

## Parallel Decomposition Approaches for Training Support Vector Machines\*

Thomas Serafini<sup>a</sup>, Gaetano Zanghirati<sup>b</sup>, Luca Zanni<sup>a</sup>

<sup>a</sup>Department of Mathematics, University of Modena and Reggio-Emilia, via Campi 213/b, 41100 Modena, Italy. E-mail: [serafini.thomas@unimo.it](mailto:serafini.thomas@unimo.it), [zanni.luca@unimo.it](mailto:zanni.luca@unimo.it).

<sup>b</sup>Department of Mathematics, University of Ferrara, via Machiavelli 35, 44100 Ferrara, Italy. E-mail: [g.zanghirati@unife.it](mailto:g.zanghirati@unife.it).

We consider parallel decomposition techniques for solving the large quadratic programming (QP) problems arising in training support vector machines. A recent technique is improved by introducing an efficient solver for the inner QP subproblems and a preprocessing step useful to hot start the decomposition strategy. The effectiveness of the proposed improvements is evaluated by solving large-scale benchmark problems on different parallel architectures.

### 1. Introduction

Support Vector Machines (SVMs) are an effective learning technique [11] which received increasing attention in the last years. 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 learning methodology 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

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

where  $K : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}$  denotes a special kernel function (linear, polynomial, Gaussian, ...) and  $\mathbf{x}^* = (x_1^*, \dots, x_n^*)^T$  is the solution of the convex quadratic programming (QP) problem

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

where  $G$  has entries  $G_{ij} = y_i y_j K(\mathbf{z}_i, \mathbf{z}_j)$ ,  $i, j = 1, 2, \dots, n$ ,  $\mathbf{1} = (1, \dots, 1)^T$  and  $C$  is a parameter of the SVM algorithm. Once the vector  $\mathbf{x}^*$  is computed,  $b^* \in \mathbb{R}$  in (1) is easily derived. An example  $\mathbf{z}_i$  is called *support vector* (SV) if  $x_i^* \neq 0$  and *bound support vector* (BSV) if  $x_i^* = C$ . The matrix  $G$  is generally dense and in many real-world applications its size is very large ( $n \gg 10^4$ ). Thus, strategies suited to exploit the special problem features become a need, since standard QP solvers based on explicit storage of  $G$  cannot be used. Among these strategies, decomposition techniques are certainly the most investigated [3–7]. They consist in splitting the original problem into a sequence of QP subproblems sized  $n_{sp} \ll n$  that can fit into the available memory. An effective parallel decomposition technique is proposed in [12]. It is based on the Joachims' decomposition idea [5] and on a special variable projection method [8,9] as QP solver for the inner subproblems. The main steps of this decomposition approach are briefly recalled in Algorithm DT.

In contrast with other decomposition algorithms, that are tailored for very small-size subproblems (typically less than  $10^2$ ), the proposed strategy is appropriately designed to be effective with subproblems large enough (typically more than  $10^3$ ) to produce few decomposition steps. A wide numerical

---

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

---

ALGORITHM DT (SVM Decomposition Technique)

---

1. Let  $\mathbf{x}^{(0)}$  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  $i = 0$ . Arbitrarily split the indices  $\{1, \dots, n\}$  into the set  $B$  of *basic* variables, with  $\#B = n_{sp}$ , and the set  $N = \{1, \dots, n\} \setminus B$  of *nonbasic* variables. Arrange the arrays  $\mathbf{x}^{(i)}$ ,  $\mathbf{y}$  and  $G$  with respect to  $B$  and  $N$ :

$$\mathbf{x}^{(i)} = \begin{bmatrix} \mathbf{x}_B^{(i)} \\ \mathbf{x}_N^{(i)} \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} \mathbf{y}_B \\ \mathbf{y}_N \end{bmatrix}, \quad G = \begin{bmatrix} G_{BB} & G_{BN} \\ G_{NB} & G_{NN} \end{bmatrix}.$$

2. Compute (in parallel) the solution  $\mathbf{x}_B^{(i+1)}$  of

$$\min_{\mathbf{x}_B \in \Omega_B} f_B(\mathbf{x}_B) = \frac{1}{2} \mathbf{x}_B^T G_{BB} \mathbf{x}_B - \mathbf{x}_B^T (\mathbf{1} - G_{BN} \mathbf{x}_N^{(i)}) \quad (3)$$

where  $\Omega_B = \{\mathbf{x}_B \in \mathbb{R}^{n_{sp}} \mid \mathbf{y}_B^T \mathbf{x}_B = -\mathbf{y}_N^T \mathbf{x}_N^{(i)}, \mathbf{0} \leq \mathbf{x}_B \leq C\mathbf{1}\}$ . Set  $\mathbf{x}^{(i+1)} = \begin{bmatrix} \mathbf{x}_B^{(i+1)T} & \mathbf{x}_N^{(i)T} \end{bmatrix}^T$ .

