagrangian multiplier in the original optimization problem at the first-order critical point. Numerical results showed that λ in the sub-problem converges to a constant after several iterations. Therefore we introduced a new estimate of the initial guess of λ_0 in the projection subproblem by this observation. The improvement can save at most 40% of the computing cost according to our numerical results. But we have not given the convergence result of the multiplier in the projection subproblem in theory yet.

We compare the numerical results of four projected gradient methods for solving random SPD SLBQP problems. The non-monotone algorithm Dai-Fletcher algorithm has the best numerical performance. Between the two new algorithms, Algorithm 3.1 has better numerical performance than Algorithm 3.2.

Acknowledgments

This work is supported by the Chinese NSF grants 10571171, 10831006 and the CAS grant kjcx-yw-s7-03.

References

Barzilai, J and JM Borwein (1988). Two-point step size gradient methods. *IMA J. Numer. Anal.*, 8, 141–148.

Birgin, EG, JM Martínez and M Raydan (2000). Nonmonotone spectral projected gradient methods on convex sets. *SIAM J. Optim.*, 10, 1196–1211.

Birgin, EG, JM Martínez and M Raydan (2003). Inexact spectral projected gradient methods on convex sets. *IMA J. Numer. Anal.*, 23, 539–559.

Brucker, P (1984). An $O(n)$ algorithm for quadratic knapsack problems. *Oper. Res. Lett.*, 3, 163–166.

Calamai, PH and JJ Moré (1987). Quasi-Newton updates with bounds. *SIAM J. Numer. Anal.*, 24, 1434–1441.

Dai, YH and R Fletcher (2005a). New algorithms for singly linearly constrained quadratic programs subject to lower and upper bounds. *Math. Prog.*, 103(3), 403–421.

Dai, YH and R Fletcher (2005b). Projected Barzilai-Borwein methods for large-scale box-constrained quadratic programming. *Numerische Mathematik A*, 100(1), 21–47.

Dai, YH and YX Yuan (2005). Analysis of monotone methods. *Journay of Industy and Management Optimization*, 1(2), 181–192.

Held, M, P Wolfe and H Crowder (1974). Validation of subgradient algorithms, *Math. Prog.*, 6, 62–88.

Helgason, R, J Kennington and H Lall (1980). A polynomially bound algorithms for a singly constrained quadratic program. *Math. Prog.* 18, 338–343.

Moré, J and G Toraldo (1989). Algorithms for bound constrained quadratic programming problems. *Numer. Math.*, 55, 377–400.

Meyer, RR (1984). Multipoint methods for separable nonlinear networks. *Math. Programming Study*, 22, 185–205.

Pardalos, PM and JB Rosen (1987). Constrained global optimization: algorithms and applications. In *Lecture Notes in Computer Science*. Berlin: Springer, p. 268.

Pardalos, PM and N Kovoor (1990). An algorithm for a singly constrained class of quadratic programs subject to upper and lower bounds. *Math. Prog.*, 46, 321–328.

Serafini, T, GV Zanghirati and L Zanni (2005). Gradient projection methods for quadratic programs and applications in training support vector machines. *Optimization Methods and Software*, 20(2–3), 353–378.

Vapnik, V (1982). *Estimation of Dependences Based on Empirical Data*, Springer-Verlag.

Yuan, YX (2006). A new stepsize for steepest descent method. *Journal of Computational Mathematics*, 24(2), 149–156.

Zhou, B, L Gao and YH Dai (2006a). Gradient methods with adaptive step-sizes. *Computational Optimization and Applications*, 35(1), 69–86.

Zhou, B, L Gao and YH Dai (2006b). Monotone projected gradient methods for large-scale box-constrained quadratic programming. *Science in China Series A*, 49(5), 688–702.

Zanni, L (2006). An improved gradient projection-based decomposition technique for support vector machines. *Computational Management Science*, 3(2), 131–145.

Yun-Shan Fu received her B.Sc degree in Computational Mathematics from the University of Jilin (China) in 2005. She is currently reading for a Ph.D. degree in the Institute of Computational Mathematics and Scientific/Engineering Computing of the Chinese Academy of Sciences. Her research interests are mainly in nonlinear optimization and combinatorial optimization.

Yu-Hong Dai received his B.Sc degree in Applied Mathematics from Beijing Institute of Technology in 1992 and Ph.D. degree in Computatonal Mathematics from Institute of Computational Mathematics and Scientific/Engineering Computating of the Chinese Academy of Sciences in 1997. He is a Professor at the Academy of Mathematics and Systems Science, Chinese Academy of Sciences. His major research interest is theory and numerical method for nonlinear programming. Recently, he also studies some special optimization probroms arising from several practical fields.