

GRADIENT PROJECTION METHODS FOR QUADRATIC PROGRAMS AND APPLICATIONS IN TRAINING SUPPORT VECTOR MACHINES*

THOMAS SERAFINI^{1,†} GAETANO ZANGHIRATI^{2,‡} LUCA ZANNI^{1,§}

¹*Dipartimento di Matematica, Università di Modena e Reggio Emilia, via Campi 213/b, 41100 Modena, Italy;*

²*Dipartimento di Matematica, Università di Ferrara, via Machiavelli 35, 44100 Ferrara, Italy.*

Gradient projection methods based on the Barzilai-Borwein spectral steplength choices are considered for quadratic programming problems with simple constraints. Well-known nonmonotone spectral projected gradient methods and variable projection methods are discussed. For both approaches the behavior of different combinations of the two spectral steplengths is investigated. A new adaptive steplength alternating rule is proposed that becomes the basis for a generalized version of the variable projection method (GVPM). Convergence results are given for the proposed approach and its effectiveness is shown by means of an extensive computational study on several test problems, including the special quadratic programs arising in training support vector machines. Finally, the GVPM behavior as inner QP solver in decomposition techniques for large-scale support vector machines is also evaluated.

Keywords: quadratic programs, gradient projection methods, support vector machines, decomposition techniques, large-scale problems.

1 Introduction

The aim of this work is to analyse gradient projection methods for minimizing quadratic functions on nonempty closed convex sets defined by simple constraints. Particular interest is devoted to a special problem of this type: the convex quadratic programming (QP) problem with box constraints and a single linear equality constraint arising in training the learning methodology named *support vector machine* (SVM).

Gradient projection methods appear promising approaches for the above problems since they are based on successive projections on the feasible region, which are nonexpensive operations when the constraints are simple. Furthermore, their low memory requirements and extreme simplicity make them attractive for large-scale problems, both in scalar and parallel environments. On the other hand, it is well known that these methods may exhibit very slow convergence if not combined with appropriate steplength selections.

Here we deal with gradient projection methods that exploit the two spectral steplengths introduced by Barzilai and Borwein in [1] for the unconstrained case. We consider a *nonmonotone spectral projected gradient method* (SPGM) developed in [3] and the *variable projection methods* (VPMs) introduced in [33, 34]. Even if the two approaches can exploit the same steplength selections and can be described within the same gradient projection scheme, they present considerable differences. In fact, the method in [3] uses a nonmonotone linesearch

*This work was supported by the Italian Education, University and Research Ministry (grants FIRB2001/RBAU01JYPN and FIRB2001/RBAU01877P).

[†]E-mail: serafini.thomas@unimo.it

[‡]E-mail: g.zanghirati@unife.it

[§]E-mail: zanni.luca@unimo.it

technique [20], while the variable projection methods use a limited minimization rule as (monotone) linesearch procedure [2]. For both schemes we are interested in investigating the improvements arising from some kinds of alternation between the two BB rules. This analysis is motivated by the promising results recently obtained in [18, 19, 21, 38] with different alternating strategies. In the nonmonotone gradient method for unconstrained problems described in [21], an alternation at each iteration between the two BB rules is suggested, while in the variable projection method for SVM QP problems discussed in [18, 19, 38], an alternation every three iterations is recommended. In this paper, we verify the effectiveness of these alternating strategies for both SPGM and VPM; moreover, we introduce a generalized VPM version based on an adaptive alternating strategy for the steplength selection. For the generalized scheme, which includes VPMs as special cases, we give both the convergence analysis and the numerical evidence of its promising performance. We evaluate the behavior of the considered approaches on several test problems and, in particular, on the QP problem arising in training standard SVMs (see [17] and the references therein for optimization problems in nonstandard SVMs).

Since we are specially interested in the latter problem, we briefly recall its main features [4, 7, 8, 37]. Given a training set of labelled examples

$$D = \{(\mathbf{z}_i, y_i), i = 1, \dots, n, \quad \mathbf{z}_i \in \mathbb{R}^m, y_i \in \{-1, 1\}\},$$

the SVM algorithm performs classification of new examples $\mathbf{z} \in \mathbb{R}^m$ by using a decision function $F : \mathbb{R}^m \rightarrow \{-1, 1\}$, of the form

$$F(\mathbf{z}) = \text{sign} \left(\sum_{i=1}^n x_i^* y_i K(\mathbf{z}, \mathbf{z}_i) + b^* \right), \quad (1)$$

in which $K : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}$ denotes a special kernel function and $\mathbf{x}^* = (x_1^*, \dots, x_n^*)^T$ is the solution of

$$\begin{aligned} \min \quad & \mathcal{F}(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T G \mathbf{x} - \sum_{i=1}^n x_i \\ \text{sub. to} \quad & \sum_{i=1}^n y_i x_i = 0, \\ & 0 \leq x_j \leq C, \quad j = 1, \dots, n, \end{aligned} \quad (2)$$

where G has entries $G_{ij} = y_i y_j K(\mathbf{z}_i, \mathbf{z}_j)$, $i, j = 1, 2, \dots, n$, and C is a parameter of the SVM algorithm. Once the vector \mathbf{x}^* is computed, the quantity $b^* \in \mathbb{R}$ in (1) is easily derived. A training example \mathbf{z}_i is called *support vector* (SV) if the corresponding x_i^* is nonzero and *bound support vector* (BSV) if $x_i^* = C$. Widely-used kernel functions are the linear kernel ($K(\mathbf{z}_i, \mathbf{z}_j) = \mathbf{z}_i^T \mathbf{z}_j$), the polynomial kernel ($K(\mathbf{z}_i, \mathbf{z}_j) = (1 + \mathbf{z}_i^T \mathbf{z}_j)^d$, $d \in \mathbb{N}$) and the Gaussian kernel ($K(\mathbf{z}_i, \mathbf{z}_j) = \exp(-\|\mathbf{z}_i - \mathbf{z}_j\|_2^2 / (2\sigma^2))$, $\sigma \in \mathbb{R}$). Since the matrix G corresponding to these kernels is in general dense, and in many real-world applications its size is very large ($n \gg 10^4$), the training phase of an SVM leads to a challenging QP problem. In fact, standard QP solvers based on explicit storage of G cannot be used and strategies that exploit the special features of the problem are absolutely necessary. Of these strategies, decomposition techniques have been most widely investigated (see [12] for a different approach in linear SVMs). They consist in splitting the large problem (2) into a sequence of smaller QP subproblems that can fit into the available memory. The packages in [5, 14, 31] are designed for subproblems of size 2 which are analytically solvable, while the techniques in [6, 22, 23, 29, 38] have the subproblem size as a parameter and

they need a numerical QP solver. In these cases, a crucial question is what kind of QP solver is most convenient. The numerical experiments of this work show that the gradient projection methods considered are very effective inner solvers for decomposition techniques and may be useful to improve the performance of existing packages or to design new decomposition schemes.

The paper is organized as follows: section 2 states the considered gradient projection approaches and the convergence analysis for the new generalized VPM, section 3 presents a numerical comparison of the methods, section 4 shows their effectiveness as inner solvers for SVMs decomposition techniques and, finally, section 5 draws some conclusions.

2 Two Gradient Projection Approaches for Quadratic Programs

We consider the numerical solution of the quadratic program

$$\min_{\mathbf{x} \in \Omega} f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T G \mathbf{x} + \mathbf{q}^T \mathbf{x} \quad (3)$$

where G is an $n \times n$ symmetric matrix, $\mathbf{q}, \mathbf{x} \in \mathbb{R}^n$ and $\Omega \subset \mathbb{R}^n$ is a nonempty closed convex set defined by simple constraints. Throughout the paper, $P_\Omega(\cdot)$ denotes the orthogonal projection on Ω . We are interested in solving this problem by gradient projection methods that fall into the following general scheme:

ALGORITHM GPM (Gradient Projection Methods)

STEP 1. *Initialization.* Let $\mathbf{x}^{(0)} \in \Omega$, $0 < \alpha_{\min} < \alpha_{\max}$, $\alpha_0 \in [\alpha_{\min}, \alpha_{\max}]$; set $k = 0$.

STEP 2. *Projection.* Terminate if $\mathbf{x}^{(k)}$ satisfies a stopping criterion; otherwise compute the direction

$$\mathbf{d}^{(k)} = P_\Omega(\mathbf{x}^{(k)} - \alpha_k(G\mathbf{x}^{(k)} + \mathbf{q})) - \mathbf{x}^{(k)}. \quad (4)$$

STEP 3. *Linesearch.* Compute

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \lambda_k \mathbf{d}^{(k)}, \quad \lambda_k \in (0, 1],$$

where λ_k is a stepsize determined by a linesearch procedure.

STEP 4. *Updating.* Compute $\alpha_{k+1} \in [\alpha_{\min}, \alpha_{\max}]$, set $k \leftarrow k + 1$ and go to step 2.

We analyse gradient projection methods that use an α_{k+1} steplength selection based on the Barzilai-Borwein (BB) spectral rules:

$$\alpha_{k+1}^{(1)} = \frac{(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)})^T (\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)})}{(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)})^T G (\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)})} = \frac{\mathbf{d}^{(k)T} \mathbf{d}^{(k)}}{\mathbf{d}^{(k)T} G \mathbf{d}^{(k)}}, \quad (5)$$

$$\alpha_{k+1}^{(2)} = \frac{(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)})^T G (\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)})}{(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)})^T G^2 (\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)})} = \frac{\mathbf{d}^{(k)T} G \mathbf{d}^{(k)}}{\mathbf{d}^{(k)T} G^2 \mathbf{d}^{(k)}}. \quad (6)$$

In particular we concentrate on the nonmonotone spectral projected gradient method introduced in [3] and on the variable projection methods proposed in [18, 33, 34]. The first approach combines the updating rule (5) with the nonmonotone linesearch procedure developed in [20], while the variable projection methods use a classical limited minimization rule as linesearch procedure and either (5) or (6). Before stating in detail the two approaches and discussing the behavior of the steplength selections, we introduce the following property:

LEMMA 2.1 Let G be an $n \times n$ symmetric matrix and $\mathbf{d} \in \mathbb{R}^n$. If $\mathbf{d}^T G \mathbf{d} > 0$ then

$$\frac{\mathbf{d}^T G \mathbf{d}}{\mathbf{d}^T G^2 \mathbf{d}} \leq \frac{\mathbf{d}^T \mathbf{d}}{\mathbf{d}^T G \mathbf{d}}. \quad (7)$$

Proof By Cauchy-Schwartz inequality we have

$$\mathbf{d}^T G \mathbf{d} \leq \sqrt{\mathbf{d}^T \mathbf{d}} \sqrt{(G \mathbf{d})^T (G \mathbf{d})} = \sqrt{\mathbf{d}^T \mathbf{d}} \sqrt{\mathbf{d}^T G^2 \mathbf{d}}$$

and we may obtain (7) by squaring and dividing by $(\mathbf{d}^T G \mathbf{d})(\mathbf{d}^T G^2 \mathbf{d})$. \square

The previous Lemma ensures that, at each iteration of a gradient projection scheme, if $\mathbf{d}^{(k)T} G \mathbf{d}^{(k)} > 0$ then the steplength candidate values given by rules (5) and (6) satisfy

$$\alpha_{k+1}^{(2)} \leq \alpha_{k+1}^{(1)}.$$

In the following we present the two gradient projection approaches and in the next section we analyse their behavior on several test problems.