3. Update (in parallel) the gradient  $\nabla f(\mathbf{x}^{(i+1)}) = \nabla f(\mathbf{x}^{(i)}) + \begin{bmatrix} G_{BB} & G_{BN} \end{bmatrix}^T (\mathbf{x}_B^{(i+1)} - \mathbf{x}_B^{(i)})$  and terminate if  $\mathbf{x}^{(i+1)}$  satisfies the KKT conditions.
  4. Change at most  $n_c$  elements of  $B$ . The entering indices are determined by a strategy based on the Zoutendijk's method [5,12]. Set  $i \leftarrow i + 1$  and go to step 2.
- 

investigation has shown that this strategy is well comparable with the most effective decomposition approaches on scalar architectures. However, its straightforward parallelization is an innovative feature. In fact, the expensive tasks (kernel evaluations and QP subproblems solution) of the few decomposition steps can be efficiently performed in parallel and promising computational results are reported in [12].

In this paper two improvements to the above approach are introduced: a parallel preprocessing step, which gives a good starting point to the decomposition technique, and a new parallel inner solver that exhibits better convergence rate. The good efficiency and scalability of the improved technique is shown by solving large-scale benchmark problems on different multiprocessor systems.

## 2. A preprocessing step for a hot start

The decomposition approach in [12] uses the null vector as starting point. This is a good choice for problems which have few support vectors, that is many null components in  $\mathbf{x}^*$ . In other cases, where there are many support vectors, the convergence of the decomposition technique could be improved by hot starting the method with a better approximation of  $\mathbf{x}^*$ . Here, we introduce a preprocessing step that produces an estimation of  $\mathbf{x}^*$  by a scalable and not excessively expensive method; this estimation is used as the initial guess  $\mathbf{x}^{(0)}$  for the decomposition technique [12].

A rough approximation  $\bar{\mathbf{x}}$  of  $\mathbf{x}^*$  can be calculated by solving  $p$  independent QP subproblems with the same structure of (2), in the following way: given a partition  $\{I_j\}_{j=1}^p$  of  $\{1, \dots, n\}$ , compute  $\forall j$  the solution  $\bar{\mathbf{x}}_{I_j}$  of the QP subproblem

$$\begin{aligned} \text{subproblem } P_j : \quad & \min \quad \frac{1}{2} \mathbf{x}_{I_j}^T G_{I_j I_j} \mathbf{x}_{I_j} - \mathbf{x}_{I_j}^T \mathbf{1} \\ & \text{sub. to} \quad \mathbf{y}_{I_j}^T \mathbf{x}_{I_j} = 0, \quad \mathbf{0} \leq \mathbf{x}_{I_j} \leq C\mathbf{1} \end{aligned} \quad (4)$$

where  $G_{I_j I_j} = \{G_{ik} \mid i, k \in I_j\}$ . The approximation  $\bar{\mathbf{x}}$  is obtained by rearranging the solutions of the subproblems:  $\bar{\mathbf{x}}^T = (\bar{\mathbf{x}}_{I_1}^T, \dots, \bar{\mathbf{x}}_{I_p}^T)$ . Note that problems  $P_1, \dots, P_p$  are independent and, on a multiprocessor system, they can be distributed among the available processing elements and managed at the same time. The described procedure is equivalent to partition the training set into  $p$  disjoint subsets  $D_j = \{(z_i, y_i) \in D \mid i \in I_j\}$  and in training an SVM on each  $D_j$ .

