Noname manuscript No.
(will be inserted by the editor)

Parallel decomposition methods for linearly constrained problems subject to simple bound with application to the SVMs training

Andrea Manno · Laura Palagi · Simone Sagratella

Received: date / Accepted: date

Abstract We consider the convex quadratic linearly constrained problem with bounded variables and with huge and dense Hessian matrix that arises in many applications such as the training problem of bias support vector machines. We propose a decomposition algorithmic scheme suitable to parallel implementations and we prove global convergence under suitable conditions. Focusing on support vector machines training, we outline how these assumptions can be satisfied in practice and we suggest various specific implementations. Extensions of the theoretical results to general linearly constrained problem are provided. We included numerical results on support vector machines with the aim of showing the viability and the effectiveness of the proposed scheme.

Keywords Decomposition algorithm · Big data · Support vector machines · Parallel computing

PACS 90C06 · 49M27 · 90C20 · 65N12 · 68T05 · 68Q32

1 Introduction

We consider the problem

The work of Laura Palagi was partially supported by the italian project PLATINO (Grant Agreement n. PON01_01007); the work of Simone Sagratella was partially supported by the grant: Avvio alla Ricerca 488, Sapienza University of Rome

A. Manno
Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano. Italy.
E-mail: andrea.manno@polimi.it

L. Palagi · S. Sagratella
Department of Computer, Control and Management Engineering, Sapienza University of Rome, Via Ariosto 25, 00185 Rome, Italy.
E-mail: laura.palagi@uniroma1.it, simone.sagratella@uniroma1.it

$$\begin{aligned} \min_{\mathbf{x}} \ & f(\mathbf{x}) \\ & \mathbf{y}^T\mathbf{x} = b \\ & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}, \end{aligned} \tag{1}$$

where $f(\cdot)$ is a convex quadratic function of the type $\frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x}$, $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{y} \in \mathbb{R}^n$, $\mathbf{l} \leq \mathbf{u} \in \mathbb{R}^n$, $b \in \mathbb{R}$.

Large-scale instances of problems of type (1) arise naturally in many real-world applications such as portfolio selection, traffic equilibrium and multicommodity network flow problems (see e.g. [6],[16],[17]). A widely used instance of (1) is the dual formulation of the so called "bias" Support Vector Machine (SVM) training problem, a well-known machine learning technique seeking for a separating surface for classification, that allows an offset in the separating surface. Specifically, in the bias SVM case: $\mathbf{y} \in \{-1, 1\}^n$, $b = 0$, $\mathbf{l} = \mathbf{0}$, $\mathbf{u} = C\mathbf{e}$, where $\mathbf{e} \in \mathbb{R}^n$ is a vector of all ones, and C is a positive constant; furthermore in the quadratic function f, $\mathbf{c} = -\mathbf{e}$; and the Hessian matrix Q is a positive semidefinite large and dense matrix. Entries of Q are defined by $Q_{rq} = y_r y_q K(\mathbf{z}_r, \mathbf{z}_q)$, $r, q = 1, 2, \ldots, n$, where $K : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ is a given kernel function [33] and $(\mathbf{z}_r, y_r)$ are elements of a training set of n input-target pairs

$$D = \{(\mathbf{z}_r, y_r),\ r = 1, \ldots, n,\ \mathbf{z}_r \in \mathbb{R}^m,\ y_r \in \{-1, 1\}\}.$$

Hence the SVM problem is

$$\begin{aligned} \min_{\mathbf{x}} \ & f(\mathbf{x}) := \frac{1}{2}\mathbf{x}^T Q\mathbf{x} - \mathbf{e}^T\mathbf{x} \\ & \mathbf{y}^T\mathbf{x} = 0 \\ & \mathbf{0} \leq \mathbf{x} \leq C\mathbf{e}, \end{aligned} \tag{2}$$

In recent years SVMs have been applied to big or huge datasets, mainly related to web-oriented applications. This implies that the Hessian matrix Q is so big that it cannot be entirely stored in memory so that traditional optimization algorithms cannot be used to efficiently solve problem (2). To overcome this difficulty, many decomposition algorithms have been proposed in the literature for SVM to solve the linearly constrained quadratic problem (2). The equality constraint in (2) represents the main difficulty in the definition of convergent decomposition algorithms. All decomposition methods essentially require the solution of a sequence of smaller problems in which only a subset of the variables (the working set) is updated, but they can differ both in the selection rule of the working set and in the updating rule of the working variables. Proving convergence of decomposition algorithms when changing one of these two crucial aspects is not straightforward and indeed it has motivated a large amount of papers in the literature. In particular the decomposition strategies for the problem of type (2) can be mainly divided into SMO (Sequential Minimal Optimization) and non-SMO methods.

SMO methods update exactly two variables at each iteration. Convergence properties of SMO-type methods can be proved only under suitable selection rules of the two variables. A well known selection rule that guarantees convergence is the Most Violating Pair (MVP) [5,20] which is implemented e.g. in LIBSVM [4]. Other convergent SMO-type methods use different working set selection rules such as the second order rule used in LASVM [11,12] and LIBSVM [10] and the cyclic rule used in [24]. Since the function is quadratic, SMO-type methods analytically compute the optimal stepsize along the selected search direction. The use of larger working sets (non-SMO methods) requires to define both the selection rules and the inexact line searches along the feasible directions (or, equivalently, the inexact solution of the subproblems) like in SVM^{light} [15] and others [13,19,24,25,27,36]. However all previously mentioned decomposition methods for solving (2) are intrinsically sequential, and developing a convergent parallel version of any of them is not straightforward at all.

To reduce the big amount of time needed for computing a solution of problem (2), many parallel algorithms have been proposed. Some of these parallel approaches consist in distributing among the available processors the most expensive tasks, such as subproblems solving and gradient updating, see [38,39]. Another way to fruitfully exploit parallelism is based on splitting the training data into subsets and distributing them among the processors [3,14,37, 40]. While achieving a good reduction of the training time with respect to sequential methods, these methods may lack convergence properties or may require strong assumptions to prove them. Actually, combining the decomposition rules for the selection of the working sets with parallelism makes the proof of convergence a very difficult task, see [23]. This is mainly due to the nonseparability of the objective function and of the feasible set of problem (2).

It is worth to mention that, in the field of SVMs, a recent trend has been toward the unbiased SVM dual problem (see e.g. LIBLINEAR [9], SAGA [7], SDCA [31,32], Mini-batch primal dual [35], Pegasos [30] and others [22,34]). The interest toward unbiased SVMs is due to the fact that, in spite of solving a simpler formulation, unbiased SVMs may produce similar generalization performance than bias ones. Moreover, from the optimization point of view, the unbias version leads to a dual formulation without the linear non separable equality constraint. On the other hand, it is well-known that in certain cases, for example when the distribution of the data in the two classes is uneven (see [30]), the bias plays a crucial role concerning the generalization performance. Recently a new provably convergent method has been proposed in [26] that, iteratively adjusting the offset values, computes a solution of the bias problem (2) by solving a sequence of dual formulations that do not include the difficult equality constraint and, then, can be solved in parallel. Even if this method solves the more general bias version, it requires multiple parallel optimizations in order to solve any single SVM training. Thus it can be inefficient in cases in which the offset must be updated many times.

In this work we propose a new class of convergent decomposition algorithms for the bias problem (2) that easily allows efficient parallel implementations.

We first consider the quadratic case (2), for which we introduce a general fully parallelizable decomposition scheme. We prove its convergence under suitable assumptions and we show how to practically satisfy them. The main idea underneath the convergence proof is to use as merit function a sufficient "predicted movement" rather than a more classical sufficient "predicted objective descent". The convergence analysis partially exploits the results in [8,36]. A specific implementation is tested on some datasets with the sole aim of showing the effectiveness of the proposed parallel strategy.

In the last part of the paper, we focus on the more general case of a problem with multiple equality constraints

$$\begin{aligned} \min_{\mathbf{x}} \ & f(\mathbf{x}) \\ & A\mathbf{x} = \mathbf{b} \\ & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}, \end{aligned} \tag{3}$$

where $f(\cdot)$ is a generic continuously differentiable and convex function, $\mathbf{x} \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{l} \leq \mathbf{u} \in \mathbb{R}^n$ (possibly with $-\infty$ or ∞ components). This formulation includes all instances of convex optimization over polyhedra.

The paper is organized as follows: from Section 2 to Section 7 we consider the specific SVM formulation (2); in particular in Section 2 we introduce some preliminary results and in Section 3 we present the general parallel algorithmic scheme. We analyze its convergence properties in Sections 4, 5 and 6; in Section 7 we discuss about some possible practical implementations and we show some exemplifying numerical tests. Finally, in Section 8 we extend the theoretical results to the general case (3).

Notation In the following we use this notation. Vectors are boldface. Given a vector $\mathbf{x} \in \mathbb{R}^n$ and a subset of indices $P \subseteq \{1, \ldots, n\}$, we denote by $\mathbf{x}_P \in \mathbb{R}^{|P|}$ the subvector made up of all components x_r with $r \in P$, and by $\mathbf{x}_{-P} \in \mathbb{R}^{n-|P|}$ the subvector made up of all components x_r with $r \notin P$. With $\|\cdot\|$ we indicate the euclidean norm, whereas the zero norm of a vector $\|\mathbf{x}\|_0$ denotes the number of nonzero components of the vector. Furthermore, given a square $n \times n$ matrix Q, we denote by Q_{*r} the $r-$th column of the matrix. Given two subsets of indices $P_r, P_q \subseteq \{1, \ldots, n\}$, we write $Q_{P_r P_q}$ to indicate the $|P_r| \times |P_q|$ submatrix of Q with row indices in block P_r and column indices in block P_q. We denote by $\lambda^Q_{\min}$ and $\lambda^Q_{\max}$ the minimum and maximum eigenvalues of Q respectively. We also use the notation $Q \succeq / \succ 0$ to denote that Q is positive semidefinite/definite. We denote the $r-$th component of the gradient of a function f as $\nabla f(\mathbf{x})_r = \frac{\partial f(\mathbf{x})}{\partial x_r}$ and as $\nabla_P f(\mathbf{x}) \in \mathbb{R}^{|P|}$ the subvector of the gradient made up of all components $\frac{\partial f(\mathbf{x})}{\partial x_r}$ with $r \in P$. We denote by $\mathcal{F}$ the feasible sets of both problems (2) and (3).

2 Optimality Conditions and Preliminary Results

Let us consider a solution $\mathbf{x}^*$ of problem (2). Since all the constraints are linear and the objective function is convex, necessary and sufficient conditions for optimality are the Karush-Kuhn-Tucker (KKT) conditions stating that there exists a scalar s such that for all indices $r \in \{1, \ldots, n\}$:

$$\begin{aligned} &\nabla f(\mathbf{x}^*)_r + s y_r \geq 0 \quad \text{if } x_r^* = 0 \\ &\nabla f(\mathbf{x}^*)_r + s y_r \leq 0 \quad \text{if } x_r^* = C \\ &\nabla f(\mathbf{x}^*)_r + s y_r = 0 \quad \text{if } 0 < x_r^* < C. \end{aligned} \tag{4}$$

It is well known (see e.g.[15]) that the KKT conditions can be written in a more compact form by introducing the following sets

$$I_{up}(\mathbf{x}) := \{r \in \{1, \ldots, n\} : \ x_r < C, \ y_r = 1, \text{ or } x_r > 0, \ y_r = -1\},$$

$$I_{low}(\mathbf{x}) := \{r \in \{1, \ldots, n\} : \ x_r < C, \ y_r = -1, \text{ or } x_r > 0, \ y_r = 1\}.$$

Assuming that $I_{up}(\mathbf{x}^*) \neq \emptyset$ and $I_{low}(\mathbf{x}^*) \neq \emptyset$, then we can rewrite (4) as

$$m(\mathbf{x}^*) = \max_{r \in I_{up}(\mathbf{x}^*)} -\nabla f(\mathbf{x}^*)_r y_r \leq \min_{r \in I_{low}(\mathbf{x}^*)} -\nabla f(\mathbf{x}^*)_r y_r = M(\mathbf{x}^*). \tag{5}$$

By the convexity of problem (2), we can say that $\mathbf{x}^*$ is optimal if and only if either $I_{up}(\mathbf{x}^*) = \emptyset$ or $I_{low}(\mathbf{x}^*) = \emptyset$ or condition (5) holds.

Such a form of the KKT conditions is the basis of most efficient sequential decomposition algorithms for the solution of problem (2). In decomposition algorithms the sequence $\{\mathbf{x}^k\}$ is obtained by changing at each iteration k only a subset of the variables, let's say $\mathbf{x}_{P_k}$ with $P_k \subset \{1, \ldots, n\}$, while the other $\mathbf{x}_{-P_k}$ remain unchanged. Thus the sequence takes the form

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{d}^k,$$

where $\mathbf{d}^k$ is a sparse feasible descent direction such that $\|\mathbf{d}^k\|_0 = |P_k|$ with $|P_k| << n$ and α^k represents a stepsize along this direction. Whatever the feasible direction $\mathbf{d}^k$ is, since the objective function is quadratic and convex, the choice of the stepsize can be performed by using an exact minimization of the objective function along $\mathbf{d}^k$. Indeed, let $\bar{\beta} > 0$ be the largest feasible step at $\mathbf{x}^k \in \mathcal{F}$ along the descent direction $\mathbf{d}^k$ then

$$\alpha^k := \min \left\{ -\frac{\nabla f(\mathbf{x}^k)^T \mathbf{d}^k}{{\mathbf{d}^k}^T Q \mathbf{d}^k}, \ \bar{\beta} \right\}. \tag{6}$$

Sequential decomposition methods differ in the choice of the direction $\mathbf{d}^k$, or equivalently in the choice of the so called working set P_k.