2.1 The Nonmonotone Spectral Projected Gradient Method

We recall the SPGM introduced in [3, alg. SPG2] for the minimization of differentiable functions on nonempty closed and convex sets. When SPGM is applied to solve the QP problem (3), it may be stated this way:

ALGORITHM SPGM (Spectral Projected Gradient Method)

STEP 1. *Initialization.* Let $\mathbf{x}^{(0)} \in \Omega$, $M \geq 1$, $0 < \sigma_1 < \sigma_2 < 1$, $\gamma \in (0, 1)$, $0 < \alpha_{\min} < \alpha_{\max}$, $\alpha_0 \in [\alpha_{\min}, \alpha_{\max}]$; set $k = 0$.

STEP 2. *Projection.* Terminate if $\mathbf{x}^{(k)}$ satisfies a stopping criterion; otherwise compute the descent direction

$$\mathbf{d}^{(k)} = P_{\Omega}(\mathbf{x}^{(k)} - \alpha_k (G \mathbf{x}^{(k)} + \mathbf{q})) - \mathbf{x}^{(k)}.$$

STEP 3. *Linesearch.* Compute

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \lambda_k \mathbf{d}^{(k)}, \quad \lambda_k \in (0, 1],$$

with λ_k given by the following linesearch procedure:

STEP 3.1. Set $\lambda = 1$, $T = (G \mathbf{x}^{(k)} + \mathbf{q})^T \mathbf{d}^{(k)}$ and $f_M = \max_{0 \leq j \leq \min\{k, M-1\}} f(\mathbf{x}^{(k-j)})$.

STEP 3.2. If $f(\mathbf{x}^{(k)} + \lambda \mathbf{d}^{(k)}) \leq f_M + \gamma \lambda T$ then

set $\lambda_k = \lambda$;

else

define $\lambda_{\text{new}} \in [\sigma_1 \lambda, \sigma_2 \lambda]$, set $\lambda = \lambda_{\text{new}}$ and go to step 3.2;

end.

STEP 4. *Updating.* If $\mathbf{d}^{(k)T} G \mathbf{d}^{(k)} \leq 0$ then

set $\alpha_{k+1} = \alpha_{\max}$;

else

compute $\alpha_{k+1}^{(1)}$ as in (5) and set $\alpha_{k+1} = \min \left\{ \alpha_{\max}, \max \left\{ \alpha_{\min}, \alpha_{k+1}^{(1)} \right\} \right\}$;

end.

Set $k \leftarrow k + 1$ and go to step 2.

TABLE I SPGM versus VPM comparison on the HARKERP2 test problem.

method	it	ls	time	f_{\min}
SPGM($\alpha^{(1)}$)	33	3	0.10	-5.000E-1
SPGM($\alpha^{(2)}$)	4340	0	12.19	-5.000E-1
VPM($\alpha^{(1)}$)	119	34	0.36	-5.000E-1
VPM($\alpha^{(2)}$)	3706	1	12.06	-5.000E-1

From the computational viewpoint, the main tasks of each iteration may be considered the projection on Ω in Step 2 and the matrix-vector product $\mathbf{w} = G\mathbf{d}^{(k)}$ first involved in the function evaluation of Step 3.2, excluding the first iteration which computes also $G\mathbf{x}^{(0)}$. In fact, suppose $\mathbf{t} = G\mathbf{x}^{(k)}$ be already computed, then all the other quantities can be obtained by vector-vector operations from \mathbf{w} and/or \mathbf{t} (the stopping rule and the λ_{new} selection will be discussed later). At the end of the iteration, \mathbf{t} itself is simply updated by $\mathbf{t} \leftarrow \mathbf{t} + \lambda_k \mathbf{w} = G\mathbf{x}^{(k+1)}$. Of course, the computational cost of these main tasks can be largely reduced for special forms of Ω and/or G .

The convergence analysis developed in [3] ensures that any accumulation point of the sequence $\{\mathbf{x}^{(k)}\}$ generated by SPGM is a constrained stationary point. From the performance viewpoint, the spectral steplength (5), coupled with a nonmonotone linesearch strategy that accepts the corresponding iterate as frequently as possible, is presented in [3] as a successful idea to accelerate the convergence rate; its efficiency is then shown by testing SPGM against the LANCELOT package on many test problems from the CUTE collection. In [3] it is also observed that the convergence rate of the classical gradient projection method with linesearch along the projection arc, both in monotone and nonmonotone versions, is very slow compared to SPGM. This confirms that the spectral steplength (5) is an essential feature for accelerating nonmonotone gradient projection schemes.

Concerning the behavior of rule (6) in SPGM, no considerations or numerical results are given in [3]. On the other hand, in the context of gradient methods for unconstrained problems, in [11] it is just claimed that (6) often performs worse than (5) in the BB method for quadratic functions. For these reasons a wide numerical experimentation on the behavior of both rules within SPGM is reported in section 3.

Here, for the purpose of illustration only and to facilitate the comparison with VPMs, we consider how SPGM performs on the CUTE HARKERP2 test problem, a convex quadratic program sized 100 subject to nonnegativity constraints. In what follows, we denote by SPGM($\alpha^{(i)}$), $i = 1, 2$, the SPGM variant using $\alpha_{k+1}^{(i)}$, $i = 1, 2$.

In Table I we report the number of iterations (it), the number of linesearches producing $\lambda_k < 1$ (ls), the time in seconds and the best computed function value (f_{\min}). The experiments are carried out in MATLAB on a Digital Alpha personal workstation 500au. The upper part of Table I corresponds to the two SPGM schemes just described, where the stopping rule is $\|P_{\Omega}(\mathbf{x}^{(k)} - (G\mathbf{x}^{(k)} + \mathbf{q})) - \mathbf{x}^{(k)}\|_{\infty} < 10^{-5}$ and the parameter settings are those suggested in [3] ($\alpha_{\min} = 10^{-30}$, $\alpha_{\max} = 10^{30}$, $\alpha_0 = \|P_{\Omega}(\mathbf{x}^{(0)} - (G\mathbf{x}^{(0)} + \mathbf{q})) - \mathbf{x}^{(0)}\|_{\infty}^{-1}$, $M = 10$, $\gamma = 10^{-4}$, $\sigma_1 = 0.1$, $\sigma_2 = 0.9$). In particular, we use the same linesearch procedure as in [3]: compute

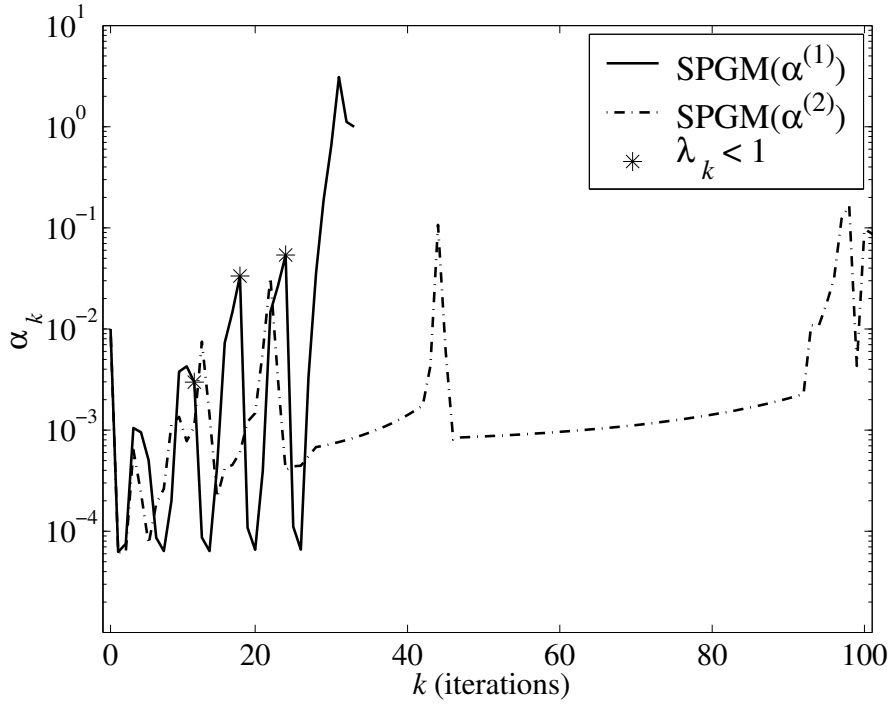


FIGURE 1 Values of α_k in $\text{SPGM}(\alpha^{(1)})$ and in $\text{SPGM}(\alpha^{(2)})$ for HARKERP2 test problem.

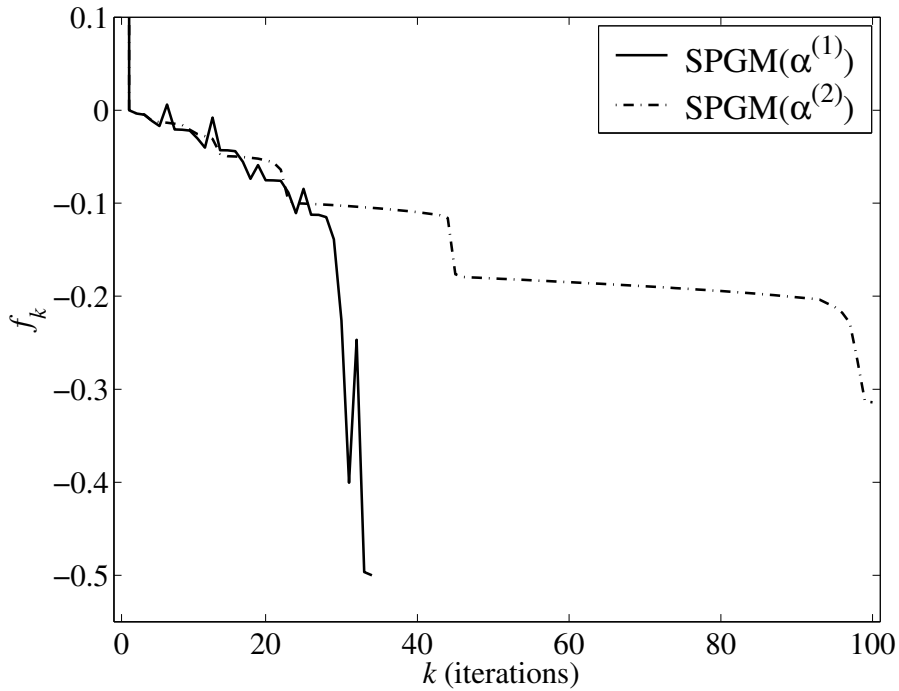


FIGURE 2 Values of $f_k = f(\mathbf{x}_k)$ in $\text{SPGM}(\alpha^{(1)})$ and in $\text{SPGM}(\alpha^{(2)})$ for HARKERP2 test problem.

$\lambda_{\text{new}} = \arg \min_{\lambda \in [0,1]} f(\mathbf{x}^{(k)} + \lambda \mathbf{d}^{(k)})$, if $\lambda_{\text{new}} \in [\sigma_1, \sigma_2 \lambda]$ then $\lambda \leftarrow \lambda_{\text{new}}$ else $\lambda \leftarrow \lambda/2$. Figures 1 and 2 show the values of α_k and $f(\mathbf{x}^{(k)})$ in the first iterations of the two SPGM methods. In Figure 1 the symbol “*” denotes iterations where the linesearch procedure implies a step reduction.