Table 1

Preprocessing with different subproblem size on UCI Adult data set ( $n = 11220$ ,  $SV = 4222$ )

$p$	$n_p$	$SV_{\text{est}}$	% corr.	time	$p$	$n_p$	$SV_{\text{est}}$	% corr.	time
2	5610	4328	90.7	40.70	33	340	5620	68.7	16.23
4	2805	4441	85.5	22.72	81	139	6415	60.9	17.80
9	1247	4780	79.6	16.82	197	57	7561	52.3	20.53
21	535	5271	73.2	15.88	387	29	8674	46.3	23.30

After the preprocessing step the estimation  $\bar{\mathbf{x}}$  is taken as initial point of the parallel decomposition technique:  $\mathbf{x}^{(0)} = \bar{\mathbf{x}}$ . The indices for the initial working set  $B$  are randomly chosen among the indices  $j$  such that  $\bar{x}_j \neq 0$  (indices of potential support vectors) and, if these are not enough, the set  $B$  is filled up to  $n_{sp}$  entries with random indices  $j$  such that  $\bar{x}_j = 0$ . In comparison with the strategy in [12], which uses  $\mathbf{x}^{(0)} = \mathbf{0}$ , this approach spends an extra time to compute:

$$G_{BN}\mathbf{x}_N^{(0)}, \quad \nabla f(\mathbf{x}^{(0)}) = G\mathbf{x}^{(0)} - \mathbf{1}. \quad (5)$$

In fact, since  $G$  is not into memory, the computations (5) require expensive kernel evaluations when many components of  $\mathbf{x}^{(0)}$  are nonzero. Numerical results in section 4 will show that this extra time is usually compensated by a decomposition iterations reduction, so that overall time is saved.

Another important issue is how to choose the partition  $I_1, \dots, I_p$ . Small values of  $p$  lead to more expensive QP subproblems  $P_j$  than for large  $p$ , but give a better approximation of  $\mathbf{x}^*$  which yields less decomposition iterations. Besides, small values of  $p$  generally imply more null components in  $\bar{\mathbf{x}}$ , so the computations (5) are faster.

Table 1 shows the numerical behavior of the preprocessing on the UCI Adult data set (available at [www.ics.uci.edu/~mllearn](http://www.ics.uci.edu/~mllearn)) when  $K(\mathbf{z}_i, \mathbf{z}_j) = \exp(-\|\mathbf{z}_i - \mathbf{z}_j\|_2^2 / (2\sigma^2))$ ,  $\sigma^2 = 10$  and  $C = 1$  are used. The experiments are carried out on a Power4 1.3GHz processor of an IBM SP4. The subproblems  $P_j$ ,  $j = 1, \dots, p-1$ , have size  $n_p$  while  $P_p$  is sized  $n - (p-1)n_p$ . All the subproblems are solved by the gradient projection method described in section 3. We stop the solver when the KKT conditions for problem  $P_j$  are satisfied within a tolerance of 0.01 (see section 4 for a motivation of this choice). In the table we report the number  $SV_{\text{est}}$  of nonzero components in  $\bar{\mathbf{x}}$  (an estimation of the final number of support vectors) and the percentage of correctly estimated SVs (% corr.). The time column reports the seconds needed for both the preprocessing and the computations (5). It can be observed that large subproblems give more accurate estimation of the final set of support vectors. Furthermore, for decreasing  $n_p$  the time reduces until computations (5) become predominant due to the increased number of  $SV_{\text{est}}$ . This suggests that a sufficiently large  $n_p$  is preferable both in terms of SV estimation and computational time.

### 3. The Generalized Variable Projection Method for the inner subproblems

In this section we discuss the solution of the subproblem (3) by the Generalized Variable Projection Method (GVPM), recently introduced in [10]. This iterative scheme is a gradient projection method that uses a limited minimization rule as linesearch technique [2] and a steplength selection based on an adaptive alternation between the two Barzilai-Borwein rules [1]. When it is applied to (3), it can be stated as in Algorithm GVPM-SVM.

It is interesting to observe that the special variable projection method used in [12] for solving (3) can be considered a special case of GVPM-SVM, obtained by setting  $i_\alpha = 2$  and  $n_{\min} = n_{\max} = 3$ ; it will be denoted by  $AL_3$ -VPM in the sequel. In comparison with  $AL_3$ -VPM, GVPM-SVM presents the same low computational cost per iteration ( $O(n_{sp}^2)$ ), but it benefits from a more effective selection rule for the steplength  $\alpha_{k+1}$ , that generally yields better convergence rate. We refer to [10] for a deeper analysis of the GVPM behavior on more general QP problems and for comparisons with other gradient projection methods.