SMO-type methods use feasible descent directions $\mathbf{d}^k$ with $\|\mathbf{d}^k\|_0 = 2$ which is the minimal possible cardinality due to the equality constraint. At a feasible point $\mathbf{x} \in \mathcal{F}$ a feasible direction with two nonzero components $\mathbf{d}^{(ij)}$ is given by

$$d_r^{(ij)} := \begin{cases} y_r & \text{if } r = i \\ -y_r & \text{if } r = j \\ 0 & \text{otherwise} \end{cases}, \qquad r = 1, \dots, n, \tag{7}$$

for any pair $(i, j) \in I_{up}(\mathbf{x}) \times I_{low}(\mathbf{x})$ (recalling $\mathbf{y}_i \in \{-1, 1\}$). We say that a pair $(i, j) \in I_{up}(\mathbf{x}) \times I_{low}(\mathbf{x})$ is a violating pair at $\mathbf{x}$ if it satisfies also the descent condition $\nabla f(\mathbf{x})^T \mathbf{d}^{(ij)} < 0$.

The exact optimal stepsize $\alpha \geq 0$ along such a direction $\mathbf{d}^{(ij)}$ can be efficiently computed by noting that in (6) we have

$$\bar{\beta} = \min \{\beta_i, \beta_j\}, \tag{8}$$

where

$$\beta_h := \begin{cases} x_h & \text{if } d_h^{(ij)} < 0 \\ C - x_h & \text{if } d_h^{(ij)} > 0. \end{cases} \tag{9}$$

Thus we get the value of the optimal stepsize α along a direction $\mathbf{d}^{(ij)}$ as

$$\alpha := \min \left\{ -\frac{\nabla f_i y_i - \nabla f_j y_j}{Q_{ii} + Q_{jj} - 2 y_i y_j Q_{ij}}, \bar{\beta} \right\}. \tag{10}$$

Among such minimal descent directions, i.e. violating pairs, a crucial role is played by the so called *Most Violating Pair* (MVP) direction (see e.g. [5]). To be more specific, given a feasible point $\mathbf{x}$, let us define the sets

$$\begin{aligned} I_{up}^{MVP}(\mathbf{x}) &:= \left\{ i \in I_{up}(\mathbf{x}) : \ i \in \arg\max_{h \in I_{up}(\mathbf{x})} -\nabla f(\mathbf{x})_h y_h \right\}, \\ I_{low}^{MVP}(\mathbf{x}) &:= \left\{ j \in I_{low}(\mathbf{x}) : \ j \in \arg\min_{h \in I_{low}(\mathbf{x})} -\nabla f(\mathbf{x})_h y_h \right\}. \end{aligned} \tag{11}$$

If $\mathbf{x}$ is not a solution of problem (2), then $(i_{\texttt{MVP}}, j_{\texttt{MVP}}) \in I_{up}^{MVP}(\mathbf{x}) \times I_{low}^{MVP}(\mathbf{x})$ is a pair, possibly not unique, that violates the KKT conditions at most and it is said a *Most Violating Pair* (MVP). In the sequel, for the sake of notational simplicity, we assume that, for every feasible $\mathbf{x}$, the MVP is unique as this makes no difference in our analysis.

The direction $\mathbf{d}_{\texttt{MVP}} \in \mathbb{R}^n$ corresponding to the pair $(i_{\texttt{MVP}}, j_{\texttt{MVP}}) \in I_{up}^{MVP}(\mathbf{x}) \times I_{low}^{MVP}(\mathbf{x})$ is, among all the feasible descent directions with only two nonzero components, the steepest descent one at $\mathbf{x}$. Most of the convergent decomposition methods for problem (2), both SMO and non-SMO, embed the MVP selection rule in the scheme usually using the value of the objective function obtained by the MVP strategy as a reference value to decide how to perform the next iteration. In our scheme we adopt the MVP rule in a complete different way. To this aim we introduce the definition of "Most Violating step" (MVP step). Let $\mathbf{x}_{\texttt{MVP}} = \mathbf{x} + \alpha_{\texttt{MVP}} \mathbf{d}_{\texttt{MVP}}$ with $\alpha_{\texttt{MVP}}$ obtained by an exact line search as in (10) with $i = i_{\texttt{MVP}}$, $j = j_{\texttt{MVP}}$.

Definition 1 (Most Violating Step) At any feasibile point $\mathbf{x} \in \mathcal{F}$, the *Most Violating Step* (MVP Step) $S_{\texttt{MVP}}$ is defined as:

$$S_{\texttt{MVP}}(\mathbf{x}) := \|\mathbf{x}_{\texttt{MVP}} - \mathbf{x}\| = |\alpha_{\texttt{MVP}}| \|\mathbf{d}_{\texttt{MVP}}\|. \tag{12}$$

In particular, since $y^i \in \{-1, 1\}$ we have that $S_{\texttt{MVP}}(\mathbf{x}) = |\alpha_{\texttt{MVP}}|\sqrt{2}$.

We can state an optimality condition using the definition of MVP step.

Proposition 1 *A point $\mathbf{x}^* \in \mathcal{F}$ is optimal for problem* (2) *if and only if either $I_{up}(\mathbf{x}^*) = \emptyset$ or $I_{low}(\mathbf{x}^*) = \emptyset$ or $S_{MVP}(\mathbf{x}^*) = 0$.*

Proof As said above $\mathbf{x}^*$ is optimal for problem (2) if and only if either $I_{up}(\mathbf{x}^*) = \emptyset$ or $I_{low}(\mathbf{x}^*) = \emptyset$ or condition (5) holds. Therefore we only have to show that, whenever $I_{up}(\mathbf{x}^*) \neq \emptyset$ and $I_{low}(\mathbf{x}^*) \neq \emptyset$, the following holds:

$$S_{\texttt{MVP}}(\mathbf{x}^*) = 0 \quad \Leftrightarrow \quad m(\mathbf{x}^*) \leq M(\mathbf{x}^*).$$

Since $I_{up}(\mathbf{x}^*) \neq \emptyset$ and $I_{low}(\mathbf{x}^*) \neq \emptyset$, we can compute a pair $(i^*_{\texttt{MVP}}, j^*_{\texttt{MVP}}) \in I^{MVP}_{up}(\mathbf{x}^*) \times I^{MVP}_{low}(\mathbf{x}^*)$ and $\mathbf{d}^*_{\texttt{MVP}} = \mathbf{d}^{(i^*_{\texttt{MVP}}, j^*_{\texttt{MVP}})}$ as in (7). The condition $m(\mathbf{x}^*) \leq M(\mathbf{x}^*)$ is equivalent to inequality $\nabla f(\mathbf{x}^*)^T \mathbf{d}^*_{\texttt{MVP}} \geq 0$. By noting that $\mathbf{d}^*_{\texttt{MVP}}$ is a feasible direction at $\mathbf{x}^*$, then from (8) we have $\bar{\beta} > 0$. Therefore by (10) we can conclude that $\nabla f(\mathbf{x}^*)^T \mathbf{d}^*_{\texttt{MVP}} \geq 0$ if and only if $\alpha^*_{\texttt{MVP}} = 0$ and, in turn, if and only if $S_{\texttt{MVP}}(\mathbf{x}^*) = 0$, so that the proof is complete. □

3 A Parallel Decomposition Model

In this section we introduce the parallel decomposition scheme for finding a solution of problem (2). The theoretical properties and implementation details are discussed in the next sections. The algorithm fits in a decomposition framework where, as usual, the solution of problem (2) is obtained by a sequence of solutions of smaller problems in which only subsets of the variables are changed. To fix notation, let $\mathbf{x}^k \in \mathcal{F}$ and consider a subset $P_i \subset \{1, \ldots, n\}$, so that $\mathbf{x}^k$ can be partitioned as $\mathbf{x}^k := (\mathbf{x}^k_{P_i}, \mathbf{x}^k_{-P_i})$. Any subproblem consisting in minimizing the objective function with respect to $\mathbf{x}_{P_i}$ with $\mathbf{x}_{-P_i}$ fixed to the current value $\mathbf{x}^k_{-P_i}$ is:

$$\min_{\mathbf{x}_{P_i} \in \mathcal{F}^k_{P_i}} f_{P_i}(\mathbf{x}_{P_i}, \mathbf{x}^k_{-P_i}) + \frac{\tau^k_i}{2} \|\mathbf{x}_{P_i} - \mathbf{x}^k_{P_i}\|^2, \tag{13}$$

where f_{P_i} denotes the function f restricted to the variables $\mathbf{x}_{P_i}$ when $\mathbf{x}_{-P_i}$ are fixed to the current value. A proximal point term with $\tau^k_i \geq 0$ has been added [27] and the feasible set is

$$\mathcal{F}^k_{P_i} := \{\mathbf{x}_{P_i} \in \mathbb{R}^{|P_i|} : \mathbf{y}^T_{P_i} \mathbf{x}_{P_i} = \mathbf{y}^T_{P_i} \mathbf{x}^k_{P_i},\ 0 \leq \mathbf{x}_{P_i} \leq C\mathbf{e}_{P_i}\}.$$

Problem (13) is still quadratic and convex with Hessian matrix $Q_{P_iP_i} + \tau^k_i I_{P_i}$ symmetric and positive semidefinite, and linear term given by $\ell^k_{P_i} + \tau^k_i x^k_{P_i}$, where

$$\ell^k_{P_i} = \sum_{P_j \in \mathcal{P}, j \neq i} Q_{P_iP_j} \mathbf{x}^k_{P_j} - \mathbf{e}_{P_i}.$$

We denote $\widehat{\mathbf{x}}^k_{P_i}$ as a solution of problem (13), which is unique either if $\tau^k_i > 0$ or if $Q_{P_iP_i}$ is positive definite.

The parallel scheme that we are going to define is not based on splitting the data set or on parallelizing the linear algebra, but on defining a bunch of subproblems to be solved by means of parallel and independent processes. Unlike sequential decomposition methods, the search direction $\mathbf{d}^k$ is obtained by summing up smaller directions obtained by solving in parallel this bunch of subproblems of type (13).

Let us define a partition $\mathcal{P} = \{P_1, P_2, \ldots, P_N\}$ of the set of all indices $\{1, \ldots, n\}$. By definition we have that $P_i \cap P_j = \emptyset$ and $\cup_i P_i = \{1, \ldots, n\}$. The basic idea underlying the definition of the parallel decomposition algorithm is summarized in the following scheme.

Algorithm 1 Parallel Decomposition Model

Initialization Choose $\mathbf{x}^0 \in \mathcal{F}$ and set $k = 0$.
Do while $\mathbf{x}^k$ is not optimal for problem (2).

S.1 (Partition definition)
Set $\mathcal{P}^k = \{P_1, P_2, \ldots, P_{N^k}\}$ and set $\tau^k_i \geq 0$ for all $i = 1, \ldots, N^k$.

S.2 (Blocks selection)
Choose a subset of blocks $\mathcal{J}^k \subseteq \mathcal{P}^k$.

S.3 (Parallel computation)
For all $P_i \in \mathcal{J}^k$ compute in parallel an optimal solution $\widehat{\mathbf{x}}^k_{P_i}$ of problem (13).

S.4 (Direction) Set $\mathbf{d}^k \in \mathbb{R}^n$ block-wise as

$$\mathbf{d}^k_{P_i} = \begin{cases} \widehat{\mathbf{x}}^k_{P_i} - \mathbf{x}^k_{P_i} & \text{if } P_i \in \mathcal{J}^k, \\ \mathbf{0} & \text{otherwise.} \end{cases} \tag{14}$$

S.5 (Stepsize) Choose a suitable stepsize $\alpha^k > 0$.
S.6 (Update) Set $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{d}^k$ and $k = k + 1$.

End While
Return $\mathbf{x}^k$.

We first show that the search direction defined in (14) ia a feasible and descent direction.

Theorem 1 *Let $\mathbf{x}^k \in \mathcal{F}$ and assume that either $Q_{P_iP_i} \succ 0$ for all $P_i \in \mathcal{J}^k$ or $\tau^k_i \geq \underline{\tau} > 0$ for all k.*

Then the direction defined by (14) *is feasible and satisfies*

$$\nabla f(\mathbf{x}^k)^T \mathbf{d}^k \leq -\rho \|\mathbf{d}^k\|^2 \tag{15}$$

with $\rho = \min_{P_i \in \mathcal{J}^k} (\tau^k_i + \lambda^{Q_{P_iP_i}}_{\min}) > 0$.