We can observe the considerable differences that the two steplength selection rules produce. In particular,

rule (5) gives the typical nonmonotone behavior of the sequence $\{f(\mathbf{x}_k)\}$, which is also a characteristic property of the BB method in unconstrained optimization (see e.g. [15]). In fact, after some iterations where $f(\mathbf{x}^{(k)})$ increases, a remarkable objective function descent follows, up to around iteration 30 where this behavior is well emphasized. Of course, in order to exploit the effectiveness of the steplength rule (5), a linesearch procedure able to accept nonmonotone steps is crucial. In this sense, the combination of nonmonotone linesearch and rule (5) is considered a successful strategy in [3]. On the other hand, rule (6) seems to be unable to produce nonmonotonicity in the sequence $\{f(\mathbf{x}_k)\}$ and implies a very slow convergence. However, as it is shown in section 3, this is not always the case and sometimes SPGM($\alpha^{(2)}$) outperforms SPGM($\alpha^{(1)}$). These different behaviors highlight the high sensitivity of SPGM to steplength selection and suggest that it is worth finding out whether improvements could be obtained by alternating between the two BB rules, as seems to be the case in unconstrained optimization [21] (see also [9, 10, 16, 32] for suggestions about other choices of the steplength). We compare some alternating strategies numerically in section 3.

2.2 The Generalized Variable Projection Method

In this subsection we propose a generalized version of the variable projection methods introduced for convex quadratic programs in [18, 33, 34]. These schemes are scaled gradient projection methods that use a limited minimization rule as linesearch procedure (see e.g. [2]) and the BB rule (5) or (6) for steplength selection. For the general nonconvex QP problem (3), a description of the unscaled VPM steps is given in Algorithm VPM.

ALGORITHM VPM (Variable Projection Method)

STEP 1. *Initialization.* Let $\mathbf{x}^{(0)} \in \Omega$, $i_\alpha \in \{1, 2\}$, $0 < \alpha_{\min} < \alpha_{\max}$, $\alpha_0 \in [\alpha_{\min}, \alpha_{\max}]$; set $k = 0$.

STEP 2. *Projection.* Terminate if $\mathbf{x}^{(k)}$ satisfies a stopping criterion; otherwise compute the descent direction

$$\mathbf{d}^{(k)} = P_\Omega(\mathbf{x}^{(k)} - \alpha_k(G\mathbf{x}^{(k)} + \mathbf{q})) - \mathbf{x}^{(k)}.$$

STEP 3. *Linesearch.* Compute

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \lambda_k \mathbf{d}^{(k)}, \quad \lambda_k \in (0, 1],$$

with λ_k given by

$$\lambda_k = \arg \min_{\lambda \in [0, 1]} f(\mathbf{x}^{(k)} + \lambda \mathbf{d}^{(k)}). \quad (8)$$

STEP 4. *Updating.* If $\mathbf{d}^{(k)T} G \mathbf{d}^{(k)} \leq 0$ then

set $\alpha_{k+1} = \alpha_{\max}$;

else

compute $\alpha_{k+1}^{(i_\alpha)}$ and set $\alpha_{k+1} = \min \left\{ \alpha_{\max}, \max \left\{ \alpha_{\min}, \alpha_{k+1}^{(i_\alpha)} \right\} \right\}$.

end.

Set $k \leftarrow k + 1$ and go to step 2.

The same considerations on the computational cost given for SPGM still hold for VPMs. The VPM convergence properties are discussed in the next subsection where a convergence analysis of its generalized version is presented.

From the performance viewpoint, numerical evidence of their greater effectiveness by comparison with clas-

sical gradient projection methods (with constant steplength and limited minimization rule, or with Armijo-type rule along the projection arc) are reported in [34]. See also [18] for an overview on VPMs for large convex quadratic programs and for a description of how the steplength selections (5) and (6) may be derived by heuristic arguments on $f(\mathbf{x}^{(k)} + \mathbf{d}^{(k)})$ and $\|\mathbf{x}^* - (\mathbf{x}^{(k)} + \mathbf{d}^{(k)})\|_2$ bounds, respectively. We also note that rule (6) may be considered a special case of a general steplength selection rule proposed in [13] for projection methods in variational inequality problems, which was obtained by similar heuristic arguments.

Now, to better understand the generalized version of these methods, we show the numerical results obtained by solving the HARKERP2 test problem with VPMs. We denote by $\text{VPM}(\alpha^{(1)})$ the version of VPM that uses the steplength selection (5) ($i_\alpha = 1$) and by $\text{VPM}(\alpha^{(2)})$ the version that uses the rule (6) ($i_\alpha = 2$). The results in the bottom half of Table I, as well as Figures 3 and 4, are obtained with $\mathbf{x}^{(0)}$, α_0 , α_{\min} , α_{\max} and stopping rule, chosen as in the previous experiments with SPGM.

The behavior of $\text{VPM}(\alpha^{(2)})$ is very similar to that of $\text{SPGM}(\alpha^{(2)})$. In fact, in spite of the stronger request on λ_k given by the limited minimization rule (8), also in this case rule (6) produces steps that do not need to be reduced by the linesearch procedure (indeed $\lambda_k < 1$ for $k = 0$ only). As for $\text{SPGM}(\alpha^{(2)})$, this steplength seems to imply too slow a descent of the objective function. On the other hand, in $\text{VPM}(\alpha^{(1)})$ the poor performance appears to be generated by an opposite phenomenon: here, the limited minimization rule implies too many linesearch reductions and consequently overly frequent rejections of the iterate corresponding to the steplength (5). This fact does not prevent the method from producing iterations with a large objective function descent, but it delays these iterations and reduces the descent magnitude, thus degrading the $\text{VPM}(\alpha^{(1)})$ performance with respect to $\text{SPGM}(\alpha^{(1)})$.

Since these drawbacks are frequently observed in VPM applications (see the experiments in section 3), we argue that, even if the steplength selections (5) and (6) allow VPMs to outperform other classical gradient projection methods, their combination with the linesearch procedure (8) cannot be considered completely satisfactory.

In what follows, we suggest a generalized VPM version that overcomes the above drawbacks. The basic idea consists in alternating cycles of iterations, where a cycle with one steplength selection rule follows a cycle with the other rule. The purpose of the strategy is to attenuate the poor behavior observed in VPMs when the BB rules are used separately. In practice, we try to use the rules alternation technique to avoid both the slow descent produced by rule (6) and the frequent linesearch step reductions arising with rule (5).

Such an approach is expected to be effective on the basis of both the good performance obtained in [18, 19, 38], where a VPM switches between (5) and (6) every three iterations, and the previously mentioned promising results obtained by alternating strategies for unconstrained problems.

In detail, our alternating strategy is as follows. Let $0 < n_{\min} \leq n_{\max}$ be the prefixed minimum and maximum cycle lengths, respectively. Let n_α be the number of consecutive iterations that use the same steplength selection rule, (5) or (6). We switch from one rule to the other if either (i) $n_\alpha \geq n_{\max}$, or (ii) $n_{\min} \leq n_\alpha < n_{\max}$ and a changing criterion is satisfied. To present the changing criterion we introduce the following definitions.

DEFINITION 2.1 (SEPARATING STEPLENGTH) Let $\mathbf{x}^{(k)} \in \Omega$, $\mathbf{d}^{(k)}$ as in (4), $\mathbf{d}^{(k)T} G \mathbf{d}^{(k)} > 0$ and $\alpha_k \in [\alpha_{\min}, \alpha_{\max}]$.

If

$$\alpha_{k+1}^{(2)} < \alpha_k < \alpha_{k+1}^{(1)}$$

then α_k is called a separating steplength.

DEFINITION 2.2 (BAD DESCENT GENERATOR) Let $\mathbf{x}^{(k)} \in \Omega$, $\mathbf{d}^{(k)}$ as in (4), $\mathbf{d}^{(k)T} G \mathbf{d}^{(k)} > 0$, $\alpha_k \in [\alpha_{\min}, \alpha_{\max}]$ and

$$\lambda_{\text{opt}} = \arg \min_{\lambda} f(\mathbf{x}^{(k)} + \lambda \mathbf{d}^{(k)}) = \frac{-(G \mathbf{x}^{(k)} + \mathbf{q})^T \mathbf{d}^{(k)}}{\mathbf{d}^{(k)T} G \mathbf{d}^{(k)}}.$$

Given two constants λ_ℓ and λ_u such that $0 < \lambda_\ell \leq 1 \leq \lambda_u$, we say that α_k is a bad descent generator if one of the following conditions is satisfied:

$$(a) \quad \lambda_{\text{opt}} < \lambda_\ell \quad \text{and} \quad \alpha_k = \alpha_k^{(1)}, \quad (9)$$

$$(b) \quad \lambda_{\text{opt}} > \lambda_u \quad \text{and} \quad \alpha_k = \alpha_k^{(2)}. \quad (10)$$

The changing criterion consists in verifying whether the last steplength used is a separating steplength or is a bad descent generator. The rationale of this criterion is the following:

- if α_k is a separating steplength and at least n_{\min} iterations with rule (5) was already performed, we switch to rule (6) in order to reduce the chance of additional linesearch step reductions;
- if α_k is a separating steplength and at least n_{\min} iterations with rule (6) was already performed, we switch to rule (5) in order to reduce the chance of additional small descents in the objective function;
- if α_k is a bad descent generator, then the last steplength produced a descent direction $\mathbf{d}^{(k)}$ along which the limited minimization rule forced too short a step ($\lambda_k = \lambda_{\text{opt}} < \lambda_\ell$) or too poor a minimum approximation ($\lambda_k = 1 \leq \lambda_u < \lambda_{\text{opt}}$). Hence we try to prevent the same drawback in the next iteration by switching to the other steplength rule.

In this way we bind the rule changing to an empirical check on the “quality” of both the new available values for α_{k+1} and the last steplength α_k . We call the VPM version based on this changing criterion *generalized variable projection method* (GVPM) and we state it in Algorithm GVPM.

It is easy to see that GVPM includes the previously discussed VPMs as special cases: VPM($\alpha^{(1)}$) (respectively VPM($\alpha^{(2)}$)) is obtained for $i_\alpha = 1$ (resp. $i_\alpha = 2$) and n_{\min} “extremely” large, while the version used in [38] is obtained for $i_\alpha = 2$ and $n_{\min} = n_{\max} = 3$. The computational cost per iteration is essentially the same as that of VPM.

We now discuss the effects that the alternating strategy presented has on performance. Again the HARK-ERP2 test problem is used to illustrate the GVPM behavior. In Table II, we report the results obtained with $n_{\max} = 10$ and different n_{\min} values. We set $i_\alpha = 2$ in the first iteration, $\lambda_\ell = 0.1$, $\lambda_u = 5$, while all the other

ALGORITHM GVPM (Generalized Variable Projection Method)

STEP 1. *Initialization.* Let $\mathbf{x}^{(0)} \in \Omega$, $i_\alpha \in \{1, 2\}$, $0 < \alpha_{\min} < \alpha_{\max}$, $\alpha_0 \in [\alpha_{\min}, \alpha_{\max}]$, $n_{\min}, n_{\max} \in \mathbb{N}$, $0 < n_{\min} \leq n_{\max}$, $\lambda_\ell \leq 1 \leq \lambda_u$; set $n_\alpha = 1$, $k = 0$.

STEP 2. *Projection.* Terminate if $\mathbf{x}^{(k)}$ satisfies a stopping criterion; otherwise compute the descent direction

$$\mathbf{d}^{(k)} = P_\Omega(\mathbf{x}^{(k)} - \alpha_k(G\mathbf{x}^{(k)} + \mathbf{q})) - \mathbf{x}^{(k)}.$$

STEP 3. *Linesearch.* Compute

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \lambda_k \mathbf{d}^{(k)}, \quad \lambda_k \in (0, 1],$$

with λ_k given by

$$\lambda_k = \arg \min_{\lambda \in [0, 1]} f(\mathbf{x}^{(k)} + \lambda \mathbf{d}^{(k)})$$

STEP 4. *Update.* If $\mathbf{d}^{(k)T} G \mathbf{d}^{(k)} \leq 0$ then

set $\alpha_{k+1} = \alpha_{\max}$;

else

compute $\alpha_{k+1}^{(1)}$, $\alpha_{k+1}^{(2)}$ and λ_{opt} .