Here, some numerical evidences of the GVPM-SVM improvements with respect to the  $AL_3$ -VPM are given by comparing their behavior on the subproblems arising when the decomposition techniques [12] is applied to train a Gaussian classifier on the MNIST database of handwritten digits ([yann.lecun.com/exdb/mnist](http://yann.lecun.com/exdb/mnist)). A classifier for the digit "8" is trained with SVM parameters  $C = 10$

---

ALGORITHM GVPM-SVM (Generalized Variable Projection Method for subproblem (3))

---

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

2. Terminate if  $\mathbf{w}^{(k)}$  satisfies a stopping criterion; otherwise compute\*

$$\mathbf{d}^{(k)} = P_{\Omega_B}(\mathbf{w}^{(k)} - \alpha_k(G_{BB}\mathbf{w}^{(k)} + (G_{BN}\mathbf{x}_N^{(i)} - \mathbf{1}))) - \mathbf{w}^{(k)}.$$

3. Compute  $\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \lambda_k \mathbf{d}^{(k)}$  with  $\lambda_k = \arg \min_{\lambda \in [0,1]} f_B(\mathbf{w}^{(k)} + \lambda \mathbf{d}^{(k)})$

4. If  $\mathbf{d}^{(k)T} G_{BB} \mathbf{d}^{(k)} = 0$  then set  $\alpha_{k+1} = \alpha_{\max}$ ; else  
compute  $\alpha_{k+1}^{(1)} = \frac{\mathbf{d}^{(k)T} \mathbf{d}^{(k)}}{\mathbf{d}^{(k)T} G_{BB} \mathbf{d}^{(k)}}$ ,  $\alpha_{k+1}^{(2)} = \frac{\mathbf{d}^{(k)T} G_{BB} \mathbf{d}^{(k)}}{\mathbf{d}^{(k)T} G_{BB}^2 \mathbf{d}^{(k)}}$ ,  $\lambda_{\text{opt}} = \arg \min_{\lambda} f_B(\mathbf{w}^{(k)} + \lambda \mathbf{d}^{(k)})$

If  $(n_\alpha \geq n_{\min})$  and  $\left( (n_\alpha \geq n_{\max}) \text{ or } (\alpha_{k+1}^{(2)} \leq \alpha_k \leq \alpha_{k+1}^{(1)}) \right)$   
or  $\left( (\lambda_{\text{opt}} < \lambda_\ell \text{ and } \alpha_k = \alpha_k^{(1)}) \text{ or } (\lambda_{\text{opt}} > \lambda_u \text{ and } \alpha_k = \alpha_k^{(2)}) \right)$  then

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

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  $P_{\Omega_B}(\cdot)$  denotes the orthogonal projection onto  $\Omega_B$ .

---

and  $\sigma = 1800$ ; the corresponding QP problem of size 60000 is solved by decomposing in subproblems sized 3700. Both solvers use the projection onto  $\Omega_B$  of the null vector as starting point and a stopping rule based on the fulfillment of the KKT conditions within a tolerance of 0.001. In the GVPM-SVM, the parameters  $i_\alpha = 2$ ,  $\alpha_{\min} = 10^{-30}$ ,  $\alpha_{\max} = 10^{30}$ ,  $n_{\min} = 3$ ,  $n_{\max} = 10$ ,  $\lambda_\ell = 0.1$  and  $\lambda_u = 5$  are used. These experiments are carried out on the platform described in section 2 and the results are summarized in Table 2; we report the number of iterations and the computational time in seconds required by the two solvers for solving the subproblems  $T_j, j = 1, \dots, 6$ , generated by the decomposition technique. From these numerical results it is evident the advantage in terms of computational time implied by the better GVPM-SVM convergence rate. Finally we remark that, since the main computational task of each iteration is the matrix-vector product  $G_{BB} \mathbf{d}^{(k)}$ , a parallel version of the GVPM-SVM can be easily obtained by a blockwise distribution of the  $G_{BB}$  rows among the available processors.

#### 4. Numerical experiments on parallel architectures

In this section we show how the parallel *Variable Projection Decomposition Technique* (VPDT) [12] can be improved by using the preprocessing step and the GVPM-SVM inner QP solver.

The new version of the decomposition scheme is called *Generalized VPDT* (GVPDT) and is coded in standard C with MPI communication routines. In order to balance the workload, the preprocessing step is implemented by dividing the training set among the available processors and then by requiring each processor to solve the same number  $p_\ell$  of problems  $P_j$  by the GVPM-SVM. We determine  $p_\ell$  in such a way that the problems  $P_j$  have similar size, as close as possible to  $1.5n_{sp}$ . For the parallel solution of (3) at each decomposition step, we use a parallel GVPM-SVM version obtained as suggested in the previous section.