Proof Note that $\widehat{\mathbf{x}}_{P_i}^k$ satisfies the optimality condition

$$\left[\nabla_{P_i} f_{P_i}(\widehat{\mathbf{x}}_{P_i}^k, \mathbf{x}_{-P_i}^k) + \tau_i^k(\widehat{\mathbf{x}}_{P_i}^k - \mathbf{x}_{P_i}^k)\right]^T \mathbf{d}_{P_i} \geq 0 \tag{16}$$

for any feasible direction $\mathbf{d}_{P_i}$ at $\widehat{\mathbf{x}}_{P_i}^k$. The direction $\mathbf{d}^k$ is partitioned into the blocks $\mathbf{d}_{P_i}^k = \widehat{\mathbf{x}}_{P_i}^k - \mathbf{x}_{P_i}^k$ for $P_i \in \mathcal{J}^k$ and $\mathbf{d}_{P_i}^k = \mathbf{0}_{P_i}^k$ for $P_i \notin \mathcal{J}^k$.

To prove that $\mathbf{d}^k$ belongs to the feasible cone at $\mathbf{x}^k \in \mathcal{F}$

$$\{\mathbf{d} \in \mathbb{R}^n : \ \mathbf{y}^T\mathbf{d} = 0, \ d_i \geq 0, \ \forall\, i : \ x_i^k = 0, \text{ and } d_i \leq 0, \ \forall\, i : \ x_i^k = C, \}.$$

Indeed we get

$$\mathbf{y}^T\mathbf{d}^k = \sum_{P_i \in \mathcal{J}^k} \mathbf{y}_{P_i}{}^T \left(\widehat{\mathbf{x}}_{P_i}^k - \mathbf{x}_{P_i}^k\right) = 0,$$

from the fact that $\mathbf{y}_{P_i}{}^T\widehat{\mathbf{x}}_{P_i}^k = \mathbf{y}_{P_i}{}^T\mathbf{x}_{P_i}^k$. Furthermore, by the feasibility of $\widehat{\mathbf{x}}_j^k$, it holds that

$$d_j^k = \widehat{x}_j^k - x_j^k = \begin{cases} \widehat{x}_j^k - 0 \geq 0 & \text{when } x_j^k = 0 \\ \widehat{x}_j^k - C \leq 0 & \text{when } x_j^k = C \end{cases} \text{ for all } j \in P_i \in \mathcal{J}^k,$$

and, therefore, $\mathbf{d}^k$ is feasibile at $\mathbf{x}^k$.

Now consider

$$\nabla f(\mathbf{x}^k)^T\mathbf{d}^k = \sum_{P_i \in \mathcal{J}^k} \nabla_{P_i} f_{P_i}(\mathbf{x}^k)^T\mathbf{d}_{P_i}^k.$$

Note that the direction $-\mathbf{d}_{P_i}^k$ is a feasible direction at $\widehat{x}_{P_i}^k$ for all $P_i \in \mathcal{J}^k$. Hence by substituting into (16) we can write

$$\nabla_{P_i} f_{P_i}(\widehat{\mathbf{x}}_{P_i}^k, \mathbf{x}_{-P_i}^k)^T\mathbf{d}_{P_i}^k \leq -\tau_i^k\|\mathbf{d}_{P_i}^k\|^2 \tag{17}$$

and it holds that

$$\begin{aligned} \nabla_{P_i} f_{P_i}(\mathbf{x}^k)^T\mathbf{d}_{P_i}^k = \\ -\left(\nabla_{P_i} f_{P_i}(\widehat{\mathbf{x}}_{P_i}^k, \mathbf{x}_{-P_i}^k) - \nabla_{P_i} f_{P_i}(\mathbf{x}^k)\right)^T \mathbf{d}_{P_i}^k + \nabla_{P_i} f_{P_i}(\widehat{\mathbf{x}}_{P_i}^k, \mathbf{x}_{-P_i}^k)^T\mathbf{d}_{P_i}^k \overset{(17)}{\leq} \\ -\mathbf{d}_{P_i}^k{}^T Q_{P_iP_i}\mathbf{d}_{P_i}^k - \tau_i^k\|\mathbf{d}_{P_i}^k\|^2 \leq \\ -\lambda_{\min}^{Q_{P_iP_i}}\|\mathbf{d}_{P_i}^k\|^2 - \tau_i^k\|\mathbf{d}_{P_i}^k\|^2. \end{aligned}$$

Being $\rho = \min_{P_i \in \mathcal{J}^k}(\tau_i^k + \lambda_{\min}^{Q_{P_iP_i}}) > 0$, we have

$$\nabla_{P_i} f_{P_i}(\mathbf{x}_{P_i}^k, \mathbf{x}_{-P_i}^k)^T\mathbf{d}_{P_i}^k \leq -\rho\|\mathbf{d}_{P_i}^k\|^2. \tag{18}$$

and the result follows. □

The scheme above encompasses different possible algorithms depending on the choice of the partition $\mathcal{P}^k$ at **S.1**, the blocks selection $\mathcal{J}^k$ at **S.2** and the stepsize rule at **S.5**.

At **S.1** a partition $\mathcal{P}^k$ of $\{1,\dots,n\}$ is defined. We point out that both the number N^k of the blocks and their composition in $\mathcal{P}^k$ can vary from one iteration to another. For notational simplicity we omit the dependency of the blocks $P_1, P_2, \dots, P_{N^k}$ on the iteration k. As usual in decomposition algorithms, a correct choice of the partition is crucial for proving the global convergence of the method. We discuss this issue in Section 5.

The size of the blocks is a key factor for computational performances. Mainly we can consider two opposite strategies: a SMO-type method that uses blocks of dimension $|P_h| = 2$ or higher dimensional blocks $|P_h| > 2$. In SMO-type methods we can take advantage of the fact that, for each block h, the subproblem (13) can be solved analytically. On the other hand in order to get fast convergence, the simultaneous optimization of a great number of SMO-blocks may be needed. This choice could be well suited for an architecture composed of a great amount of simple processing units, like that of the recent Graphic Processing Units (GPUs). Using higher dimensional blocks may be a suitable choice whenever a few powerful processing units are available. Since the subproblem (13) is still convex quadratic over a simple polyhedron, all the exact or approximate methods can be applied for its solution, even sequential decomposition methods if the blocks are large enough.

Once we have determined a blocks partition of the whole set of variables, only a subset of the resulting subproblems may, in general, be involved in the optimization process. Indeed, we may further restrict the blocks used to update the current iterate by selecting at **S.2** a subset $\mathcal{J}^k$ of the blocks in $\mathcal{P}^k$. These blocks are the only ones used at **S.3** to compute a search direction $\mathbf{d}^k$ according to (14). The selection of blocks makes the algorithmic scheme more flexible since one can set the overall computational burden. The main computational burden is due to the gradient update which may be needed both for checking optimality of the current point $\mathbf{x}^k$ and, as explained in the next sections, for ensuring the convergence (blocks selection). It is well known that, for large scale problems, the gradient update is a big effort due to expensive kernel evaluations. Indeed at each iteration, it requires the computation of the columns of Q related to those variables that are chosen to be in the selected blocks $\mathcal{J}^k$ using the following iterative updating rule

$$\nabla f(\mathbf{x}^{k+1}) = \nabla f(\mathbf{x}^k) + \alpha^k \sum_{P_i \in \mathcal{J}^k} \sum_{h \in P_i} Q_{*h} d_h^k. \tag{19}$$

Hence the choice of $\mathcal{J}^k$ may take into account both the decrease of the objective function and the computational effort for updating the gradient. A minimal threshold on the percentage of objective function decrease associated to each block, with respect to the cumulative decrease of every block, could be a possible discriminant for a block selection rule in order to avoid useless computations.

At **S.3** we obtain an optimal solution $\widehat{\mathbf{x}}^k_{P_i}$ of problem (13) for each $P_i \in \mathcal{J}^k$. The computational burden of this step consists in

- computing the vector $\ell^k_{P_i}$ to construct the objective function of (13) for all blocks $P_i \in \mathcal{J}^k$,
- solving the $|\mathcal{J}^k|$ subproblems.

These $|\mathcal{J}^k|$ convex quadratic problems can be distributed to different processes in order to be solved in a parallel fashion.

At **S.4** the algorithm computes the search direction $\mathbf{d}^k$.

At **S.5** the algorithm computes the stepsize α^k.

We show in the next section (Theorems 2, 3 and 4) that, in order to have convergence of the algorithm, α^k can be computed according to a simple diminishing stepsize rule or a linesearch procedure (including the exact minimization rule). We remark that the use of a diminishing stepsize rule as theoretical tool to prove convergence is widely employed in recent algorithms for Machine Learning. See e.g. the comprehensive surveys on optimization methods [2] and the papers [8,29].

4 Theoretical Analysis

We analyse the theoretical properties of Algorithm 1. We first show in this section that under suitable assumptions the sequence $\{\mathbf{x}^k\}$ produced by the algorithm satisfies

$$\lim_{k\to\infty} S_{\texttt{MVP}}(\mathbf{x}^k) = 0. \tag{20}$$

However this is not enough to guarantee the asymptotic convergence of $\{\mathbf{x}^k\}$ to a solution of problem (2). This is due to the implicit discontinuous nature of the indices sets I_{up} and I_{low} that enters in the definition of $S_{\texttt{MVP}}(\mathbf{x}^k)$. We discuss these aspects in Section 6.

We introduce the definition of descent block and of descent iteration that play a fundamental role in the convergence analysis.

Definition 2 (Descent block) Given $\epsilon > 0$. At a feasible point $\mathbf{x}^k$, we say that the block of variables $P_i \subseteq \{1,\ldots,n\}$ is a *descent block* if it satisfies

$$\|\widehat{\mathbf{x}}^k_{P_i} - \mathbf{x}^k_{P_i}\| \geq \epsilon S_{\texttt{MVP}}(\mathbf{x}^k), \tag{21}$$

where $\widehat{\mathbf{x}}^k_{P_i}$ is the optimal solution of the corresponding problem (13) and $S_{\texttt{MVP}}(\mathbf{x}^k)$ is defined in (12).

Definition 3 (Descent iteration) An iteration k of Algorithm 1 is said a *descent iteration* if the set $\mathcal{J}^k$, selected at **S.2**, contains at least one descent block P_i at $\mathbf{x}^k$.

Under the assumption that at least one descent block is selected for optimization at **S.2** of the parallel algorithmic model, we will prove that by using

a suitable α^k at **S.5** the sequence $\{\mathbf{x}^k\}$ produced by the algorithm satisfies (20), i.e.

$$\lim_{k\to\infty} S_{\texttt{MVP}}(\mathbf{x}^k) = 0.$$

In the next section, we prove that this assumption is easy to achieve.

We first consider the case when the stepsize α^k is determined by a standard Armjio linesearch procedure along the direction $\mathbf{d}^k$.

Theorem 2 *Let $\{\mathbf{x}^k\}$ be the sequence generated by Algorithm 1 where $\alpha^k \leq 1$ at* **S.5** *satisfies the following Armjio condition*

$$f(\mathbf{x}^k + \alpha^k \mathbf{d}^k) \leq f(\mathbf{x}^k) + \theta \alpha^k \nabla f(\mathbf{x}^k)^T \mathbf{d}^k, \tag{22}$$

with $\theta \in (0,1)$. Assume that for all k

(i) *there exists an integer $\tilde{k}$, with $k \leq \tilde{k} \leq L + k$ for a finite $L \geq 0$, such that $\tilde{k}$ is a descent iteration;*
(ii) *either $Q_{P_i P_i} \succ 0$, or $\tau_i^k \geq \underline{\tau} > 0$, for all $P_i \in \mathcal{J}^k$.*

If Algorithm 1 does not terminate in a finite number of iterations to a solution of problem (2)*, then $\{\mathbf{x}^k\}$ admits a limit point satisfying* (20)*.*

Proof Let us assume that an infinite sequence $\{\mathbf{x}^k\}$ is generated. Since $\alpha^k \leq 1$, from Theorem 1 we know that $\{\mathbf{x}^k\}$ is a feasible sequence and that (18) holds. From conditions (22) and (18) we can write

$$f(\mathbf{x}^{k+1}) - f(\mathbf{x}^k) \leq -\alpha^k \theta \rho \|\mathbf{d}^k\|^2. \tag{23}$$

Therefore the sequence $\{f(\mathbf{x}^k)\}$ is decreasing and bounded below, so that it converges and we get

$$\lim_{k\to\infty} (f(\mathbf{x}^{k+1}) - f(\mathbf{x}^k)) = 0. \tag{24}$$

Let $\bar{\mathbf{x}}$ be a limit point of $\{\mathbf{x}^k\}$, at least one of such points exists being $\mathcal{F}$ compact. Since, by the compactness of $\mathcal{F}$ and by the continuity of f, $-\infty < f(\bar{\mathbf{x}}) - f(\mathbf{x}^0)$ for all $\mathbf{x}^0 \in \mathcal{F}$, then, by (23) and (24), we can write

$$\sum_{k=0}^{\infty} \alpha^k \|\mathbf{d}^k\|^2 < +\infty. \tag{25}$$

By condition (i) we can define an infinite subsequence $\{x^k\}_{\widetilde{K}}$ made up of only descent iterations. Then, by (25), it follows that

$$\sum_{k=0, k\in\widetilde{K}}^{\infty} \alpha^k \|\mathbf{d}^k\|^2 < +\infty. \tag{26}$$