If ($n_\alpha \geq n_{\min}$) then

If ($n_\alpha \geq n_{\max}$) or (α_k is a *separating steplength* or a *bad descent generator*) then

set* $i_\alpha \leftarrow \text{mod}(i_\alpha, 2) + 1$, $n_\alpha = 0$;

end.

end.

Compute $\alpha_{k+1} = \min \left\{ \alpha_{\max}, \max \left\{ \alpha_{\min}, \alpha_{k+1}^{(i_\alpha)} \right\} \right\}$;

end.

Set $k \leftarrow k + 1$, $n_\alpha \leftarrow n_\alpha + 1$ and go to step 2.

*Here $\text{mod}(i, j)$ is the remainder of the integer ratio i/j .

TABLE II GVPM on the HARKERP2 test problem.

n_{\min}	iter	ls	time	f_{\min}
1	85	27	0.20	-5.000E-1
2	73	27	0.17	-5.000E-1
3	35	8	0.10	-5.000E-1
4	24	4	0.07	-5.000E-1
5	56	6	0.14	-5.000E-1
6	71	12	0.17	-5.000E-1
7	83	12	0.20	-5.000E-1

parameters and the stopping rule are the same as in the previous experiments. For the case $n_{\min} = 4$, in order to facilitate a comparison with VPMs, we plot the values of α_k and $f(\mathbf{x}^{(k)})$ in Figures 3 and 4, respectively.

These experiments suggest that the alternating strategy appears to be a very promising way to reduce the disadvantages due to the use of a single BB rule in all the VPM iterations. In fact, the steplength alternation allows a large objective function descent to appear immediately after a few iterations with a small descent (see Figure 4), similarly to what is observed for SPGM($\alpha^{(1)}$). Hence, this example shows how the GVPM can remarkably improve the VPMs, as we expected; furthermore, note also that comparable performances with SPGM($\alpha^{(1)}$) are now obtained (see for example the results for $n_{\min} = 3, 4$ in Table II).

We refer to section 3 for a deeper insight into the above and a comparison with VPMs and SPGMs on many

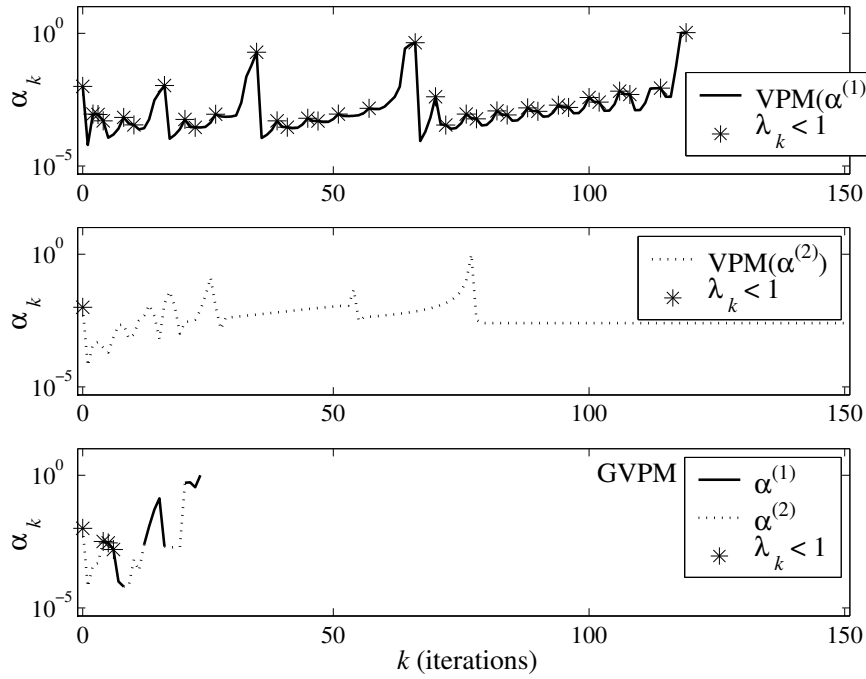


FIGURE 3 α_k values in VPMs and in GVPM for HARKERP2 test problem.

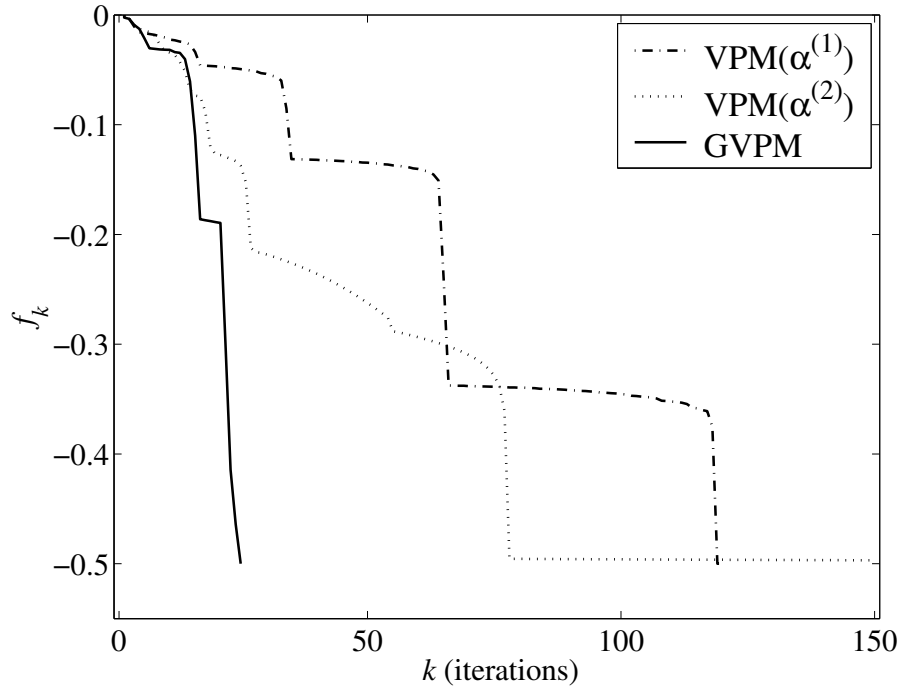


FIGURE 4 f_k values in VPMs and in GVPM for HARKERP2 test problem.

other problems.

Remark. The proposed adaptive alternating strategy is designed to improve the behavior of a gradient projection method based on a monotone linesearch strategy. Generally, it is not successful within schemes based on different linesearch strategies; we will give numerical evidences of this fact in the next section, where the adaptive rule will be tested within SPGMs. In order to introduce an effective adaptive alternating strategy also for nonmonotone or hybrid schemes (the latter using an alternation of monotone and nonmonotone linesearches),

further study will be needed and we will deal with this topic in a future work.

2.3 Convergence analysis for GVPM

In this subsection we prove that, as for both standard gradient projection methods and SPGMs, every accumulation point of the sequence $\{\mathbf{x}^{(k)}\}$ generated by GVPM is a constrained stationary point. To this end, the following two lemmas are useful.

LEMMA 2.2 Let $\mathbf{z} \in \Omega$, $\alpha \in (0, \alpha_{\max}]$ and $\mathbf{d} = P_{\Omega}(\mathbf{z} - \alpha(G\mathbf{z} + \mathbf{q})) - \mathbf{z}$. We have:

$$(a) \quad (G\mathbf{z} + \mathbf{q})^T \mathbf{d} \leq -\frac{1}{\alpha} \|\mathbf{d}\|_2^2 \leq -\frac{1}{\alpha_{\max}} \|\mathbf{d}\|_2^2,$$

(b) $\mathbf{d} = \mathbf{0}$ if and only if \mathbf{z} is a constrained stationary point, that is

$$(G\mathbf{z} + \mathbf{q})^T (\mathbf{x} - \mathbf{z}) \geq 0, \quad \forall \mathbf{x} \in \Omega.$$

Proof The two inequalities are well-known characteristic properties of gradient projection schemes (see [2] and [3, Lemma 2.1]). \square

The previous Lemma ensures that in the GVPM, if $\mathbf{d}^{(k)} \neq \mathbf{0}$ then $\mathbf{d}^{(k)}$ is a descent direction at $\mathbf{x}^{(k)}$. This basic property is used to prove the next result.

LEMMA 2.3 Let $\mathbf{x}^{(k)}$, $\mathbf{d}^{(k)}$ and $\mathbf{x}^{(k+1)}$, $k = 0, 1, \dots$, be as in the GVPM algorithm. If $\mathbf{x}^{(k)}$ is not a constrained stationary point, then

$$(a) \quad \lambda_k \geq \lambda_{\min} = \begin{cases} \min \left\{ 1, \frac{1}{\alpha_{\max} \tau_{\max}(G)} \right\} & \text{if } \tau_{\max}(G) > 0 \\ 1 & \text{if } \tau_{\max}(G) \leq 0 \end{cases}$$

$$(b) \quad f(\mathbf{x}^{(k+1)}) \leq f(\mathbf{x}^{(k)}) + \frac{1}{2} \lambda_k (G\mathbf{x}^{(k)} + \mathbf{q})^T \mathbf{d}^{(k)}$$

where $\tau_{\max}(G)$ is the maximum eigenvalue of G .

Proof To prove (a) we observe that λ_k solves the problem

$$\min_{\lambda \in [0, 1]} \frac{1}{2} \lambda^2 \mathbf{d}^{(k)T} G \mathbf{d}^{(k)} + \lambda (G\mathbf{x}^{(k)} + \mathbf{q})^T \mathbf{d}^{(k)}$$

where $(G\mathbf{x}^{(k)} + \mathbf{q})^T \mathbf{d}^{(k)} < 0$. It is easy to show that

$$\lambda_k = \begin{cases} \min \left\{ 1, \frac{-(G\mathbf{x}^{(k)} + \mathbf{q})^T \mathbf{d}^{(k)}}{\mathbf{d}^{(k)T} G \mathbf{d}^{(k)}} \right\} & \text{if } \mathbf{d}^{(k)T} G \mathbf{d}^{(k)} > 0 \\ 1 & \text{if } \mathbf{d}^{(k)T} G \mathbf{d}^{(k)} \leq 0. \end{cases} \quad (11)$$

From Lemma 2.2 and $\mathbf{d}^{(k)T} G \mathbf{d}^{(k)} \leq \tau_{\max}(G) \|\mathbf{d}^{(k)}\|_2^2$ assertion (a) immediately follows.