We compare VPDT and GVPDT on two different multiprocessor systems of the CINECA Supercomputing Center (Bologna, Italy): an IBM SP4 equipped with 16 nodes of 32 IBM Power4 1.3GHz

Table 2

Test problems from MNIST database ( $n_{sp} = 3700$ ).

			AL <sub>3</sub> -VPM		GVPM-SVM					AL <sub>3</sub> -VPM		GVPM-SVM	
test	SV	BSV	it.	time	it.	time	test	SV	BSV	it.	time	it.	time
$T_1$	545	2	347	3.1	377	3.1	$T_4$	2694	32	978	25.5	769	20.4
$T_2$	1727	75	966	17.6	793	14.0	$T_5$	2963	5	1444	42.6	909	26.4
$T_3$	2262	95	875	19.4	764	17.3	$T_6$	2993	0	1290	37.9	1102	32.6

CPUs each and 64GB virtually shared RAM per node, and an IBM Linux Cluster with 128 Intel Pentium III processors at 1.13GHz, coupled in 64 two-CPU nodes with 1GB shared RAM per node. The largest test problems arising in training Gaussian SVMs on the MNIST database (digit “8” classifier,  $C = 10$ ,  $\sigma = 1800$ ) and the UCI Adult data set ( $C = 1$ ,  $\sigma^2 = 10$ ) are solved. The two decomposition techniques are run with the same parameter setting:  $n_{sp} = 3700$ ,  $n_c = 1500$  for the MNIST set and  $n_{sp} = 1300$ ,  $n_c = 650$  for the UCI Adult set. Furthermore, to avoid expensive recomputations of  $G$  entries, caching areas of 500Mb and 300Mb per processor are used for the MNIST set and the UCI Adult set, respectively. The decomposition procedures stop when the KKT conditions are satisfied within a tolerance of 0.001. As a consequence, in the GVPDT preprocessing step, the KKT conditions for each problem  $P_j$  are satisfied within 0.01 since, generally, higher accuracies increase the preprocessing computational cost without improving the convergence of the decomposition technique.

The results obtained by using different numbers of processors (PEs) are reported in Table 3. The columns  $sp_r$  and  $eff_r$  show the relative speedup ( $sp_r(q) = t_1/t_q$ , where  $t_j$  is the time needed by the program on  $j$  PEs) and the relative efficiency ( $eff_r(q) = sp_r(q)/q$ ), respectively. In order to allow a comparison with a widely used (serial) decomposition technique, the number of iterations and the time required by the *SVM<sup>light</sup>* package [5] are also reported. We use *SVM<sup>light</sup>* with pr\_LOQO as inner QP solver, caching areas sized as above, empirical optimal values of  $n_{sp}$  and  $n_c$  ( $n_{sp} = 8$ ,  $n_c = 4$  for MNIST and  $n_{sp} = 20$ ,  $n_c = 10$  for UCI Adult) and default setting for the other parameters. In case of MNIST test problem, the GVPDT results are obtained without preprocessing since, due to the small percentage of support vectors, the starting point  $\mathbf{x}^{(0)} = \mathbf{0}$  already yields a very good convergence rate without the extra time typically required by a preprocessing step. This means that the GVPDT performance improvement is due to the effectiveness of GVPM-SVM only. Furthermore, we observe that this improvement is obtained by preserving the very good scalability of the VPDT (the superlinear speedup is due also to the larger caching area available when more processors are used). In case of UCI Adult test problem, where the GVPDT uses the preprocessing step, a remarkable iteration count reduction is obtained. This feature does not only contribute to reduce the computational time, but also implies significant improvements to the scalability of the decomposition technique. As a final remark, note that when the number of PEs increases the workload of each processor to solve the inner subproblems decreases, hence the communication time becomes more and more comparable with the computational time. This explains the suboptimal speedup and efficiency shown for 16 PEs. Of course, for larger problems full efficiency can be recovered.

## 5. Conclusions

We presented new developments of the parallel decomposition technique for SVMs training recently introduced in [12]. The improvements concerned the computation of a good starting point for the iterative decomposition technique and the introduction of an efficient parallel solver for the QP subproblem of each iteration. A good starting point is obtained by a preprocessing step in which independent SVMs are trained on disjoint subsets of the original training set. This step is very well suited for parallel implementations and yields significant benefits in the case where the decomposition technique of [12] exhibits unsatisfactory convergence rate.