By standard arguments we know that a stepsize satisfying the Armijo rule (22) produces at each iteration an $\alpha^k > 0$ (see [1]) such that

$$\sum_{k=0, k\in\widetilde{K}}^{\infty} \alpha^k = +\infty. \tag{27}$$

By (26) and (27), we obtain

$$\liminf_{k\to\infty, k\in\widetilde{K}} \|\mathbf{d}^k\| = 0.$$

Now since each $\mathcal{J}^k$ with $k \in \widetilde{K}$ contains a descent block at $\mathbf{x}^k$, by (21) we can conclude that

$$\liminf_{k\to\infty, k\in\widetilde{K}} S_{\texttt{MVP}}(\mathbf{x}^k) = 0,$$

and then, since $S_{\texttt{MVP}}(\mathbf{x}^k) \geq 0$ for all k, we can write

$$\liminf_{k\to\infty} S_{\texttt{MVP}}(\mathbf{x}^k) = 0. \tag{28}$$

Now suppose by contradiction that $\limsup_{k\to\infty} S_{\texttt{MVP}}(\mathbf{x}^k) > 0$, then for any $\gamma > 0$ sufficiently small we would have $S_{\texttt{MVP}}(\mathbf{x}^k) > \gamma$ for infinitely many k and $S_{\texttt{MVP}}(\mathbf{x}^k) < \frac{\gamma}{2}$ for infinitely many k. Therefore, one can always find an infinite set of indices, say $\mathcal{N}$, having the following property: for any $n \in \mathcal{N}$, there exists an integer $i_n > n$ such that $S_{\texttt{MVP}}(\mathbf{x}^n) < \frac{\gamma}{2}$ and $S_{\texttt{MVP}}(\mathbf{x}^{i_n}) > \gamma$. Then it is easy to see that $\mathbf{x}^n \neq \mathbf{x}^{i_n}$ and then $\sum_{k=n}^{i_n-1} \alpha^k \|\mathbf{d}^k\| > 0$ for all $n \in \mathcal{N}$. And then

$$\liminf_{n\in\mathcal{N}, n\to\infty} \sum_{k=n}^{i_n-1} \alpha^k \|\mathbf{d}^k\| > 0,$$

which is in contradiction with (25). Then we finally obtain (20). □

Note that the same result holds if we choose any $\mathbf{x}^{k+1}$ satisfying $f(\mathbf{x}^{k+1}) \leq f(\mathbf{x}^k + \alpha^k \mathbf{d}^k)$ being α^k the Armijo stepsize. In particular $\mathbf{x}^{k+1}$ can be determined by performing an exact linesearch along $\mathbf{d}^k$. Since f is quadratic, an exact minimization along direction $\mathbf{d}^k$ can be performed analytically:

$$\alpha^k := \max\left\{\min\left\{-\frac{\nabla f(\mathbf{x}^k)^T \mathbf{d}^k}{\mathbf{d}^{k\,T} Q \mathbf{d}^k}, \overline{\alpha}^k\right\}, 0\right\}, \tag{29}$$

where

$$\overline{\alpha}^k := \min_{i\in\{1,\ldots,n\}:\mathbf{d}_i^k\neq 0} \left\{\overline{\alpha}_i^k = \begin{cases} \mathbf{x}_i^k & \text{if } \mathbf{d}_i^k < 0 \\ C - \mathbf{x}_i^k & \text{if } \mathbf{d}_i^k > 0 \end{cases}\right\}.$$

Note that in this case it is not necessary to impose $\alpha^k \leq 1$ since the feasibility is guaranteed by construction.

If n is huge, performing either an exact line search or an Armijo one may be computationally expensive. We propose to save computations by using a diminishing stepsize strategy. We give two different convergence results based on slightly different hypotheses.

Theorem 3 *Let $\{\mathbf{x}^k\}$ be the sequence generated by Algorithm 1 where at* **S.5** *$\alpha^k \in (0, 1]$ satisfies the following condition*

$$\alpha^k \to 0 \text{ and } \sum_{k=0}^{\infty} \alpha^k = +\infty. \tag{30}$$

Assume that for all k

(i) *k is a descent iteration;*
(ii) *either $Q_{P_iP_i} \succ 0$, or $\tau_i^k \geq \underline{\tau} > 0$, for all $P_i \in \mathcal{J}^k$.*

If Algorithm 1 does not terminate in a finite number of iterations to a solution of problem (2)*, then $\{\mathbf{x}^k\}$ admits a limit point satisfying* (20)*.*

Proof Assume that an infinite sequence is generated. Since $\alpha^k \leq 1$, from Theorem 1 we know that $\{\mathbf{x}^k\}$ is a feasible sequence and that (18) holds.

Since f is quadratic, for any $k \geq 0$ we can write (Descent Lemma [1]):

$$f(\mathbf{x}^{k+1}) - f(\mathbf{x}^k) \leq \alpha^k \nabla f(\mathbf{x}^k)^T \mathbf{d}^k + \frac{1}{2}(\alpha^k)^2 \lambda_{\max}^Q \|\mathbf{d}^k\|^2. \tag{31}$$

By using (ii) and (18) we can rewrite inequality (31):

$$f(\mathbf{x}^{k+1}) - f(\mathbf{x}^k) \leq \alpha^k \left(-\rho + \frac{\alpha^k \lambda_{\max}^Q}{2} \right) \|\mathbf{d}^k\|^2. \tag{32}$$

Since, by (30), $\alpha^k \to 0$ it follows that there exist $\bar{\rho} > 0$ and $\bar{k}$ sufficiently large such that for all $k \geq \bar{k}$ inequality (32) implies:

$$f(\mathbf{x}^{k+1}) - f(\mathbf{x}^k) \leq -\alpha^k \bar{\rho} \|\mathbf{d}^k\|^2.$$

Since, as said in the proof of Theorem 2, $-\infty < f(\bar{\mathbf{x}}) - f(\mathbf{x}^{\bar{k}})$ for all $\mathbf{x}^{\bar{k}} \in \mathcal{F}$ and any limit point $\bar{\mathbf{x}}$ of $\{\mathbf{x}^k\}$, in a similar way we can write

$$\sum_{k=\bar{k}}^{\infty} \alpha^k \|\mathbf{d}^k\|^2 < +\infty. \tag{33}$$

By (30), $\sum_{k=\bar{k}}^{\infty} \alpha^k = +\infty$, and then we obtain

$$\liminf_{k\to\infty} \|\mathbf{d}^k\| = 0.$$

Now since each $\mathcal{J}^k$ contains a descent block at $\mathbf{x}^k$, by (21) we can conclude that

$$\liminf_{k\to\infty} S_{\texttt{MVP}}(\mathbf{x}^k) = 0,$$

and the thesis follows under the same reasoning of the proof of Theorem 2. □

As stated in Theorem 3, a diminishing stepsize rule requires all iterations to be descent. In some cases (i.e. when the variables are randomly partitioned), it could be useful to relax this condition, requiring that only a subsequence of the iterations are descent, as well as for Theorem 2. This is formalized in the next theorem where we assume the additional mild hypothesis of monotonicity of the sequence $\{\alpha^k\}$.

Theorem 4 *Let $\{\mathbf{x}^k\}$ be the sequence generated by Algorithm 1 where at* **S.5** *$\alpha^k \in (0, 1]$ satisfies* (30)*. Assume that for all k*

(i) *there exists an integer $\tilde{k}$, with $k \leq \tilde{k} \leq L + k$ for a finite $L \geq 0$, such that $\tilde{k}$ is a descent iteration;*

(ii) *either* $Q_{P_iP_i} \succ 0$, *or* $\tau_i^k \geq \underline{\tau} > 0$, *for all* $P_i \in \mathcal{J}^k$;
(iii) $\alpha^k \geq \alpha^{k+1}$.

If Algorithm 1 does not terminate in a finite number of iterations to a solution of problem (2), *then* $\{\mathbf{x}^k\}$ *admits a limit point satisfying* (20).

Proof Following the same reasoning of Theorem 3, inequality (33) holds. By condition (i) we can define an infinite subsequence $\{k\}_{\tilde{K}}$ containing only descent iterations. Then by (33) it follows that

$$\sum_{k=\bar{k},k\in\tilde{K}}^{\infty} \alpha^k \|\mathbf{d}^k\|^2 < +\infty. \tag{34}$$

We can write the following chain of inequalities

$$+\infty = \sum_{k=\bar{k}+1}^{\infty} \sum_{h=0}^{L-1} \alpha^{L\cdot k+h} \leq L \sum_{k=\bar{k}+1}^{\infty} \alpha^{L\cdot k} \leq L \sum_{k=\bar{k},k\in\tilde{K}}^{\infty} \alpha^k,$$

where the equality is due to (30), the first inequality holds by (iii) and the second inequality holds by (i) and (iii). Then the thesis follows from the same reasoning of the proof of Theorem 2. □

A simple dminishing rule could be $\alpha^k = \frac{1}{k^\xi}$, with $\xi \in (0, 1]$, but different choices are also possible. Although preliminary tests showed that the exact minimization is more effective than any other choice, the diminishing stepsize strategy, besides being easy to implement, requires much less computations and this could be of great practical interest for high dimensional instances (training sets with many samples and many dense features).

5 Construction of the Partitions

To make the results stated in the previous section of practical interest, the major difficulty is to state conditions that ensure that a descent iteration is generated according to Definition 3.

We prove that the MVP can be useful to construct a descent iteration. To this aim we first prove, in the next lemma, that there is a relation between the steplenght produced optimizing over a generic block P_h and the one produced optimizing over any violating pair (i, j) belonging to P_h. This result will be useful in order to practically build a descent block and it is used in Theorem 5.

In this section we use the simplified assumption that any principal submatrix of Q of order 2 is positive definite.

Lemma 1 *Assume that any principal submatrix* Q *of order* 2 *is positive definite. Let* $\mathbf{x}^k \in \mathcal{F}$ *be a feasible point for problem* (2) *and let* $(i, j) \in I_{up}(\mathbf{x}^k) \times$

$I_{low}(\mathbf{x}^k)$ be such that $\nabla f(\mathbf{x}^k)^T\mathbf{d}^{(ij)} < 0$ where $\mathbf{d}^{(ij)}$ is defined as in (7). Suppose that a block $P_h \subseteq \{1,\dots,n\}$ exists such that $(i,j) \subseteq P_h$. Let $\overline{\mathbf{x}}^k$ be the unique solution of

$$\min_{\mathbf{x}_{\{i,j\}} \in \mathcal{F}_{\{i,j\}}} f_{\{i,j\}}\left(\mathbf{x}_{\{i,j\}}, \mathbf{x}^k_{-\{i,j\}}\right). \tag{35}$$

Then there exists a scalar $\bar{\epsilon} > 0$ such that

$$\|\widehat{\mathbf{x}}^k_{P_h} - \mathbf{x}^k_{P_h}\| \geq \bar{\epsilon}\|\overline{\mathbf{x}}^k - \mathbf{x}^k\|, \tag{36}$$

where $\widehat{\mathbf{x}}^k_{P_h}$ is a solution of problem (13).

Proof First we note that

$$\overline{\mathbf{x}}^k = \mathbf{x}^k + \bar{\alpha}\mathbf{d}^{(i,j)}$$

with $\mathbf{d}^{(i,j)}$ defined in (7) and $\bar{\alpha} > 0$ computed as in (10). For the sake of notational simplicity, we set $\overline{\mathbf{d}} = \mathbf{d}^{(i,j)}$ and $\overline{\mathbf{d}}_{P_h}$ is the corresponding subvector in $\mathbb{R}^{|P_h|}$.

Two cases are possible:

(a) $\widehat{\mathbf{x}}^k_{P_h} + \mu\overline{\mathbf{d}}_{P_h} \notin \mathcal{F}_{P_h}$ for all $\mu > 0$, or
(b) $\mu > 0$ exists such that $\widehat{\mathbf{x}}^k_{P_h} + \mu\overline{\mathbf{d}}_{P_h} \in \mathcal{F}_{P_h}$, that is $\overline{\mathbf{d}}_{P_h}$ is a feasible direction at $\widehat{\mathbf{x}}^k_{P_h}$.