In order to prove (b), we recall that

$$\begin{aligned}
f(\mathbf{x}^{(k+1)}) &= f(\mathbf{x}^{(k)}) + \frac{1}{2}(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)})^T G(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}) \\
&\quad + (G\mathbf{x}^{(k)} + \mathbf{q})^T (\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}) \\
&= f(\mathbf{x}^{(k)}) + \frac{1}{2}\lambda_k^2 \mathbf{d}^{(k)T} G \mathbf{d}^{(k)} + \lambda_k (G\mathbf{x}^{(k)} + \mathbf{q})^T \mathbf{d}^{(k)}.
\end{aligned} \tag{12}$$

Consider the case $\mathbf{d}^{(k)T} G \mathbf{d}^{(k)} > 0$; from (11) we have

$$\lambda_k = \min \left\{ 1, \frac{-(G\mathbf{x}^{(k)} + \mathbf{q})^T \mathbf{d}^{(k)}}{\mathbf{d}^{(k)T} G \mathbf{d}^{(k)}} \right\}.$$

If $\lambda_k = 1$ then $-(G\mathbf{x}^{(k)} + \mathbf{q})^T \mathbf{d}^{(k)} \geq \mathbf{d}^{(k)T} G \mathbf{d}^{(k)}$, so assertion (b) holds because of (12). On the other hand, if

$$\lambda_k = -(G\mathbf{x}^{(k)} + \mathbf{q})^T \mathbf{d}^{(k)} / (\mathbf{d}^{(k)T} G \mathbf{d}^{(k)})$$

then we have

$$\begin{aligned}
f(\mathbf{x}^{(k+1)}) &= f(\mathbf{x}^{(k)}) - \frac{1}{2} \frac{\left((G\mathbf{x}^{(k)} + \mathbf{q})^T \mathbf{d}^{(k)} \right)^2}{\mathbf{d}^{(k)T} G \mathbf{d}^{(k)}} \\
&= f(\mathbf{x}^{(k)}) + \frac{1}{2} \lambda_k (G\mathbf{x}^{(k)} + \mathbf{q})^T \mathbf{d}^{(k)}.
\end{aligned}$$

Consider now the case $\mathbf{d}^{(k)T} G \mathbf{d}^{(k)} \leq 0$: from (11) we have $\lambda_k = 1$ and by using (12) the inequality (b) easily follows. \square

Lemma 2.3 shows that in GVPM we have $\inf \lambda_k \geq \lambda_{\min} > 0$. Moreover, it gives an inequality for the sequence $\{f(\mathbf{x}^{(k)})\}$ which is a special case of the inequality required in the nonmonotone linesearch procedure used in SPGM (corresponding to $M = 1$ and $\gamma = 1/2$). This allows one to derive the following convergence result for GVPM:

THEOREM 2.1 *Let $\{\mathbf{x}^{(k)}\}$ be the sequence generated by the GVPM algorithm. Then every accumulation point $\bar{\mathbf{x}}$ of $\{\mathbf{x}^{(k)}\}$ is a constrained stationary point.*

Proof By using the assertions of Lemma 2.3, one proceeds exactly as in case 2 of Theorem 2.4 in [3]. \square

The conclusion of Theorem 2.1 also holds for VPMs, since they are special cases of GVPM, and it joins the convergence results presented in [18, 34] for the convex case. For convex quadratic programs, an additional result can be given for the GVPM convergence rate.

THEOREM 2.2 *Let the function $f(\mathbf{x})$ in (3) be convex and bounded below on Ω . The sequence $\{\mathbf{x}^{(k)}\}$ generated by the GVPM algorithm converges at least R -linearly to a solution of (3).*

Proof The theorem can be proved by following the proof of Theorem 2.1 in [34], where the R-linear convergence of VPMs for convex problems is established. In fact, GVPM differs from VPMs only in the choice of the steplength $\alpha_{k+1} \in [\alpha_{\min}, \alpha_{\max}]$ and the convergence analysis developed in [34] does not depend on the α_{k+1} choice within the interval $[\alpha_{\min}, \alpha_{\max}]$. \square

3 Numerical Experiments

Here we analyse the behavior of the above gradient projection methods on three sets of QP test problems with simple constraints: (i) some problems of the CUTE collection, (ii) randomly generated problems and (iii) problems arising in training SVMs.

The aim of these experiments is to show the improvements that appropriate BB rules alternations can produce in both monotone and nonmonotone gradient projection schemes.

Besides $\text{SPGM}(\alpha^{(1)})$ and $\text{SPGM}(\alpha^{(2)})$, we test SPGM versions where the steplength $\alpha_{k+1} \in [\alpha_{\min}, \alpha_{\max}]$ is selected in one of the following ways:

AL₁-SPGM: start with rule (5) and switch the BB rules at each iteration;

AL₃-SPGM: start with rule (5) and switch the BB rules every three iterations;

AD₁-SPGM: adaptively alternate as in the GVPM with the initial value $i_\alpha = 1$ and $n_{\min} = 1$;

AD₃-SPGM: adaptively alternate as in the GVPM with the initial value $i_\alpha = 1$ and $n_{\min} = 3$.

Analogously, besides $\text{VPM}(\alpha^{(1)})$ and $\text{VPM}(\alpha^{(2)})$, we consider VPM versions where the steplength is chosen this way:

AL₁-VPM: start with rule (6) and switch the BB rules at each iteration;

AL₃-VPM: start with rule (6) and switch the BB rules every three iterations.

Furthermore, the GVPM versions with the initial value $i_\alpha = 2$ and $n_{\min} = 1$ (GVPM₁) or $n_{\min} = 3$ (GVPM₃) are used.

All the other parameters required by the methods are set as in the previous section and the limit of 30000 iterations is added to the stopping rule. Since both AD-SPGMs and GVPMs depend on the initial i_α value, we consider only those versions that generally give better performance. The same holds for AL-SPGMs and AL-VPMs.

Some remarks about the choice of the above methods are in order. First of all, the interest for AL₁-SPGM and AL₃-SPGM alternating strategies is motivated by the good results they produced in different contexts [21, 38]. Second, as observed in the remark of subsection 2.2, one can reasonably expect that the GVPM adaptive alternation does not imply remarkable improvements within nonmonotone schemes; AD₁-SPGM and

TABLE III Box constrained QP test problems from CUTE.

Test Problem	Name	n	f_{\min}
TP1	BIGGSB1	1000	1.590E-02
TP2	BQPGABIM	50	-3.790E-05
TP3	BQPGASIM	50	-5.520E-05
TP4	CHENHARK	1000	-2.000E+00
TP5	NCVXBQP2	10000	-1.334E+10
TP6	NCVXBQP3	10000	-6.558E+09

AD₃-SPGM give numerical evidence. Last, the analogous versions of the monotone approach allow a complete comparison and emphasize the GVPM benefits.

We recall that, in all the results reported, the problem solution time is in seconds and does not include data loading or generation. The experiments of the first two sets are carried out in MATLAB on a Digital personal workstation 500au at 500MHz with 512MB of RAM.

The CUTE box-constrained QP problems we consider are listed in Table III, together with the best computed function value, which is the same for all methods. In TP1, where the starting point is not available in the CUTE model, we use $x_i^{(0)} = 0.5$, $i = 1, \dots, n$, while in all the other cases we use the starting point of the model. The numerical results are reported in Table IV; here, and in the following tables, the symbol “*” means that the 30000 iterations limit is reached.

For the second set of experiments we randomly generate strictly convex QP problems of the form (3) with

$$\begin{aligned}\Omega &= \{ \mathbf{x} \in \mathbb{R}^n, \mathbf{x} = (x_1, \dots, x_n)^T, \quad -1 \leq x_i \leq 1, \quad i = 1, \dots, n \}, \\ \mathbf{q} &= (q_1, \dots, q_n)^T, \quad q_i = -1 + 2(i-1)/(n-1), \quad i = 1, \dots, n, \\ G &= QDQ^T,\end{aligned}$$

where Q is an orthogonal matrix and $D = \text{diag}\{\tau_1, \tau_2, \dots, \tau_n\}$ is a diagonal matrix with τ_i , $i = 1, \dots, n$, as diagonal entries (eigenvalues of G). The matrix Q is obtained from the QR factorization of a random matrix generated by the MATLAB `rand` command: `[Q R] = qr(-1 + 2*rand(n))`. Given $\tau_{\max}(G)$ and the spectral condition number of G , $K(G) = \tau_{\max}(G)/\tau_{\min}(G)$, let $\tau_1 = \tau_{\min}(G)$ and $\tau_n = \tau_{\max}(G)$. Then the following three eigenvalues distributions are considered:

A) uniform distribution:

$$\tau_i = \tau_1 + \theta_i(\tau_n - \tau_1), \quad \theta_i \in (0, 1), \quad i = 2, \dots, n-1;$$

B) 90% of the eigenvalues close to $\tau_{\min}(G)$:

$$\bar{\tau} = \tau_1 K(G)^{0.2}, \quad \tau_i = \tau_1 + \theta_i(\bar{\tau} - \tau_1), \quad i = 2, \dots, [0.9n],$$

$$\tau_i = \bar{\tau} + \theta_i(\tau_n - \bar{\tau}), \quad i = [0.9n] + 1, \dots, n-1;$$

C) 90% of the eigenvalues close to $\tau_{\max}(G)$:

$$\begin{aligned}\bar{\tau} &= \tau_1 K(G)^{0.9}, \quad \tau_i = \tau_1 + \theta_i(\bar{\tau} - \tau_1), \quad i = 2, \dots, \lfloor 0.1n \rfloor, \\ \tau_i &= \bar{\tau} + \theta_i(\tau_n - \bar{\tau}), \quad i = \lfloor 0.1n \rfloor + 1, \dots, n - 1;\end{aligned}$$

where each $\theta_i \in (0, 1)$ is given by the MATLAB function `rand`.

The behavior of SPGMs and VPMs on some problems generated in this way is shown in Tables V–VII. We fix $n = 200$ and we consider different values for $K(G)$ and $\tau_{\max}(G)$. For each value of $K(G)$ and $\tau_{\max}(G)$, the average results obtained by solving 30 random problems are reported (the null vector is the initial guess).

Considering the iteration counts in the two experiment sets, the following observations can be made:

1) SPGM behavior:

- SPGM($\alpha^{(1)}$) does not definitively outperform SPGM($\alpha^{(2)}$): even if the former is better on CUTE test problems, the second set of experiments shows that the opposite is true for particular eigenvalue distributions (see the case with $\tau_{\max} = 10^4$ in Tables V–VII).
- Of the two AL-SPGM versions, AL₁-SPGM seems preferable. The BB rule alternation used in AL₁-SPGM often improves (sometimes significantly) the SPGM($\alpha^{(1,2)}$) performance, as is claimed to happen in gradient methods for unconstrained problems [21]. However, the results for TP5, TP6 in Table IV and for $K(G) = 10^4, \tau_{\max} = 10^2$ and $K(G) = 10^6, \tau_{\max} = 10^2, 10^4$ in Table VI show that this is not always the case. In particular, for some of these test problems the AL₁-SPGM iteration amount is greatly increased with respect to SPGM($\alpha^{(1)}$). In conclusion, a simple BB rule alternation at each iteration appears promising for improving SPGM($\alpha^{(1,2)}$), but cannot be considered a completely satisfactory strategy.
- The best AD-SPGM result outperform both SPGM($\alpha^{(1,2)}$) and AL-SPGM_{1,3} only in two particular cases: $K(G) = 10^6, \tau_{\max} = 10^4$ in Tables V and VI. This poor performance is not surprising, as we already pointed out at the end of subsection 2.2.

2) VPM behavior:

- As for SPGMs, one cannot state which BB rule works definitively better than the other within VPMs. In both the test problem sets, there are cases where VPM($\alpha^{(1)}$) is more efficient than VPM($\alpha^{(2)}$) and cases where the opposite happens.
- The AL-VPM schemes also display behavior that is heavily problem-dependent and one version is not always preferable to the other. In many cases, the best result given by AL₁-VPM and AL₃-VPM is better than the result obtained with both VPM($\alpha^{(1)}$) and VPM($\alpha^{(2)}$). This confirms the importance of BB rules alternations also in VPM schemes. However, the AL-VPM performance is often significantly improved by the GVPMS.