For a parallel solution of the QP subproblem at each decomposition iteration, we proposed a generalized variable projection method based on an adaptive steplength selection. Since this solver has the same cost per iteration and a better convergence rate than the variable projection method used in [12], a remarkable time reduction is observed in the subproblems solution.

Table 3

VPDT and GVPDT on MNIST ( $n = 60000$ ) and UCI Adult ( $n = 32562$ ) test problems.

PEs	MNIST (SV = 3155, BSV = 160)				UCI Adult (SV = 11698, BSV = 10605)											
	VPDT		GVPDT		VPDT		GVPDT									
	it.	time	$sp_r$	$eff_r$	it.	time	$sp_r$	$eff_r$	it.	time	$sp_r$	$eff_r$	it.	time	$sp_r$	$eff_r$
IBM SP4																
1	6	825.4			6	782.4			45	255.3			17	224.2		
2	6	367.2	2.2	1.1	6	345.2	2.3	1.1	46	169.9	1.5	0.8	18	131.0	1.7	0.9
4	6	180.1	4.6	1.1	6	167.7	4.7	1.2	48	116.4	2.2	0.5	18	80.9	2.8	0.7
8	6	118.9	6.9	0.9	6	114.2	6.9	0.9	47	60.4	4.2	0.5	18	42.2	5.3	0.7
16	6	82.2	10.0	0.6	6	75.9	10.3	0.6	46	42.3	6.0	0.4	19	26.2	8.6	0.5
	$SVM^{light}$ : 9226 it. 901.6 sec.				$SVM^{light}$ : 4407 it. 278.7 sec.											
IBM Linux Cluster																
1	6	1373.6			6	1229.1			45	613.9			17	525.8		
2	6	657.4	2.1	1.0	6	611.2	2.0	1.0	45	349.4	1.8	0.9	18	277.5	1.9	1.0
4	6	349.9	3.9	1.0	6	316.2	3.9	1.0	47	230.7	2.7	0.7	17	167.8	3.1	0.8
8	6	192.6	7.1	0.9	6	170.9	7.2	0.9	47	118.3	5.2	0.7	18	79.4	6.6	0.8
16	6	114.5	12.0	0.7	6	101.3	12.1	0.8	46	73.4	8.4	0.5	18	48.4	10.9	0.7
	$SVM^{light}$ : 9301 it. 1017.5 sec.				$SVM^{light}$ : 4515 it. 437.9 sec.											

The computational experiments on well known large-scale test problems showed that the new decomposition approach, based on the above improvements, outperforms the technique in [12] both in serial and parallel environments and can further achieve a better scalability.

## 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] C.C. Chang, C.J. Lin (2002), LIBSVM: a Library for Support Vector Machines, available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [4] R. Collobert, S. Benjo (2001), SVMtorch: Support Vector Machines for Large-Scale Regression Problems, *Journal of Machine Learning Research* **1**, 143–160.
- [5] T. Joachims (1998), Making Large-Scale SVM Learning Practical, *Advances in Kernel Methods*, B. Schölkopf *et al.*, eds., MIT Press, Cambridge, MA.
- [6] C.J. Lin (2001), On the Convergence of the Decomposition Method for Support Vector Machines, *IEEE Transactions on Neural Networks* **12**(6), 1288–1298.
- [7] J.C. Platt (1998), Fast Training of Support Vector Machines using Sequential Minimal Optimization, *Advances in Kernel Methods*, B. Schölkopf *et al.*, eds., MIT Press, Cambridge, MA.
- [8] V. Ruggiero, L. Zanni (2000), A Modified Projection Algorithm for Large Strictly Convex Quadratic Programs, *J. Optim. Theory Appl.* **104**(2), 281–299.
- [9] 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.
- [10] T. Serafini, G. Zanghirati, L. Zanni (2003), Gradient Projection Methods for Large Quadratic Programs and Applications in Training Support Vector Machines, Technical Report 48, Dept. of Mathematics, University of Modena and Reggio Emilia.
- [11] V.N. Vapnik (1998), *Statistical Learning Theory*, John Wiley and Sons, New York.
- [12] G. Zanghirati, L. Zanni (2003), A Parallel Solver for Large Quadratic Programs in Training Support Vector Machines, *Parallel Computing* **29**(4), 535–551.