(a) By construction it holds that $\mathbf{y}^T_{P_h}\overline{\mathbf{d}}_{P_h} = 0$. Then it holds that for all $\mu > 0$:

$$\mathbf{y}^T_{P_h}(\widehat{\mathbf{x}}^k_{P_h} + \mu\overline{\mathbf{d}}_{P_h}) = \mathbf{y}^T_{P_h}\widehat{\mathbf{x}}^k_{P_h} = \mathbf{y}^T_{P_h}\mathbf{x}^k_{P_h},$$

where the last equality holds since $\widehat{\mathbf{x}}^k_{P_h} \in \mathcal{F}_{P_h}$. Therefore we can conclude that for all $\mu > 0$:

$$\widehat{\mathbf{x}}^k_{P_h} + \mu\overline{\mathbf{d}}_{P_h} \notin [0, C]^{|P_h|},$$

and then either $\widehat{x}^k_i$ or $\widehat{x}^k_j$ must be on a bound. In particular, supposing ,without loss of generality, that i is the component on the bound, if $\overline{d}_i > 0$ then $\widehat{x}^k_i = C$ and then we can write

$$0 < \bar{\alpha}\overline{d}_i = \overline{x}^k_i - x^k_i \leq C - x^k_i = \widehat{x}^k_i - x^k_i;$$

otherwise $\overline{d}_i < 0$ then $\widehat{x}^k_i = 0$ and then

$$0 > \bar{\alpha}\overline{d}_i = \overline{x}^k_i - x^k_i \geq 0 - x^k_i = \widehat{x}^k_i - x^k_i.$$

In both cases it holds that

$$|\bar{\alpha}\overline{d}_i| \leq |\widehat{x}^k_i - x^k_i|.$$

Therefore noting that $|\bar{\alpha}\overline{d}_i| = |\bar{\alpha}|$ and that $|\bar{\alpha}|\sqrt{2} = \|\bar{\mathbf{x}}^k - \mathbf{x}^k\|$ we can conclude that

$$\|\widehat{\mathbf{x}}^k_{P_h} - \mathbf{x}^k_{P_h}\| \geq |\bar{\alpha}| = \frac{1}{\sqrt{2}}\|\overline{\mathbf{x}}^k - \mathbf{x}^k\|.$$

(b) Since $\overline{\mathbf{d}}_{P_h}$ is a feasible direction at $\widehat{\mathbf{x}}^k_{P_h}$ and since $\widehat{\mathbf{x}}^k_{P_h}$ is an optimal solution of problem (13) at $\mathbf{x}^k$, by (16), we can write

$$\left[\nabla_{P_h} f_{P_h}\left(\widehat{\mathbf{x}}^k_{P_h}, \mathbf{x}^k_{-P_h}\right) + \tau_i^k(\widehat{\mathbf{x}}^k_{P_h} - \mathbf{x}^k_{P_h})\right]^T \overline{\mathbf{d}}_{P_h} \geq 0. \tag{37}$$

Since $\overline{\mathbf{x}}^k$ is a solution of (35), and being $-\overline{\mathbf{d}}$ a feasible direction for (35) at $\overline{\mathbf{x}}^k$, then, by the minimum principle and since $(i,j) \subseteq P_h$, we can write

$$\nabla_{P_h} f_{P_h}\left(\overline{\mathbf{x}}^k_{P_h}, \mathbf{x}^k_{-P_h}\right)^T \overline{\mathbf{d}}_{P_h} \leq 0.$$

And therefore by (37) we can write

$$\left[\nabla_{P_h} f_{P_h}\left(\widehat{\mathbf{x}}^k_{P_h}, \mathbf{x}^k_{-P_h}\right) + \tau_h^k(\widehat{\mathbf{x}}^k_{P_h} - \mathbf{x}^k_{P_h})\right]^T \overline{\mathbf{d}}_{P_h} \geq \nabla_{P_h} f_{P_h}\left(\overline{\mathbf{x}}^k_{P_h}, \mathbf{x}^k_{-P_h}\right)^T \overline{\mathbf{d}}_{P_h}. \tag{38}$$

By assumptions, $\sigma > 0$ exists such that

$$\begin{aligned}\sigma\|\overline{\mathbf{x}}^k_{P_h} - \mathbf{x}^k_{P_h}\|^2 &\leq (\overline{\mathbf{x}}^k_{P_h} - \mathbf{x}^k_{P_h})^T Q_{P_h P_h}(\overline{\mathbf{x}}^k_{P_h} - \mathbf{x}^k_{P_h}) \\ &= \left[\nabla_{P_h} f_{P_h}\left(\overline{\mathbf{x}}^k_{P_h}, \mathbf{x}^k_{-P_h}\right) - \nabla_{P_h} f_{P_h}\left(\mathbf{x}^k_{P_h}, \mathbf{x}^k_{-P_h}\right)\right]^T \bar{\alpha}\overline{\mathbf{d}}_{P_h}.\end{aligned} \tag{39}$$

Then combining (38) and (39) we can write

$$\begin{aligned}&\sigma\|\overline{\mathbf{x}}^k_{P_h} - \mathbf{x}^k_{P_h}\|^2 \leq \tau_h^k(\widehat{\mathbf{x}}^k_{P_h} - \mathbf{x}^k_{P_h})^T \bar{\alpha}\overline{\mathbf{d}}_{P_h} + \\ &\left[\nabla_{P_h} f_{P_h}\left(\widehat{\mathbf{x}}^k_{P_h}, \mathbf{x}^k_{-P_h}\right) - \nabla_{P_h} f_{P_h}\left(\mathbf{x}^k_{P_h}, \mathbf{x}^k_{-P_h}\right)\right]^T \bar{\alpha}\overline{\mathbf{d}}_{P_h} \leq \\ &(\tau_h^k + \|Q_{P_h P_h}\|)\|\widehat{\mathbf{x}}^k_{P_h} - \mathbf{x}^k_{P_h}\|\|\bar{\alpha}\overline{\mathbf{d}}_{P_h}\| = \\ &(\tau_h^k + \lambda^Q_{\max})\|\widehat{\mathbf{x}}^k_{P_h} - \mathbf{x}^k_{P_h}\|\|\overline{\mathbf{x}}^k_{P_h} - \mathbf{x}^k_{P_h}\|.\end{aligned}$$

Therefore we obtain

$$\|\widehat{\mathbf{x}}^k_{P_h} - \mathbf{x}^k_{P_h}\| \geq \frac{\sigma}{\tau_h^k + \lambda^Q_{\max}}\|\overline{\mathbf{x}}^k_{P_h} - \mathbf{x}^k_{P_h}\| = \frac{\sigma}{\tau_h^k + \lambda^Q_{\max}}\|\overline{\mathbf{x}}^k - \mathbf{x}^k\|,$$

and finally we have the proof.

□

Theorem 5 *Assume that any principal submatrix of Q of order 2 is positive definite. Let $\mathbf{x}^k \in \mathcal{F}$ be a feasible point for problem (2) and let $(i,j) \in I_{up}(\mathbf{x}^k) \times I_{low}(\mathbf{x}^k)$ be such that $\nabla f(\mathbf{x}^k)^T \mathbf{d}^{(ij)} < 0$ where $\mathbf{d}^{(ij)}$ is defined as in (7). Suppose that a block $P_h \subseteq \{1,\ldots,n\}$ exists such that $(i,j) \subseteq P_h$. Let $\overline{\mathbf{x}}^k$ be the unique solution of problem (35) and suppose that $\widetilde{\epsilon} > 0$ exists such that*

$$\|\overline{\mathbf{x}}^k - \mathbf{x}^k\| \geq \widetilde{\epsilon} S_{MVP}(\mathbf{x}^k). \tag{40}$$

Then P_h is a descent block.

Proof By Lemma 1 we know that (36) holds. Therefore by combining (36) and (40) we obtain the proof. □

Theorem 5 shows that we can build a descent block at the cost of computing a pair that satisfies (40). Clearly the MVP does it, but it is easy to see that any pair that "sufficiently" violates KKT conditions can be used as well.

Now we give a further theoretical result which guarantees that at each iteration of Algorithm 1 at least one descent block can be built.

Theorem 6 *Let* $\mathbf{x}^k$ *be a feasible, but not optimal, point for problem* (2) *then at least one descent block* $P_h \subseteq \{1,\dots,n\}$ *exists.*

Proof By Proposition 1, if $\mathbf{x}^k$ is not optimal then $I_{up}(\mathbf{x}^k) \neq \emptyset$, $I_{low}(\mathbf{x}^k) \neq \emptyset$ and $S_{\texttt{MVP}}(\mathbf{x}^k) > 0$. Therefore any $P_h \supseteq (i_{\texttt{MVP}}, j_{\texttt{MVP}})$ is a descent block. □

6 Global Convergence

So far we have proved that, under some suitable conditions, Algorithm 1 either converges in a finite number of iterations to a solution of problem (2) or the produced sequence $\{\mathbf{x}^k\}$ satisfies (20). However the fact that $S_{\texttt{MVP}}(\mathbf{x}^k)$ goes to zero is not enough to guarantee the asymptotic convergence of Algorithm 1 to a solution of problem (2). This is due to the discontinuous nature of the indices sets I_{up} and I_{low} that enters in the definition of $S_{\texttt{MVP}}(\mathbf{x}^k)$. Actually, this is a well known theoretical issue in decomposition methods for the SVM training problem. Indeed, even in the case when the algorithm were proved to asymptotically converge to an optimal solution, the validity of a stopping criterion based on the KKT conditions (5) must be verified [21]. A possible way to sorting out these theoretical issues is to use some theoretical tricks. For example by properly inserting some standard MVP iterations in the produced sequence $\{\mathbf{x}^k\}$ [25] or by dealing with $\epsilon-$solutions [18]. All these theoretical efforts can be encompassed in a realistic numerical setting. Indeed all the papers discussing about decomposition methods rely on the fact that the indices sets I_{up} and I_{low} can be computed in exact arithmetic. In practice what it can actually be computed are the following $\epsilon-$perturbations of the sets I_{up} and I_{low}

$$I^{\epsilon}_{up}(\mathbf{x}) := \{r \subseteq \{1,\dots,n\} : \ x_r \leq C-\epsilon, \ y_r = 1, \ \text{or } x_r \geq \epsilon, \ y_r = -1\},$$

$$I^{\epsilon}_{low}(\mathbf{x}) := \{r \subseteq \{1,\dots,n\} : \ x_r \leq C-\epsilon, \ y_r = -1, \ \text{or } x_r \geq \epsilon, \ y_r = 1\},$$

with $\epsilon > 0$. Consequently we can define at a feasible point $\mathbf{x}$ the following quantities

$$m^{\epsilon}(\mathbf{x}) = \max_{r \in I^{\epsilon}_{up}(\mathbf{x})} -\nabla f(\mathbf{x})_r y_r, \qquad M^{\epsilon}(\mathbf{x}) = \min_{r \in I^{\epsilon}_{low}(\mathbf{x})} -\nabla f(\mathbf{x})_r y_r.$$

and the corresponding sets $I^{\epsilon,MVP}_{up}(\mathbf{x})$ and $I^{\epsilon,MVP}_{low}(\mathbf{x})$ obtained by using the $\epsilon-$perturbations of sets I_{up} and I_{low} in (11).

As a matter-of-fact an effective optimality condition which can be used is

$$m^{\epsilon}(\mathbf{x}^k) \leq M^{\epsilon}(\mathbf{x}^k) + \eta, \tag{41}$$

where $\eta > 0$ is a given tolerance. Note that any asymptotically convergent decomposition algorithm can actually converge only to a point satisfying (41), rather than (5).

It is easy to see that $I^{\epsilon}_{up}(\mathbf{x}) \subseteq I_{up}(\mathbf{x})$ and $I^{\epsilon}_{low}(\mathbf{x}) \subseteq I_{low}(\mathbf{x})$ for all $\epsilon > 0$. Furthemore in [28], it has been proved the following result.

Proposition 2 *Let $\{\mathbf{x}^k\}$ be a sequence of feasible points converging to a point $\overline{\mathbf{x}} \in \mathcal{F}$. Then, there exists a scalar $\bar{\epsilon} > 0$ (depending only on $\overline{\mathbf{x}}$) such that for every $\epsilon \in (0, \bar{\epsilon}]$ there exists an index $\bar{k} = k_\epsilon$ for which*

$$I^{\epsilon}_{up}(\mathbf{x}^k) \equiv I_{up}(\mathbf{x}^k) \qquad \text{and} \qquad I^{\epsilon}_{low}(\mathbf{x}^k) \equiv I_{low}(\mathbf{x}^k) \qquad \textit{for all } k \geq \bar{k}.$$

This proposition allows to state that for k sufficiently large and ϵ sufficiently small using the index sets $I^{\epsilon}_{up}(\mathbf{x})$ and $I^{\epsilon}_{low}(\mathbf{x})$ is equivalent to using the exact ones I_{up} and I_{low} and we have that

$$m^{\epsilon}(\mathbf{x}) = m(\mathbf{x}) \qquad \text{and} \qquad M^{\epsilon}(\mathbf{x}) = M(\mathbf{x}),$$

so that, for any $\epsilon \in (0, \bar{\epsilon}]$ and $k \geq \bar{k}$, (41) reduces to the concept of η-optimal solution introduced in [18]. However, this is not true either when $\mathbf{x}$ is far from a solution or when we set an ϵ that is not small enough, being $\bar{\epsilon}$ unknown. Reducing ϵ to the machine precision ϵ_{mach} is the best that we can do in a numerical implementation, so that one can argue that for $\epsilon = \epsilon_{\text{mach}}$ if $I^{\epsilon}_{up}(\mathbf{x}) = \emptyset$ or $I^{\epsilon}_{low}(\mathbf{x}) = \emptyset$, a solution has been reached within the possible tolerance.

Given a point $\mathbf{x}^k \in \mathcal{F}$, we consider the MVP $\epsilon-$step $S^{\epsilon}_{\texttt{MVP}}(\mathbf{x}^k)$ obtained by using $I^{\epsilon}_{up}(\mathbf{x}^k)$ and $I^{\epsilon}_{low}(\mathbf{x}^k)$ instead of $I_{up}(\mathbf{x}^k)$ and $I_{low}(\mathbf{x}^k)$ in the defintion (12). As a consequence of the definition itself, for any MVP $\epsilon-$direction $\mathbf{d}^k_{\texttt{MVP},\epsilon}$ we get that the feasible $\epsilon-$stepsize $\bar{\beta}^k_\epsilon$ defined as in (8) remains bounded from zero by ϵ.

It is easy to see that all results stated so far for Algorithm 1 are still valid if we consider the $\epsilon-$definition $S^{\epsilon}_{\texttt{MVP}}(\mathbf{x}^k)$ rather than $S_{\texttt{MVP}}(\mathbf{x}^k)$. Furthemore we have the following result, that fills the gap of convergence.