- Concerning GVPMs, we observe that except for two cases (TP6 in Table IV and $K(G) = 10^6, \tau_{\max} = 10^4$ in Table VI), the best result produced by GVPM_1 and GVPM_3 is better than the results given by both VPM and AL-VPM schemes, or comparable to within 10%. Furthermore, deeper insight into the comparison between $\text{AL}_1\text{-VPM}$ and GVPM_1 (resp. $\text{AL}_3\text{-VPM}$ and GVPM_3) shows how GVPM adaptive alternation can be an useful strategy for improving behavior due to a switching after a prefixed number of iterations. Finally, it is worth emphasizing that the best GVPMs results are always competitive with (and sometimes better than) the best SPGMs results.

The next experiments on SVM QP problems reinforce all the above considerations and show how the GVPMs can be a valuable alternative to both $\text{SPGM}(\alpha^{(1,2)})$ and AL-SPGMs.

To evaluate the above methods on some QP problems of the form (2) we train Gaussian SVMs on two real-world data sets: the MNIST database of handwritten digits [24] and the UCI Adult data set [27].

These experiments are carried out on a Compaq XP1000 workstation at 667MHz with 1GB of RAM, with standard C codes. All the methods considered compute the projection on the special feasible region of (2) by solving the equivalent separable QP problem via the algorithm proposed in [30]. Since this is an $O(n)$ algorithm, the main computational cost of each iteration of all methods becomes the matrix-vector product $G\mathbf{d}^{(k)}$. The implementation of this product exploits the expected vector sparsity, often allowing a dramatic reduction of the operation cost and a great improvement in the performance of the solvers. The parameters are set as in the previous experiments, but a different stopping rule is used, which is based on the fulfilment of the KKT conditions within 10^{-3} (as is usually the case in SVM applications [23, 31]). The equality constraint multiplier is computed as suggested in [23] and the null vector is always the initial guess.

A first set of six test problems ($T_i, i = 1, \dots, 6$) sized $n = 3700$ is considered. These problems are obtained as follows. We consider the whole MNIST database and we train a Gaussian classifier for digit “8” with SVM parameters $C = 10$ and $\sigma = 1800$: this leads to a QP problem of the form (2) sized 60000. Then we approach this large problem with the decomposition technique introduced in [38] and briefly discussed in the next section. The problems $T_i, i = 1, \dots, 6$, are the subproblems generated in the six iterations of the decomposition technique (where the subproblem size is set to 3700).

The iterations required by both the VPM and SPGM approaches on these six test problems are reported in Table VIII, where SV and BSV denote the number of support vectors and bound support vectors, respectively. We observe that, generally speaking, GVPM_3 yields the best performance, except in the case of test problem T_1 where $\text{AL}_3\text{-VPM}$, GVPM_3 and $\text{SPGM}(\alpha^{(1)})$ show very similar results. In particular GVPM_3 improves $\text{AL}_3\text{-VPM}$, which is the inner QP solver used in the decomposition technique [38]. The next section will show the decomposition technique performance improvements due to this adaptive QP solver.

We conclude these experiments by comparing the GVPM_3 behavior with the packages used as QP inner solvers in two well-known decomposition techniques for problem (2) [23, 29] (we refer the reader to [3] for a $\text{SPGM}(\alpha^{(1)})$ versus LANCELOT comparison on box-constrained problems).

Two solvers are suggested in Joachims’ *SVM^{light}* package [23]: the Hildreth-D’Esopo method and a version of Vanderbei’s primal-dual infeasible interior point method [36], named `pr_LOQO` and implemented by Smola [35]. We use the latter because it appears more efficient when the problem size increases. Moreover, we consider MINOS (ver. 5.5) [28], which is the solver used in [29]. A solution accuracy comparable with that given by `GVPM3` is obtained by setting `sigfig_max = 8` in `pr_LOQO` and `Optimality tolerance = 10-4` in MINOS, while for better performance `Superbasics limit = 1000` is used in MINOS. Default settings are otherwise used.

For these experiments, test problems of size $n = 800, 1600$ and 3200 are constructed from MNIST by taking into account the first $n/2$ inputs of digit “8” and the first $n/2$ inputs of the other digits. From the UCI Adult database, the versions with size $n = 1605, 2265$ and 3185 are considered. The following setting of the SVM parameters is used: $C = 10$, $\sigma = 2000$ for the MNIST database and $C = 1$, $\sigma^2 = 10$ for the UCI Adult data sets.

Table IX reports the results obtained by the two packages and the `GVPM3`, run on the same Compaq platform: it is evident that the gradient projection method considered allows a remarkable performance improvement with respect to both `pr_LOQO` and MINOS, because of its low iteration cost and its good convergence rate.

Finally, we also tested gradient projection schemes to train polynomial SVMs: in this case their convergence rate appears heavily dependent on the data normalization and they generally show less encouraging results. However, preliminary investigations reveal that the diagonally scaled [2] versions of gradient projection methods can achieve equally appreciable performance. We will deal with this topic in a future work.

Summing up, the good `GVPM3` effectiveness on medium-scale quadratic programs makes it an attractive inner solver in decomposition techniques for large-scale SVMs, as we explain in the next section.

TABLE IV VPMs and SPGMs on CUTE test problems

Problem	SPGM($\alpha^{(1)}$)			SPGM($\alpha^{(2)}$)			AL ₁ -SPGM			AL ₃ -SPGM			AD ₁ -SPGM			AD ₃ -SPGM		
	it	ls	time	it	ls	time	it	ls	time	it	ls	time	it	ls	time	it	ls	time
TP1	2051	383	10.14	4004	1	18.83	867	13	4.62	1690	278	9.68	1343	137	6.80	1901	318	9.92
TP2	24	2	0.06	33	1	0.07	29	1	0.10	31	1	0.10	26	2	0.07	33	2	0.08
TP3	33	2	0.07	37	1	0.08	31	1	0.10	32	1	0.10	33	1	0.08	31	2	0.08
TP4	3900	762	21.89	10576	0	59.24	1359	44	8.10	2649	421	17.11	2292	241	12.98	2149	364	12.77
TP5	83	4	4.44	*			250	4	13.42	223	20	12.13	289	5	15.38	128	6	6.88
TP6	116	6	6.11	*			888	62	46.87	839	22	44.97	1131	42	59.25	195	11	10.24

Problem	VPM($\alpha^{(1)}$)			VPM($\alpha^{(2)}$)			AL ₁ -VPM			AL ₃ -VPM			GVPM ₁			GVPM ₃		
	it	ls	time	it	ls	time	it	ls	time	it	ls	time	it	ls	time	it	ls	time
TP1	*			3973	100	15.91	816	226	3.81	3135	974	14.56	777	191	3.02	1119	304	4.25
TP2	75	38	0.10	33	6	0.05	26	7	0.07	27	10	0.07	28	8	0.05	29	8	0.05
TP3	79	40	0.11	30	9	0.05	30	9	0.80	28	8	0.07	29	10	0.05	30	10	0.05
TP4	*			10700	33	53.00	1639	436	8.63	2436	737	12.60	1545	323	7.01	2163	596	9.60
TP5	106	49	5.23	*			245	6	12.30	119	30	6.01	196	10	9.82	117	21	5.86
TP6	612	304	29.46	*			1228	4	60.65	198	43	9.80	1221	6	59.76	228	54	11.10

TABLE V Random QP test problems, eigenvalues distribution A ($N = 200$).

$K(G)$	τ_{\max}	SPGM($\alpha^{(1)}$)			SPGM($\alpha^{(2)}$)			AL ₁ -SPGM			AL ₃ -SPGM			AD ₁ -SPGM			AD ₃ -SPGM		
		it	ls	time	it	ls	time	it	ls	time	it	ls	time	it	ls	time	it	ls	time
10^2	1	16	0	0.07	22	0	0.10	17	0	0.09	16	0	0.09	17	0	0.08	17	0	0.08
	10^4	83	9	0.34	76	2	0.31	80	4	0.40	78	8	0.40	79	6	0.34	83	7	0.35
10^4	1	16	0	0.07	23	0	0.10	18	0	0.09	16	0	0.09	18	0	0.08	17	0	0.08
	10^2	301	51	1.27	528	3	2.13	205	7	1.02	269	41	1.38	254	27	1.11	290	49	1.27
	10^4	1669	323	7.19	975	13	3.94	958	30	4.77	1171	186	6.08	1457	149	6.44	1702	302	7.70
10^6	1	16	0	0.07	23	0	0.10	18	0	0.10	16	0	0.09	18	0	0.08	18	0	0.08
	10^2	267	45	1.12	547	2	2.22	192	7	0.96	264	40	1.35	233	25	1.02	258	42	1.13
	10^4	3044	583	13.38	2720	8	11.97	2765	60	15.49	2818	450	15.49	2155	202	9.79	2534	417	11.98
$K(G)$	τ_{\max}	VPM($\alpha^{(1)}$)			VPM($\alpha^{(2)}$)			AL ₁ -VPM			AL ₃ -VPM			GVPM ₁			GVPM ₃		
		it	ls	time	it	ls	time	it	ls	time	it	ls	time	it	ls	time	it	ls	time
10^2	1	21	10	0.09	22	0	0.09	17	3	0.09	16	3	0.08	18	3	0.08	17	3	0.07
	10^4	466	234	1.65	81	18	0.29	72	24	0.33	84	30	0.38	75	24	0.28	81	25	0.30
10^4	1	22	10	0.09	23	0	0.09	17	3	0.08	16	4	0.08	18	3	0.07	18	3	0.07
	10^2	1941	972	7.04	525	17	1.89	224	62	1.01	246	77	1.11	219	57	0.81	207	59	0.76
	10^4	*			1050	75	3.76	985	275	4.42	1198	352	5.39	832	202	3.10	1076	307	3.96
10^6	1	22	10	0.09	23	0	0.09	17	3	0.08	16	3	0.08	18	3	0.07	18	3	0.07
	10^2	2121	1062	7.80	552	14	1.98	235	64	1.05	222	70	1.00	196	51	0.73	195	56	0.71
	10^4	*			2515	64	9.67	2695	732	13.94	3200	939	16.43	1938	425	7.76	2260	634	9.27

TABLE VI Random QP test problems, eigenvalues distribution B ($N = 200$).