Theorem 7 *Let $\epsilon > 0$ and $\eta > 0$ be given. Let $\{\mathbf{x}^k\}$ be a sequence of feasible points such that $I^{\epsilon}_{up}(\mathbf{x}^k) \neq \emptyset$, $I^{\epsilon}_{low}(\mathbf{x}^k) \neq \emptyset$ and*

$$\lim_{k\to\infty} S^{\epsilon}_{MVP}(\mathbf{x}^k) = 0.$$

Then $\bar{k} > 0$ exists such that, for all $k \geq \bar{k}$, $\mathbf{x}^k$ satisfies (41).

Proof By definition of $S^{\epsilon}_{\texttt{MVP}}(\mathbf{x}^k)$ we get

$$0 = \lim_{k\to\infty} S^{\epsilon}_{\texttt{MVP}}(\mathbf{x}^k) = \sqrt{2} \lim_{k\to\infty} |\alpha^k_{\texttt{MVP},\epsilon}|. \tag{42}$$

Since by construction $\bar{\beta}^k_\epsilon \geq \epsilon$, by (6) we get that (42) implies that $\bar{k} > 0$ exists such that, for all $k \geq \bar{k}$, we have $-\nabla f(\mathbf{x}^k)^T \mathbf{d}^k_{\texttt{MVP},\epsilon} \leq \eta$, which implies (41). □

7 Practical Algorithmic Realizations

Algorithm 1 includes a vast amount of specific strategies that may vary according to different implementation choices. Various alternatives may be related to the blocks dimension, the blocks composition, the blocks selection, the way to enforce the convergence conditions and the methods used to solve the subproblems. Different algorithms can be designed exploiting these degrees of freedom. In this section we propose a possible implementable scheme of Algorithm 1 and we apply it to some SVM training problems. We included some numerical results on a small benchmark for SVMs training with the sole aim of showing the viability and the effectiveness of the proposed scheme. We choose a SMO-type parallel scheme derived from Algorithm 1, that we called PARSMO, for which we developed two Matlab prototypes that differ in the block selection. Actually we do not realize a truly parallel implementation, but we simply realize sequential prototypes with the aim of highlighting the benefits of simultaneously moving along multiple SMO directions. Of course this gives only a flavour of the actual CPU-time saving of a parallel implementation because communication time among processor must be taken into account. Indeed, being a SMO-type implementation, the computational effort of each processor is very light and the communications must be very fast thus being suitable for a multicore environment. In order to enforce convergence, following Theorem 6, we include an MVP pair among the blocks selected at each iteration.

The PARSMO scheme is reported in Algorithm 2.

Algorithm 2 PARSMO

Initialization Set $\mathbf{x}^0 = 0$, $\nabla f^0 = -\mathbf{e}$, $q \geq 1$, $\epsilon > 0$, $\eta > 0$ and $k = 0$.
Select
$$(i_1, j_1) \in I_{up}^{\epsilon,MVP}(\mathbf{x}^k) \times I_{low}^{\epsilon,MVP}(\mathbf{x}^k).$$

Do while $\left(-\nabla f_{i_1}^k y_{i_1} + \nabla f_{j_1}^k y_{j_1} \geq \eta\right)$

S.1 (Blocks definition)
Choose $(q-1)$ pairs $\{(i_2, j_2), (i_3, j_3), \ldots, (i_q, i_q)\}$.
Set $\mathcal{J}^k = \{(i_1, j_1), (i_2, j_2), \ldots, (i_q, i_q)\}$.

S.2 (Parallel computation)
For each pair $(i_h, j_h) \in \mathcal{J}^k$ compute in parallel:
1. kernel columns Q_{*i_h} and Q_{*j_h} (if not available in the cache),
2. $t_h \mathbf{d}^{(i_h j_h)}$ with $\mathbf{d}^{(i_h j_h)}$ defined as in (7) and t_h as in (10).

S.3 (Direction) $\mathbf{d}^k = \sum_{(i_h, j_h) \in \mathcal{J}^k} t_h \mathbf{d}^{(i_h j_h)}$

S.4 (Stepsize) Compute the steplenght α^k as in (29).

S.5 (Update)
Set $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{d}^k$.
Set $\nabla f^{k+1} = \nabla f^k + \alpha^k \sum_{h=1}^{q} t_h \left(d_{i_h}^k Q_{*i_h} + d_{j_h}^k Q_{*j_h}\right)$.
Set $k = k + 1$.
Select
$$(i_1, j_1) \in I_{up}^{\epsilon,MVP}(\mathbf{x}^k) \times I_{low}^{\epsilon,MVP}(\mathbf{x}^k).$$

End While
Return $\mathbf{x}^k$.

Let us analyse the PARSMO scheme into details. The starting feasible point $\mathbf{x}^0$ is set, as usual in SVM training, equal to the null vector. In this case, the gradient ∇f^0 is readily obtained as $\nabla f^0 = -\mathbf{e}$. PARSMO algorithm selects at each iteration the ϵ-MVP $(i_1, j_1) \in I_{up}^{\epsilon,MVP}(\mathbf{x}^k) \times I_{low}^{\epsilon,MVP}(\mathbf{x}^k)$, using the $\epsilon-$perturbations of the sets I_{up} and I_{low} in (11), and further $(q-1)$ pairs, that all together made up the set of blocks $\mathcal{J}^k$.

The search direction $\mathbf{d}^k$ at **S.3** is the sum of all the SMO steps related to the pairs in $\mathcal{J}^k$ obtained by analytically computing by (10) the optimal stepsise t_h, along the directions $\mathbf{d}^{(i_h j_h)}$ for $h = 1, \ldots, q$ corresponding to the q subproblems of type (13). It has $\|\mathbf{d}^k\|_0 = 2q$ with $q \geq 1$ which depends on the number of parallel/distibuted processes that we want to activate.

Finally at **S.4** the steplenght α^k that exactly minimizes the objective function along $\mathbf{d}^k$ is obtained by (29); note that this step requires no further kernel evaluations. The same holds for the gradient updated by the rule (19).

In PARSMO it remains to specify how to select the $q-1$ pairs forming $\mathcal{J}^k$. We propose two different choices that allow to use different caching strategies. Indeed, the most expensive computational burden is due to kernel columns computation, and caching is a well known strategy to save computations in SVMs decompostion algorithms. The two Matlab implementations of PARSMO use the cache in two opposite ways.

- PARSMO^{light}: the $q-1$ pairs are selected by choosing those pairs that ϵ-most violate the first order optimality condition, similar to the SVM$^{\text{light}}$ algorithm [15];
- PARSMO^{cache}: the $q-1$ pairs are selected following the same SVM^{light} rule, but restricted to the index sets $\mathcal{C} \cap I^{\epsilon}_{up}(\mathbf{x}^k) \times \mathcal{C} \cap I^{\epsilon}_{low}(\mathbf{x}^k)$, where $\mathcal{C}$ is the index set of the kernel columns available in the cache.

To be more precise, in PARSMO^{light} we select the q pairs $(i_h, j_h) \in I^{\epsilon}_{up}(\mathbf{x}^k) \times I^{\epsilon}_{low}(\mathbf{x}^k)$ sequentially so that

$$-y_{i_1}\nabla f(\mathbf{x}^k)_{i_1} \geq -y_{i_2}\nabla f(\mathbf{x}^k)_{i_2} \geq \cdots \geq -y_{i_q}\nabla f(\mathbf{x}^k)_{i_q},$$

and

$$-y_{j_1}\nabla f(\alpha^k)_{j_1} \leq -y_{j_2}\nabla f(\mathbf{x}^k)_{j_2} \leq \cdots \leq -y_{j_q}\nabla f(\mathbf{x}^k)_{j_q}.$$

In this case, although we can use a standard caching strategy, we cannot control the number of kernel columns evaluations at each iteration that in the worst case can be up to $2q$. The computation of the kernel columns Q_{*i_h} and $Q_{*j_h}, \forall h \in \{1, \ldots, q\}$, can be performed in parallel by the processors empowered to solve the subproblems. In this case the number of kernel evaluations per iteration would be of course greater than those of a standard MVP, but the overall number of iterations may decrease. Thus we keep the advantages of performing simple analytic optimizations, as in SMO methods, whilst moving $2q$ components at a time, as in SVM^{light}. We note that reconstruction of the overall gradient ∇f^{k+1} can be parallelized among the q processors and requires a synchronization step to take into account the stepsize α^k. Thus the CPU-time needed is essentially equivalent to a gradient update of a single SMO step. In this approach the transmission time among the processors may be quite significant and this strictly depends on the parallel architecture.

In algorithm PARSMO^{cache}, in addition to an ϵ-MVP $(i_1, j_1) \in I^{\epsilon,MVP}_{up}(\mathbf{x}^k) \times I^{\epsilon,MVP}_{low}(\mathbf{x}^k)$, the $q-1$ pairs are selected exclusively among the indices of the columns currently available in the cache. Being $\mathcal{C}$ the index set of the kernel columns available in the cache the $q-1$ pairs (i_h, j_h) in $\mathcal{J}^k$ are selected following the rule described above for PARSMO^{light}, but restricted to the index sets $\mathcal{C} \cap I^{\epsilon}_{up}(\mathbf{x}^k) \times \mathcal{C} \cap I^{\epsilon}_{low}(\mathbf{x}^k)$. In this case the number of kernel evaluations per iteration is at most two as in a standard MVP implementation. The rationale of this version is to improve the performances of a classical MVP algorithm by using simultaneous multiple SMO optimizations without increasing the amount of kernel evaluations.

In order to have a flavour of the potentiality of these two parallelizable strategies, we performed some simple Matlab experiments for the two versions

PARSMO^{light} and PARSMO^{cache}. Both PARSMO^{light} and PARSMO^{cache} make use of a standard caching strategy, see [4]. All experiments have been carried out on a 64-bit intel-Core i7 CPU 870 2.93Ghz $\times$ 8, with a cache memory of 500 columns.

We perform experiments simulating $q = 1, 2, 4$ and 8 parallel processes. The case with $q = 1$ corresponds to a classical MVP algorithm with a standard use of caching strategy which is exactly the rule implemented in LIBSVM 2.7 [4].

We report this case to compare the performance with a standard sequential MVP implementation in order to analyze possible advantages of the PARSMO scheme. It is worth noting that to preserve the good numerical behavior of PARSMO^{light} and PARSMO^{cache}, the use of the "gathering" steplenght α^k is necessary. In fact, further tests not reported here showed that, by removing the use of α^k, oscillatory and divergence phenomena may occur when using multiple parallel processes. This enforces the practical relevance of our theoretical analysis.

We tested both PARSMO^{light} and PARSMO^{cache} on six benchmark problems available at the LIBSVM site `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`, using a standard setting for the parameters ($C = 1$, gaussian parameter $\gamma = 1/\#\text{features}$), see Table 1.

name	#features	#training data	kernel type
a9a	123	32561	gaussian
gisette scale	5000	6000	linear
cod-rna	8	59535	gaussian
real-sim	20958	72309	linear
rcv1	47236	20242	linear
w8a	300	49749	linear

Table 1 Training problems description.

To evaluate the behavior of the algorithms we report the "relative error" (RE)

$$RE = \frac{|f^* - f|}{|f^*|},$$

where f^* is the (known) optimal value of the objective function.

In particular, as regards PARSMO^{light}, we plot the RE versus

i) the number of iterations (see Figure 1);
ii) the number of kernel evaluations per process, which is obtained by dividing the total number of kernel evaluations by the number of parallel processes involved (see Figure 2). We put in the picture also the performance of the LIBSVM 3.22 which implements a second order selection rule [10] with $q = 1$.

Our results show that the larger q is, the steeper the RE decrease is. This emphasizes the positive effect of moving along multiple SMO directions at a time. In particular $q = 8$ turns out to be always the best one among all the versions including the second order LIBSVM 3.22.

As regards PARSMO^{cache}, we note that, except for the MVP pair which can require the computation of the kernel columns Q_{*i_1} and Q_{*j_1}, each SMO process computes only the analytical solution of the two-dimensional subproblem, since kernel columns are already available in the cache. Thus, PARSMO^{cache} may produce a CPU-time saving even by running the algorithm in a sequential fashion. In order to show the efficiency of its steps, in Figure 3 we plot RE versus the CPU-time consumed. In this case we do not report the comparison with the LIBSVM software because our code is in Matlab and the time comparison with a C++ implementation is not fair.

PARSMO^{cache} with $q > 1$ seems to be faster than a classical MVP algorithm. This is due to the use of multiple search directions without suffering from an increase of time consuming kernel evaluations or from the need of iterative solutions of larger quadratic subproblems. It is important to outline that PARSMO^{cache} achieves its good performances by combining a convergent parallel structure with an efficient sequential implementation, and it seems to be useful also in a single-core environment.

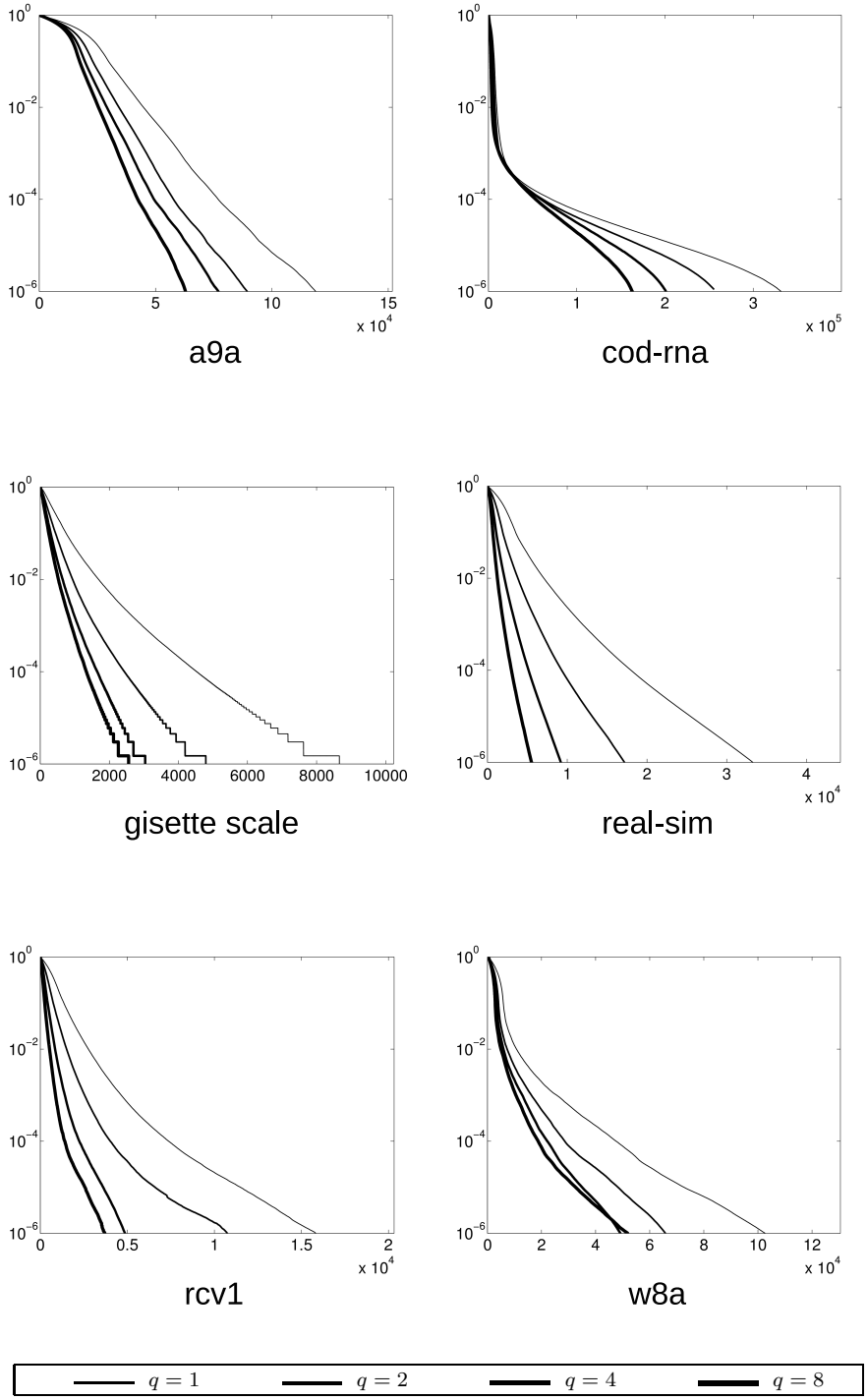

Fig. 1 PARSMO^{light}: Relative Error versus iterations.

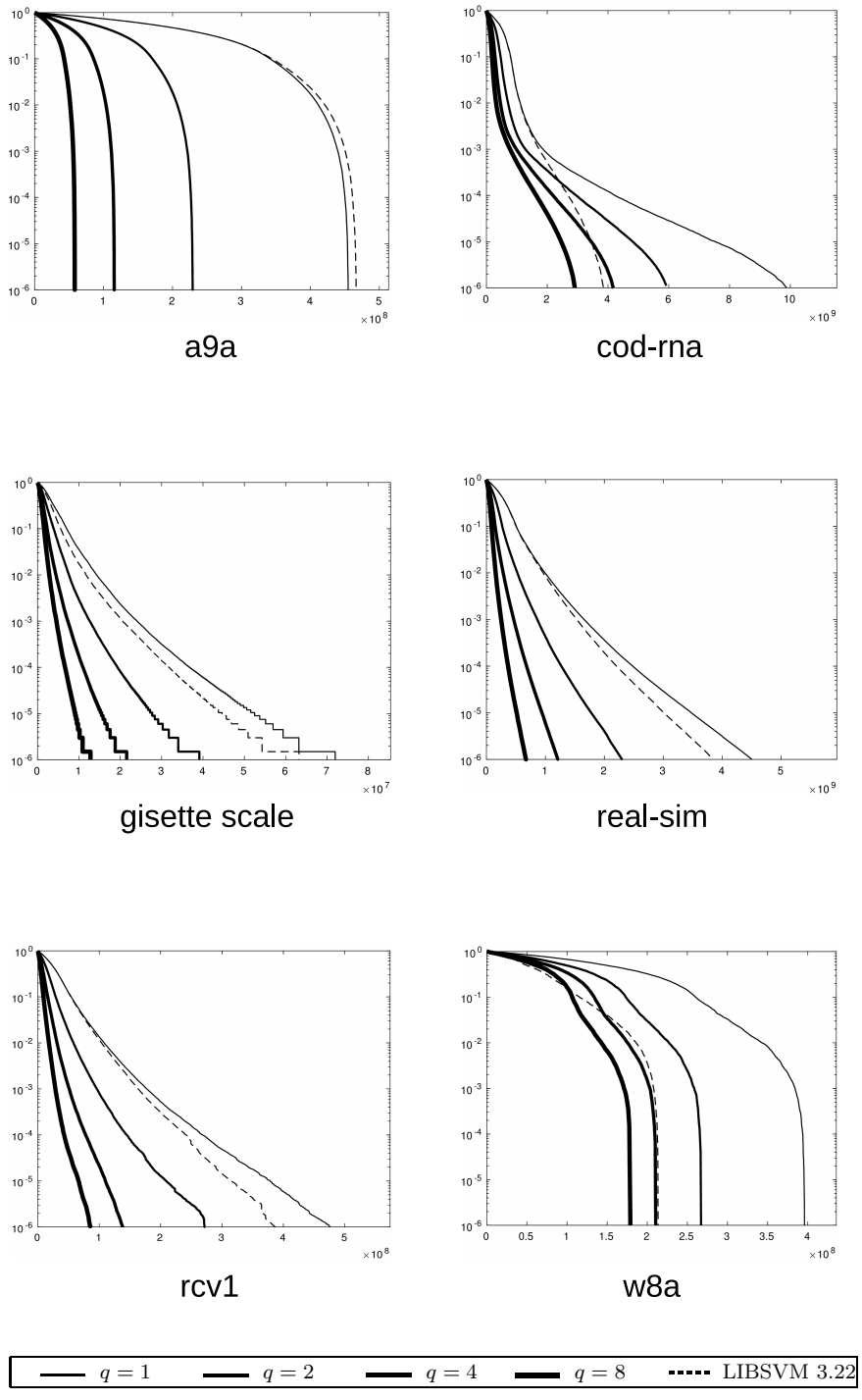

Fig. 2 PARSMO^{light}: Relative Error versus kernel evaluations per process.

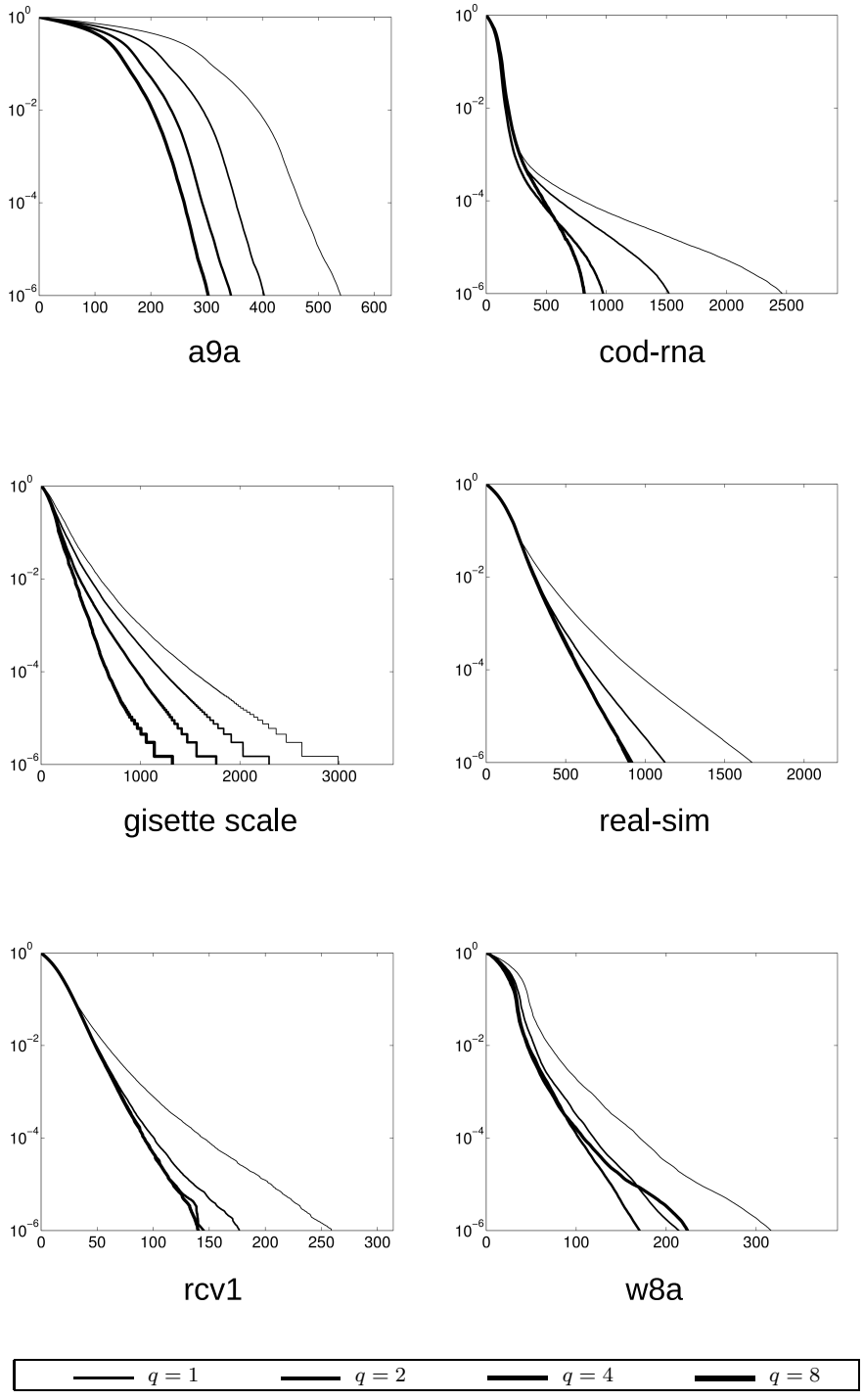

Fig. 3 PARSMO^{cache}: Relative Error versus (sequential) CPU-time.

8 The General problem

In this section we consider the general formulation (3) where f is a twice continuously differentiable convex function and the feasible set is defined as

$$\mathcal{F} = \{\mathbf{x} \in \mathbb{R}^n : A\mathbf{x} = \mathbf{b}, \quad \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}\}.$$

We want to give an hint of how the approach for convex quadratic problems with one single equality constraint can be extended to problems with a general convex objective function and multiple equality constraints. Indeed, up to authors' knowledge, the study of convergent parallel decomposition methods for these general problems with multiple constraints is still at its early stages, see e.g. [29, Section IV].

To guarantee that the solution set $\widetilde{S} = \arg\min\{f(x) \; x \in \mathcal{F}\}$ of problem (3) is nonempty, we introduce the following assumption.

Assumption 1 *The feasible set $\mathcal{F}$ is nonempty and bounded.*

Following the notation of the preceding section, given a subset $P_i \subset \{1, \ldots, n\}$, and fixing $\mathbf{x}_{-P_i}$ to the current value $\mathbf{x}^k_{-P_i}$, we define the reduced feasible set as

$$\mathcal{F}^k_{P_i} := \{\mathbf{x}_{P_i} \in \mathbb{R}^{|P_i|} : A_{P_i}\mathbf{x}_{P_i} = A_{P_i}\mathbf{x}^k_{P_i}, \; \mathbf{l}_{P_i} \leq \mathbf{x}_{P_i} \leq \mathbf{u}_{P_i}\}$$

where A_{P_i} is the $m \times |P_i|$ submatrix of A with column indices in P_i.

The problem of minimizing over $\mathbf{x}_{P_i}$ is then given by (13) with the corresponding $\mathcal{F}^k_{P_i}$.

We can consider again Algorithm 1. The following theorem generalizes Theorem 1.

Theorem 8 *Let $\mathbf{x}^k \in \mathcal{F}$. Assume that, for all $P_i \in \mathcal{J}^k$, either $\nabla_{P_i} f(\cdot, \mathbf{x}^k_{-P_i})$ is strongly monotone with modulus σ^k_i, or $\tau^k_i > 0$.*

Then the direction $\mathbf{d}^k$ defined by (14) is feasible and satisfies

$$\nabla f(\mathbf{x}^k)^T \mathbf{d}^k \leq -\rho \|\mathbf{d}^k\|^2 \tag{43}$$

with $\rho = \min_{P_i \in \mathcal{J}^k}(\tau^k_i + \sigma^k_i) > 0$.