$K(G)$	τ_{\max}	SPGM($\alpha^{(1)}$)			SPGM($\alpha^{(2)}$)			AL ₁ -SPGM			AL ₃ -SPGM			AD ₁ -SPGM			AD ₃ -SPGM		
		it	ls	time	it	ls	time	it	ls	time	it	ls	time	it	ls	time	it	ls	time
10^2	1	17	0	0.08	66	0	0.29	31	0	0.17	30	0	0.16	32	0	0.15	28	0	0.13
	10^4	89	9	0.37	84	3	0.35	83	6	0.42	81	9	0.42	81	6	0.35	81	7	0.35
10^4	1	20	0	0.09	245	0	1.11	45	0	0.24	40	0	0.22	44	0	0.21	35	0	0.16
	10^2	479	85	2.22	1150	3	5.19	2255	2	12.47	2540	56	14.16	916	31	4.37	561	58	2.67
	10^4	2084	472	9.19	1195	19	4.90	1024	27	5.16	1275	204	6.72	1504	150	6.74	1844	324	8.47
10^6	1	20	0	0.09	255	0	1.15	46	0	0.25	40	0	0.22	43	0	0.20	36	0	0.17
	10^2	861	158	4.05	9296	2	47.98	4950	2	28.72	6787	42	40.66	1715	38	8.31	1130	115	5.46
	10^4	21641	4312	123.93	18222	32	97.59	*			*			18990	369	107.97	13270	1381	71.52

$K(G)$	τ_{\max}	VPM($\alpha^{(1)}$)			VPM($\alpha^{(2)}$)			AL ₁ -VPM			AL ₃ -VPM			GVPM ₁			GVPM ₃		
		it	ls	time	it	ls	time	it	ls	time	it	ls	time	it	ls	time	it	ls	time
10^2	1	22	10	0.09	66	0	0.26	31	0	0.15	26	5	0.13	29	1	0.12	25	5	0.10
	10^4	318	160	1.12	87	19	0.31	79	26	0.36	84	30	0.38	78	26	0.29	83	27	0.30
10^4	1	27	12	0.11	245	0	0.97	48	1	0.24	32	6	0.16	42	2	0.17	30	6	0.12
	10^2	2641	1321	10.62	1149	14	4.58	2214	9	10.99	541	127	2.64	2357	9	9.88	572	140	2.31
	10^4	*			1313	98	4.74	817	219	3.67	1105	328	4.99	778	179	2.90	834	239	3.07
10^6	1	27	11	0.10	255	0	1.01	48	1	0.24	32	6	0.16	42	1	0.17	30	6	0.12
	10^2	5211	2605	22.19	9160	3	42.08	5989	9	31.67	943	213	4.62	5815	9	26.02	1018	238	4.16
	10^4	*			16721	90	79.07	*			15853	3661	87.64	*			17782	4229	87.19

TABLE VII Random QP test problems, eigenvalues distribution C ($N = 200$).

$K(G)$	τ_{\max}	SPGM($\alpha^{(1)}$)			SPGM($\alpha^{(2)}$)			AL ₁ -SPGM			AL ₃ -SPGM			AD ₁ -SPGM			AD ₃ -SPGM		
		it	ls	time	it	ls	time	it	ls	time	it	ls	time	it	ls	time	it	ls	time
10^2	1	15	0	0.06	25	0	0.11	18	0	0.10	15	0	0.08	18	0	0.08	17	0	0.08
	10^4	80	8	0.33	72	2	0.29	73	4	0.36	75	7	0.38	73	6	0.32	73	7	0.31
10^4	1	14	0	0.06	21	0	0.09	16	0	0.09	15	0	0.08	17	0	0.08	16	0	0.07
	10^2	232	40	0.97	299	2	1.19	168	6	0.84	185	27	0.94	203	19	0.87	239	37	1.05
	10^4	2241	638	10.19	994	14	4.00	979	26	4.87	1002	158	5.19	1439	145	6.34	1658	294	7.48
10^6	1	15	0	0.07	21	0	0.09	16	0	0.08	15	0	0.08	16	0	0.07	16	0	0.07
	10^2	464	85	2.02	430	2	1.73	201	6	1.00	264	39	1.35	237	22	1.03	248	38	1.08
	10^4	3029	585	13.30	2362	9	9.99	1974	44	10.64	2993	477	16.41	2235	204	10.39	2142	349	10.06
$K(G)$	τ_{\max}	VPM($\alpha^{(1)}$)			VPM($\alpha^{(2)}$)			AL ₁ -VPM			AL ₃ -VPM			GVPM ₁			GVPM ₃		
		it	ls	time	it	ls	time	it	ls	time	it	ls	time	it	ls	time	it	ls	time
10^2	1	21	9	0.08	25	0	0.10	18	1	0.09	16	4	0.08	18	2	0.07	17	3	0.07
	10^4	268	134	0.95	77	16	0.28	68	21	0.31	76	27	0.34	71	23	0.27	73	23	0.27
10^4	1	20	9	0.08	21	0	0.08	17	2	0.08	16	3	0.08	17	2	0.07	16	3	0.07
	10^2	2531	1266	10.98	310	14	1.11	184	47	0.83	184	59	0.83	166	40	0.62	174	48	0.65
	10^4	*			1074	74	3.88	898	248	4.04	1156	343	5.23	654	155	2.47	908	259	3.39
10^6	1	20	9	0.08	21	0	0.08	17	3	0.08	16	3	0.07	16	2	0.07	16	3	0.07
	10^2	1642	829	6.11	430	14	1.56	280	76	1.27	251	77	1.14	255	60	0.97	272	77	1.02
	10^4	*			2305	56	8.86	2637	719	13.58	2713	795	13.61	1952	417	7.99	1864	522	7.37

TABLE VIII Iteration numbers on the SVM test problems from MNIST database.

test	SV	BSV	VPM				GVPM		SPGM			
			$\alpha^{(1)}$	$\alpha^{(2)}$	AL ₁	AL ₃	1	3	$\alpha^{(1)}$	$\alpha^{(2)}$	AL ₁	AL ₃
T_1	545	2	1150	1077	2088	347	605	377	360	1102	2113	1942
T_2	1727	75	5863	3195	7900	966	910	793	1211	2259	8195	8692
T_3	2262	95	7099	1967	6821	875	1007	764	1051	2123	6058	8032
T_4	2694	32	7768	2171	6293	978	1325	769	1108	2244	6394	7886
T_5	2963	5	12565	2812	9120	1444	1584	909	1521	2655	9479	12722
T_6	2993	0	10015	2068	8596	1290	3200	1102	1501	2208	8059	11007

TABLE IX Results for pr_LOQO, MINOS and GVPM₃ on SVM test problems.

Prob.	n	pr_LOQO				MINOS				GVPM ₃			
		it	time	SV	BSV	it	time	SV	BSV	it	time	SV	BSV
MNIST	800	14	9.42	283	1	622	6.84	281	1	161	0.50	281	1
	1600	15	112.38	458	9	1043	46.11	455	9	277	3.20	456	9
	3200	15	1124.61	809	25	1894	337.24	806	25	513	21.27	808	25
UCI Adult	1605	15	131.6	691	584	1066	33.8	691	585	153	1.00	691	584
	2265	15	383.9	1011	847	1580	101.3	1007	849	196	2.75	1011	847
	3185	15	1081.4	1300	1109	2004	248.7	1294	1116	282	6.95	1299	1113

4 Gradient Projection Decomposition Techniques for SVMs

The most popular decomposition technique that uses a numerical solver for the inner QP subproblems is the SVM^{light} algorithm introduced in [23] (see [25, 26] for its convergence properties). Unfortunately, this technique is designed to be effective only when the subproblem size is very small (generally less than 10^2); it is not designed to benefit from high-performance solvers for medium-scale QP problems. Thus, a different implementation is proposed in [38], which is still based on the SVM^{light} idea but is appropriately designed to decompose the problem efficiently into medium-scale subproblems. In [38], the AL₃-VPM is used as the inner subproblem solver. Here, in view of the good results obtained in the previous experiments by GVPM₃, we evaluate the improvement in the performance of the decomposition technique when the new solver is used. For the sake of completeness, SPGM($\alpha^{(1)}$) is also considered as inner solver. In the following we call the decomposition technique [38] equipped with these inner solvers the *gradient projection-based decomposition technique* (GPDT). To describe the following experiments, we briefly recall in Algorithm DT the main decomposition steps for QP problems of the form (2). We refer the reader to [23] and [38] for further details of the two implementations.

We compare the numerical results obtained on some large-scale test problems by GPDT and by the SVM^{light} software (version 3.5)¹ equipped with pr_LOQO. We consider the largest test problems available in the MNIST and the UCI Adult data sets: the former is sized 60000 and is obtained as in [31] by training just the class “8” classifier, while the latter is sized 32562. Gaussian SVMs are trained with $C = 10$, $\sigma = 1800$ for MNIST and $C = 1$, $\sigma^2 = 10$ for UCI Adult. The experiments are carried out with standard C codes on the above Compaq workstation. Both softwares use a 500 MB caching area and the default SVM^{light} stopping rule. The parameters

¹We also tested the more recent version 5.0, but we got slightly worse performance.

STEP 1. *Initialization.* Let $\mathbf{x}^{(1)} = (x_1^{(1)}, \dots, x_n^{(1)})^T$ be a feasible point for (2), let n_{sp} and n_c be two integer values such that $n \geq n_{sp} \geq n_c > 0$ and set $k = 1$. Arbitrarily split the indices of the variables $x_i^{(k)}$ into the set \mathcal{B} of *basic* variables, with $\#\mathcal{B} = n_{sp}$, and the set \mathcal{N} of *nonbasic* variables. Arrange the arrays $\mathbf{x}^{(k)}$ and G with respect to \mathcal{B} and \mathcal{N} :

$$\mathbf{x}^{(k)} = \begin{bmatrix} \mathbf{x}_{\mathcal{B}}^{(k)} \\ \mathbf{x}_{\mathcal{N}}^{(k)} \end{bmatrix}, \quad G = \begin{bmatrix} G_{\mathcal{B}\mathcal{B}} & G_{\mathcal{B}\mathcal{N}} \\ G_{\mathcal{N}\mathcal{B}} & G_{\mathcal{N}\mathcal{N}} \end{bmatrix}.$$

STEP 2. *QP subproblem.* Compute the solution $\mathbf{x}_{\mathcal{B}}^{(k+1)}$ of

$$\begin{aligned} \min \quad & \frac{1}{2} \mathbf{x}_{\mathcal{B}}^T G_{\mathcal{B}\mathcal{B}} \mathbf{x}_{\mathcal{B}} + \left(G_{\mathcal{B}\mathcal{N}} \mathbf{x}_{\mathcal{N}}^{(k)} - (1, 1, \dots, 1)^T \right)^T \mathbf{x}_{\mathcal{B}} \\ \text{sub. to} \quad & \sum_{i \in \mathcal{B}} y_i x_i = - \sum_{i \in \mathcal{N}} y_i x_i^{(k)}, \\ & 0 \leq x_i \leq C \quad \forall i \in \mathcal{B}, \end{aligned}$$

and set $\mathbf{x}^{(k+1)} = \left(\mathbf{x}_{\mathcal{B}}^{(k+1)T}, \mathbf{x}_{\mathcal{N}}^{(k)T} \right)^T$.

STEP 3. *Gradient updating.* Update the gradient $\nabla \mathcal{F}(\mathbf{x}^{(k+1)})$ and terminate if $\mathbf{x}^{(k+1)}$ satisfies the KKT conditions.

STEP 4. *Updating of \mathcal{B} .* Change at most n_c elements of \mathcal{B} . The entering indices are determined by a strategy based on the Zoutendijk's method. Set $k \leftarrow k + 1$ and go to step 2.

TABLE X *SVM^{light}* package on SVMs test problems.

Problem	n_{sp}	n_c	it	time	SV	BSV
	10*	10*	10007	4243.05	3156	160
	4	2	17342	3338.00	3154	160
MNIST	8	4	9171	3193.14	3153	160
$n = 60000$	20	10	3874	3364.17	3152	160
	40	20	1987	3573.68	3156	160
	90	30	895	3791.08	3153	160
	10*	10*	9930	881.19	11690	10602
	4	2	14596	880.80	11572	10740
UCI Adult	8	4	10710	907.45	11623	10675
$n = 32562$	20	10	4459	850.82	11695	10594
	40	20	2514	876.71	11750	10587
	80	40	1267	926.87	11776	10560

*Default parameter setting

setting for the GPDT inner solvers is the same as the previous experiments, while in *SVM^{light}* software default settings are used.

Tables X and XI report the results for different values of n_{sp} (subproblem size) and n_c (maximum number of new variables entering the working set). For *SVM^{light}*, these values are assigned to command line options `q` and `n`, respectively. In both Tables, for each n_{sp} value we report the results corresponding to an empirical approximation of the optimal n_c value, i.e. the one giving the lowest computational time. The iteration numbers (it) refer to the decomposition iterations.

The experiments confirm that *SVM^{light}* achieves the best performance when very small n_{sp} values are used

TABLE XI GPDT on SVMs test problems.

n_{sp}	n_c	SPGM(α^1)				GVPM ₃				AL ₃ -VPM			
		it	time	SV	BSV	it	time	SV	BSV	it	time	SV	BSV
MNIST $n = 60000$													
2900	1100	10	2357.29	3157	159	9	1985.74	3157	159	9	2087.93	3157	160
3100	1200	7	1881.61	3158	159	7	1732.80	3157	159	7	1872.39	3158	159
3700	1500	6	2028.81	3157	160	6	1778.30	3157	159	6	1849.71	3156	159
UCI Adult $n = 32562$													
1000	600	70	689.62	11738	10559	71	675.99	11742	10583	75	757.55	11757	10563
1300	750	42	590.68	11772	10566	42	590.81	11772	10564	42	621.80	11760	10544
1600	800	34	691.08	11757	10536	35	687.80	11753	10544	36	696.15	11759	10566

($n_{sp} = 8$ for MNIST and $n_{sp} = 20$ for UCI Adult). With increasing n_{sp} values each iteration becomes too expensive and the effectiveness decreases. Since in these iterations the QP subproblem solution is a very cheap task by comparison with the kernel evaluations required by the data and gradient updating, the use of an inner solver better than pr_LOQO does not reduce the training time enough.

On the other hand, GPDT shows the lowest solution time when sufficiently large n_{sp} values are used ($n_{sp} = 3100$ for MNIST and $n_{sp} = 1300$ for UCI Adult). This is due both to the high performance of the inner solvers and to the special strategy described in [38] to reduce the kernel evaluations required by the G_{BB} , G_{BN} and $\nabla\mathcal{F}(\mathbf{x}^{(k+1)})$ updating in each decomposition iteration.

Thanks to these two features, when n_{sp} increases, the GPDT iterations do not become excessively expensive and the good convergence rate of the decomposition technique can be fully exploited. On these test problems, of the three inner QP solvers, GVPM₃ appears the most effective and in some cases yields significant improvements with respect to the AL₃-VPM solver used in [38].

Furthermore, a very important GPDT property must be mentioned: its easy parallelization. In fact, since the computational burden is given by few expensive iterations where the heaviest parts are the matrix-vector products in the inner solver and the kernel evaluations, a suitable data distribution allows a very effective GPDT parallel implementation on distributed memory multiprocessor machines. Thus, the large computational resources typically available on modern parallel computers can be fully exploited to solve large or even huge problems of form (2). We do not develop this topic further but refer the interested reader to [38].

5 Conclusions and future developments

In this paper, gradient projection methods based on the two Barzilai-Borwein spectral steplengths are investigated for QP problems with simple constraints, such as those deriving from SVM training. The nonmonotone spectral projected gradient scheme [3] and the variable projection methods [33, 34] are considered. Both classical and alternated BB steplength selections are evaluated within the above approaches. Following the recent suggestions in [18, 19, 21, 38] two alternating strategies are tested that change the BB rule at each iteration or every three iterations. Our experiments show that, when the BB rules are used singly, one rule does not perform definitively better than the other for both SPGM and VPM. As regards the rule alternations, they seem very

promising strategies: in fact, they often remarkably reduce the iteration counts with respect to the cases where a single BB rule is used. However, since this is not always the case, further improved strategies are desirable.

In this work we present a generalized VPM (GVPM) that exploits a new *adaptive* steplength selection between the BB rules and includes the other VPMs as special cases. The GVPM convergence analysis is provided. Extensive numerical experimentation is carried out on three test sets: some CUTE problems, some randomly generated problems and some problems arising in SVM training. This computational experience shows that GVPM generally performs better than VPMs and, further, it often appears to be a valuable alternative to SPGMs. In the case of SVM problems, GVPM largely outperforms both pr_LOQO and MINOS; furthermore, it is successfully tested as an inner solver in the decomposition technique proposed in [38].

Future work will include further extensions of the approaches considered, such as scaling and additional improvements to the steplength selection strategies.

Acknowledgment

The authors are most grateful to Prof. Roger Fletcher for the valuable discussions and suggestions on the Barzilai-Borwein method, as well as to the referees for their constructive comments.

References

- [1] J. Barzilai, J.M. Borwein (1988), Two Point Step Size Gradient Methods, *IMA J. Numer. Anal.* **8**, 141–148.
- [2] D.P. Bertsekas (1999), *Nonlinear Programming*, Athena Scientific, Belmont, MA.
- [3] E.G. Birgin, J.M. Martínez, M. Raydan (2000), Nonmonotone Spectral Projected Gradient Methods on Convex Sets, *SIAM J. Optim.* **10**(4), 1196–1211.
- [4] C.J.C. Burges (1998), A Tutorial on Support Vector Machines for Pattern Recognition, *Data Mining and Knowledge Discovery* **2**(2), 121–167.
- [5] C.C. Chang, C.J. Lin (2002), LIBSVM: a Library for Support Vector Machines, available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [6] R. Collobert, S. Benjo (2001), SVM-Torch: Support Vector Machines for Large-Scale Regression Problems, *Journal of Machine Learning Research* **1**, 143–160.
- [7] C. Cortes, V.N. Vapnik (1995), Support Vector Network, *Machine Learning* **20**, 1–25.
- [8] N. Cristianini, J. Shawe-Taylor, *An Introduction to Support Vector Machines and other Kernel-Based Learning Methods*, Cambridge University Press, (2000).
- [9] Y.H. Dai (2001), Alternate Stepsize Gradient Method, AMSS-2001-041, Academy of Mathematics and Systems Sciences, Chinese Academy of Sciences, China.
- [10] Y.H. Dai, J. Yuan, Y. Yuan (2002), Modified Two-Point Stepsize Gradient Methods for Unconstrained Optimization, *Comput. Optim. and Appl.* **22**, 103–109.
- [11] Y.H. Dai, L.Z. Liao (2002), R -linear convergence of the Barzilai and Borwein Gradient Method, *IMA J. Numer. Anal.* **22**, 1–10.
- [12] M.C. Ferris, T.S. Munson (2000), Interior Point Methods for Massive Support Vector Machines, Data Mining Institute Technical Report 00-05, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin.
- [13] C. Fisk, S. Nguyen (1980), Solution Algorithms for Network Equilibrium Models with Asymmetric User Costs, *Transportation Sci.* **16**, 361–381.
- [14] G.W. Flake, S. Lawrence (2002), Efficient SVM Regression Training with SMO, *Machine Learning* **46**(1), 271–290, available at <http://www.neci.nec.com/homepages/flake/nodelib/html>.
- [15] R. Fletcher (2001), On the Barzilai-Borwein Method, Numerical Analysis Report NA/207.

- [16] A. Friedlander, J.M. Martínez, B. Molina, M. Raydan (1999), Gradient Method with Retards and Generalizations, *SIAM J. Numer. Anal.* **36**, 275–289.
- [17] G. Fung, O.L. Mangasarian (2001), Proximal Support Vector Machines Classifiers, Data Mining Institute Technical Report 01-02, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin.
- [18] E. Galligani, V. Ruggiero, L. Zanni (2000), Projection-Type Methods for Large Convex Quadratic Programs: Theory and Computational Experience, *Monograph 5*, Scientific Research Program “Numerical Analysis: Methods and Mathematical Software”, University of Ferrara, Ferrara, Italy, available at <<http://www.unife.it/AnNum97/monografie.htm>>.
- [19] E. Galligani, V. Ruggiero, L. Zanni (2003), Variable Projection Methods for Large-Scale Quadratic Optimization in Data Analysis Applications, *Equilibrium Problems and Variational Models*, P. Daniele, F. Giannessi and A. Maugeri, eds., Nonconvex Optim. and Its Applic. **68**, Kluwer Academic Publishers, Boston, 186–211.
- [20] L. Grippo, F. Lampariello, S. Lucidi (1986), A Nonmonotone Line Search Technique for Newton’s Method, *SIAM J. Numer. Anal.* **23**, 707–716.
- [21] L. Grippo, M. Sciandrone (2002), Nonmonotone Globalization Techniques for the Barzilai-Borwein Gradient Method, *Computational Optimization and Applications*, **23**, 143–169.
- [22] C.W. Hsu, C.J. Lin (2002), A Simple Decomposition Method for Support Vector Machines, *Machine Learning*, **46**, 291–314.
- [23] T. Joachims (1998), Making Large-Scale SVM Learning Practical, *Advances in Kernel Methods – Support Vector Learning*, B. Schölkopf, C.J.C. Burges and A. Smola, eds., MIT Press, Cambridge, Massachusetts.
- [24] Y. LeCun, MNIST Handwritten Digit Database, available at <<http://www.research.att.com/~yann/ocr/mnist>>.
- [25] C.J. Lin (2001), On the Convergence of the Decomposition Method for Support Vector Machines, *IEEE Trans. on Neural Networks* **12**(6), 1288–1298.
- [26] C.J. Lin (2002), Linear Convergence of a Decomposition Method for Support Vector Machines, Technical Report, Department of Computer Science and Information Engineering, National Taiwan University.
- [27] P.M. Murphy, D.W. Aha (1992), UCI Repository of Machine Learning Databases, available at <<http://www.ics.uci.edu/~mllearn/MLRepository.html>>.
- [28] B. Murtagh, M. Saunders (1995), MINOS 5.4 User’s Guide, System Optimization Laboratory, Stanford University.
- [29] E. Osuna, R. Freund, F. Girosi (1997), An Improved Training Algorithm for Support Vector Machines, *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, J. Principe, L. Giles, N. Morgan and E. Wilson, eds., Amelia Island, FL, 276–285.
- [30] P.M. Pardalos, N. Kovoor (1990), An Algorithm for a Singly Constrained Class of Quadratic Programs Subject to Upper and Lower Bounds, *Math. Programming* **46**, 321–328.
- [31] J.C. Platt (1998), Fast Training of Support Vector Machines using Sequential Minimal Optimization, *Advances in Kernel Methods – Support Vector Learning*, B. Schölkopf, C. Burges and A. Smola, eds., MIT Press, Cambridge, Massachusetts.
- [32] M. Raydan, B.F. Svaiter (2002), Relaxed Steepest Descent and Cauchy-Barzilai-Borwein Method, *Comput. Optim. and Appl.* **21**, 155–167.
- [33] V. Ruggiero, L. Zanni (2000), A Modified Projection Algorithm for Large Strictly Convex Quadratic Programs, *J. Optim. Theory Appl.* **104**(2), 281–299.
- [34] V. Ruggiero, L. Zanni (2000), Variable Projection Methods for Large Convex Quadratic Programs, *Recent Trends in Numerical Analysis*, D. Trigiante, ed., Advances in the Theory of Computational Mathematics 3, Nova Science Publ., 299–313.
- [35] A.J. Smola, pr_LOQO optimizer, available at <<http://www.kernel-machines.org/code/prloqo.tar.gz>>.
- [36] R.J. Vanderbei (1998), LOQO: An Interior Point Code for Quadratic Programming, Technical Report SOR-94-15 Revised, Princeton University.
- [37] V.N. Vapnik (1998), Statistical Learning Theory, John Wiley and sons, New York.
- [38] G. Zanghirati, L. Zanni (2003), A Parallel Solver for Large Quadratic Programs in Training Support Vector Machines, *Parallel Computing* **29**, 535–551.