Proof The direction $\mathbf{d}^k$ is clearly feasible at $\mathbf{x}^k$ since it belongs to the cone

$$\{\mathbf{d} \in \mathbb{R}^n : \; A\mathbf{d} = 0, \; \mathbf{d}_i \geq 0, \; \forall \, i : \; \mathbf{x}^k_i = \mathbf{l}_i, \text{ and } \mathbf{d}_i \leq 0, \; \forall \, i : \; \mathbf{x}^k_i = \mathbf{u}_i\}.$$

Note that $\widehat{\mathbf{x}}^k_{P_i}$ satisfies the optimality condition (16), for any feasible direction $\mathbf{d}_{P_i}$ at $\widehat{\mathbf{x}}^k_{P_i}$, and note that the direction $-\mathbf{d}^k_{P_i}$ is a feasible direction at $\widehat{\mathbf{x}}^k_{P_i}$ for all $P_i \in \mathcal{J}^k$. Hence (17) holds and we can write

$$\begin{aligned} &\nabla_{P_i} f(\mathbf{x}^k)^T \mathbf{d}^k_{P_i} = \\ &- \left(\nabla_{P_i} f(\widehat{\mathbf{x}}^k_{P_i}, \mathbf{x}^k_{-P_i}) - \nabla_{P_i} f(\mathbf{x}^k)\right)^T \mathbf{d}^k_{P_i} + \nabla_{P_i} f(\widehat{\mathbf{x}}^k_{P_i}, \mathbf{x}^k_{-P_i})^T \mathbf{d}^k_{P_i} \leq \\ &-(\sigma^k_i + \tau^k_i)\|\mathbf{d}^k_{P_i}\|^2. \end{aligned}$$

The thesis holds readily. □

The definition of descent block must be modified because we do not have anymore a reference point such as $\mathbf{x}^k_{\texttt{MVP}}$ to compare with. In this more general setting we use as optimality measure the distance of the feasible point $\mathbf{x}^k$ from the solution set $\widetilde{S}$ of problem (3), namely

$$\operatorname{dist}(\mathbf{x}^k, \widetilde{S}) := \min_{\mathbf{x}\in\widetilde{S}} \left\|\mathbf{x} - \mathbf{x}^k\right\|.$$

Definition 4 (Descent block) Given $\epsilon > 0$. At a feasible point $\mathbf{x}^k$, we say that the block of variables $P_i \subseteq \{1, \ldots, n\}$ is a *descent block* if it satisfies

$$\|\widehat{\mathbf{x}}^k_{P_i} - \mathbf{x}^k_{P_i}\| \geq \epsilon \operatorname{dist}(\mathbf{x}^k, \widetilde{S}), \tag{44}$$

where $\widehat{\mathbf{x}}^k_{P_i}$ is the optimal solution of the corresponding problem (13) and $\widetilde{S}$ is the solution set of problem (3).

Accordingly, we have the definition of descent iteration as given in (3).

Similarly to what done in the previous sections, we will prove that by using a suitable α^k at **S.5** the sequence $\{\mathbf{x}^k\}$ produced by the algorithm satisfies

$$\lim_{k\to\infty} \operatorname{dist}(\mathbf{x}^k, \widetilde{S}) = 0. \tag{45}$$

The proof of the following theorem can be directly obtained by those of Theorems 2, 3, and 4, and then it is omitted.

Theorem 9 *Let $\{\mathbf{x}^k\}$ be the sequence generated by Algorithm 1, and assume that conditions of Theorem 8 are satisfied. Let $\alpha^k \in (0, 1]$ at* **S.5***, and assume that one of the following conditions holds:*

(i) *either the Armijo condition* (22) *or the monotone diminishing rule* (30) *with $\alpha^k \geq \alpha^{k+1}$ $\forall\, k$ holds, and for all k there exists an integer $\tilde{k}$, with $k \leq \tilde{k} \leq L + k$ for a finite $L \geq 0$, such that $\tilde{k}$ is a descent iteration;*

(ii) *the diminishing rule* (30) *holds, and k is a descent iteration for all k.*

Then either Algorithm 1 terminates in a finite number of iterations to a solution $\bar{\mathbf{x}} \in \widetilde{S}$ or $\{\mathbf{x}^k\}$ admits a limit point and it satisfies (45)*, and thus it is in $\widetilde{S}$.*

Now the open problems are two: (i) how to check if a block is descent, and (ii) how to generate a descent block.

We give a simple sufficient condition to check if P_i is a descent block. First of all, by exploiting the boundedness of $\mathcal{F}$, we can define the maximum distance in $\mathcal{F}$: $\varphi := \max_{\mathbf{x},\mathbf{y}\in\mathcal{F}} \|\mathbf{x} - \mathbf{y}\| \leq \sqrt{n}\, \|\mathbf{u} - \mathbf{l}\|_\infty$, which is a bounded measure. Then, for a given $\epsilon > 0$ sufficiently small (e.g. $\epsilon << \frac{1}{\varphi}$), and since $\varphi \geq \operatorname{dist}(\mathbf{x}^k, \widetilde{S})$, the sufficient condition is

$$\|\widehat{\mathbf{x}}^k_{P_i} - \mathbf{x}^k_{P_i}\| \geq \epsilon\varphi.$$

On the other hand, to define a simple procedure that generates descent blocks one can certainly exploit the specific structure of the problem instance. However, in general, the pseudo randomic selection of all possible partitions is a simple effective strategy well suited for huge data applications.

Acknowledgments

The authors thank Prof. Marco Sciandrone (Dipartimento di Ingegneria dell'Informazione, Università di Firenze) for fruitful discussions and suggestions that improved significantly the paper.

References

1. Bertsekas, D.P.: Nonlinear Programming. Athena Scientific (1995)
2. Bottou, L., Curtis, F.E., Nocedal, J.: Optimization methods for large-scale machine learning. arXiv preprint arXiv:1606.04838 (2016)
3. Cao, L.J., Keerthi, S.S., Ong, C.J., Zhang, J.Q., Lee, H.P.: Parallel sequential minimal optimization for the training of support vector machines. IEEE T Neural Networ **17**(4), 1039–1049 (2006)
4. Chang, C.C., Lin, C.J.: LIBSVM: A library for support vector machines. ACM Trans Intell Syst Technol **2**, 27:1–27:27 (2011). Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm
5. Chen, P.H., Fan, R.E., Lin, C.J.: A study on SMO-type decomposition methods for Support Vector Machines. IEEE T Neural Networ **17**(4), 893–908 (2006)
6. Correa, J.R., Schulz, A.S., Stier-Moses, N.E.: Selfish routing in capacitated networks. Math Oper Res **29**(4), 961–976 (2004)
7. Defazio, A., Bach, F., Lacoste-Julien, S.: SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In: Adv Neur In 2, vol. 2, pp. 1646–1654 (2014)
8. Facchinei, F., Scutari, G., Sagratella, S.: Parallel selective algorithms for nonconvex big data optimization. IEEE T Signal Proces **63**(7), 1874–1889 (2015). DOI 10.1109/TSP.2015.2399858
9. Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J.: LIBLINEAR: A library for large linear classification. J Mach Learn Res **9**, 1871–1874 (2008)
10. Fan, R.E., Chen, P.H., Lin, C.J.: Working set selection using second order information for training support vector machines. J Mach Learn Res **6**, 1889–1918 (2005)
11. Glasmachers, T., Igel, C.: Maximum-gain working set selection for SVMs. J Mach Learn Res **7**, 1437–1466 (2006)
12. Glasmachers, T., Igel, C.: Second-order SMO improves SVM online and active learning. Neural Comput **20**(2), 374–382 (2008)
13. Gonzalez-Lima, M.D., Hager, W.W., Zhang, H.: An affine-scaling interior-point method for continuous knapsack constraints with application to support vector machines. SIAM Journal on Optimization **21**(1), 361–390 (2011)
14. Graf, H.P., Cosatto, E., Bottou, L., Dourdanovic, I., Vapnik, V.: Parallel support vector machines: The cascade SVM. In: L.K. Saul, Y. Weiss, L. Bottou (eds.) Adv Neur In 17, pp. 521–528. MIT Press, Cambridge, MA (2004). URL `http://books.nips.cc/papers/files/nips17/NIPS2004_0190.pdf`
15. Joachims, T.: Making large scale SVM learning practical. In: C. Schölkopf, C. Burges, A. Smola (eds.) Advances in Kernel Methods - Support Vector Learning. MIT Press, Cambridge, MA (1998)
16. Kamesam, P.V., Meyer, R.R.: Multipoint methods for separable nonlinear networks. In: Mathematical Programming at Oberwolfach II, pp. 185–205. Springer Berlin Heidelberg (1984)
17. Kao, C., Lee, L.F., Pitt, M.M.: Simulated maximum likelihood estimation of the linear expenditure system with binding non-negativity constraints. Ann Econ Financ **2**(1), 203–223 (2001)
18. Keerthi, S.S., Gilbert, E.G.: Convergence of a generalized SMO algorithm for SVM classifier design. Mach Learn **46**(1-3), 351–360 (2002)
19. Lin, C., Lucidi, S., Palagi, L., Risi, A., Sciandrone, M.: Decomposition algorithm model for singly linearly-constrained problems subject to lower and upper bounds. J Optimiz Theory App **141**(1), 107–126 (2009)

20. Lin, C.J.: Asymptotic convergence of an SMO algorithm without any assumptions. IEEE T Neural Networ **13**, 248–250 (2002)
21. Lin, C.J.: A formal analysis of stopping criteria of decomposition methods for support vector machines. IEEE T Neural Networ **13**(5), 1045–1052 (2002)
22. Liu, J., Wright, S., R, C., Bittorf, V., Sridhar, S.: An asynchronous parallel stochastic coordinate descent algorithm. J Mach Learn Res **16**, 285–322 (2015)
23. Liuzzi, G., Palagi, L., Piacentini, M.: On the convergence of a Jacobi-type algorithm for singly linearly-constrained problems subject to simple bounds. Optim Lett **5**(2), 347–362 (2011)
24. Lucidi, S., Palagi, L., Risi, A., Sciandrone, M.: A convergent decomposition algorithm for support vector machines. Comput Optim Appl **38**(2), 217–234 (2007)
25. Lucidi, S., Palagi, L., Risi, A., Sciandrone, M.: A convergent hybrid decomposition algorithm model for SVM training. IEEE T Neural Networ **20**(6), 1055–1060 (2009)
26. Manno, A., Sagratella, S., Livi, L.: A convergent and fully distributable SVMs training algorithm. In: Neural Networks (IJCNN), 2016 International Joint Conference on, pp. 3076–3080. IEEE (2016)
27. Palagi, L., Sciandrone, M.: On the convergence of a modified version of SVM^{light} algorithm. Optim Method Softw **20**(2-3), 311–328 (2005)
28. Risi, A.: Convergent decomposition methods for support vector machines. Ph.D. thesis, Sapienza University of Rome (2008)
29. Scutari, G., Facchinei, F., Lampariello, L., Sardellitti, S., Song, P.: Parallel and distributed methods for constrained nonconvex optimization-part ii: Applications in communications and machine learning. IEEE Transactions on Signal Processing **65**(8), 1945–1960 (2017)
30. Shalev-Shwartz, S., Singer, Y., Srebro, N., Cotter, A.: Pegasos: Primal estimated sub-gradient solver for SVM. Math Program **127**(1), 3–30 (2011). DOI 10.1007/s10107-010-0420-4
31. Shalev-Shwartz, S., Zhang, T.: Stochastic dual coordinate ascent methods for regularized loss minimization. J Mach Learn Res **14**(1), 567–599 (2013)
32. Shalev-Shwartz, S., Zhang, T.: Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. Math Program **155**(1-2), 105–145 (2016)
33. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press, New York, NY, USA (2004)
34. Steinwart, I., Hush, D., Scovel, C.: Training SVMs without offset. J Mach Learn Res **12**, 141–202 (2011)
35. Takác, M., Bijral, A., Richtárik, P., Srebro, N.: Mini-batch primal and dual methods for SVMs. In: In 30th International Conference on Machine Learning, pp. 537–552. Springer (2013)
36. Tseng, P., Yun, S.: A coordinate gradient descent method for linearly constrained smooth optimization and support vector machines training. Comput Optim Appl **47**(2), 179–206 (2010)
37. Yang, J.: An improved cascade SVM training algorithm with crossed feedbacks. In: Computer and Computational Sciences, 2006. IMSCCS '06. First International Multi-Symposiums on, vol. 2, pp. 735–738 (2006)
38. Zanghirati, G., Zanni, L.: A parallel solver for large quadratic programs in training support vector machines. Parallel Comput **29**(4), 535 – 551 (2003)
39. Zanni, L., Serafini, T., Zanghirati, G.: Parallel software for training large scale support vector machines on multiprocessor systems. J Mach Learn Res **7**, 1467–1492 (2006)
40. Zhang, J.P., Li, Z.W., Yang, J.: A parallel SVM training algorithm on large-scale classification problems. In: Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on, vol. 3, pp. 1637–1641 Vol. 3 (2005)