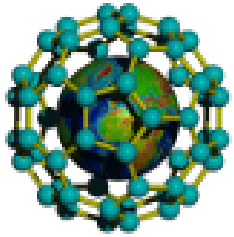


GOL

A Two-Level Decomposition Framework Exploiting First and Second Order Information for SVM Training Problems

SIMAI Congress 2020+2021, 31st August 2021

Giulio Galvan, **Matteo Lapucci**, Chih-Jen Lin, Marco Sciandrone

DINFO, Univerisity of Florence

SVM Training Problem (Dual)

$$\begin{aligned} \min_{\alpha \in \mathbb{R}^n} \; & f(\alpha) = \frac{1}{2}\alpha^T Q \alpha - \alpha^T e \\ \text{s.t.} \; & y^T \alpha = 0, \\ & 0 \leq \alpha \leq C, \end{aligned}$$

- $Q_{ij} = y_i y_j k(x_i, x_j)$
- $Q \in \mathbb{R}^{n \times n}$ positive semidefinite matrix (**dense!**)
- $e = (1, \ldots, 1) \in \mathbb{R}^n$

Decomposition Approaches

- SOTA for SVM training: **Decomposition Methods**

Decomposition Approaches

- SOTA for SVM training: **Decomposition Methods**
  - working set (WS):
    - $W \subset \{1, \ldots, n\}$,
    - $\overline{W} = \{1, \ldots, n\} \setminus W$
    - $q = |W|$

Decomposition Approaches

- SOTA for SVM training: **Decomposition Methods**
  - working set (WS):
    - $W \subset \{1, \ldots, n\}$,
    - $\overline{W} = \{1, \ldots, n\} \setminus W$
    - $q = |W|$
  - subproblem:

$$\begin{aligned} \min_{\alpha_W} \; & f(\alpha_W; \alpha_{\overline{W}}^k) = \frac{1}{2} \alpha_W^T Q_{WW} \alpha_W + p_W^T \alpha_W \\ \text{s.t.} \; & y_W^T \alpha_W = -y_{\overline{W}}^T \alpha_{\overline{W}}^k, \\ & 0 \leq \alpha_W \leq C. \end{aligned}$$

Decomposition Approaches

- SOTA for SVM training: **Decomposition Methods**
  - working set (WS):
    - $W \subset \{1, \ldots, n\}$,
    - $\overline{W} = \{1, \ldots, n\} \setminus W$
    - $q = |W|$
  - subproblem:

$$\min_{\alpha_W} f(\alpha_W; \alpha_{\overline{W}}^k) = \frac{1}{2}\alpha_W^T Q_{WW}\alpha_W + p_W^T \alpha_W$$
$$\text{s.t. } y_W^T \alpha_W = -y_{\overline{W}}^T \alpha_{\overline{W}}^k,$$
$$0 \leq \alpha_W \leq C.$$

  - subproblems solved cheaply

Decomposition Approaches

- SOTA for SVM training: **Decomposition Methods**
  - working set (WS):
    - $W \subset \{1, \ldots, n\}$,
    - $\overline{W} = \{1, \ldots, n\} \setminus W$
    - $q = |W|$
  - subproblem:

$$\begin{aligned} \min_{\alpha_W} \; & f(\alpha_W; \alpha_{\overline{W}}^k) = \frac{1}{2} \alpha_W^T Q_{WW} \alpha_W + p_W^T \alpha_W \\ \text{s.t.} \; & y_W^T \alpha_W = -y_{\overline{W}}^T \alpha_{\overline{W}}^k, \\ & 0 \leq \alpha_W \leq C. \end{aligned}$$

  - subproblems solved cheaply
  - dynamically compute columns of Q
    - do not fill memory
    - only a small part of Q needed in the process

Decomposition Approaches

Algorithm 1: General Decomposition Framework for SVM Training

Input: $\alpha^0 = 0$, $\nabla f(\alpha^0) = -e$, $k = 0$

1: **while** *the stopping criterion is not satisfied* **do**
2: select the working set W^k
3: retrieve the columns Q_W
4: set $W = W^k$ and compute a solution $\alpha_W^\star$ of the subproblem with variables α_W
5: set $\alpha_i^{k+1} = \begin{cases} \alpha_i^\star & \text{for } i \in W \\ \alpha_i^k & \text{otherwise} \end{cases}$
6: set

$$\nabla f(\alpha^{k+1}) = \nabla f(\alpha^k) + Q_W(\alpha^{k+1} - \alpha^k)$$

$$= \nabla f(\alpha^k) + \sum_{i \in W} Q_i(\alpha_i^{k+1} - \alpha_i^k)$$

7: set $k = k + 1$
8: **end while**
9: **return** $\alpha^\star = \alpha^k$

Decomposition Approaches

- **Cases:**
  - $q = 1$ meaningless: $y^T \alpha = 0$.

Decomposition Approaches

- **Cases:**
  - $q = 1$ meaningless: $y^T \alpha = 0$.
  - $q = 2$: **Sequential Minimal Optimization (SMO)**

Decomposition Approaches

- **Cases:**
  - $q = 1$ meaningless: $y^T \alpha = 0$.
  - $q = 2$: **Sequential Minimal Optimization (SMO)**
    - closed form solution of subproblems!

Decomposition Approaches

- **Cases:**
  - $q = 1$ meaningless: $y^T \alpha = 0$.
  - $q = 2$: **Sequential Minimal Optimization (SMO)**
    - closed form solution of subproblems!
  - $q > 2$: requires solver.

Optimality Conditions

- KKTs hold if and only if $\alpha^\star$ is optimal;

Optimality Conditions

- KKTs hold if and only if $\alpha^\star$ is optimal;
- Stopping condition: small violation of KKTs.

Optimality Conditions

- KKTs hold if and only if $\alpha^\star$ is optimal;
- Stopping condition: small violation of KKTs.
- KKTs violation can be viewed in terms of "pairwise" violations

Optimality Conditions

- KKTs hold if and only if $\alpha^\star$ is optimal;
- Stopping condition: small violation of KKTs.
- KKTs violation can be viewed in terms of "pairwise" violations
  - the pair of variables that causes the largest violation of KKTs is called Most Violating Pair (MVP)

WSSR for SMO

- **WSS1:**
  - select MVP (classical rule)

WSSR for SMO

- **WSS1:**
  - select MVP (classical rule)
- **WSS2** (LIBSVM):
  - select 1 variable by WSS1,
  - then select the other one exploiting second order information;

WSSR for SMO

- **WSS1:**
  - select MVP (classical rule)
- **WSS2** (LIBSVM):
  - select 1 variable by WSS1,
  - then select the other one exploiting second order information;
- Both rules enjoy properties of:
  - asymptotic convergence;
  - finite termination.

WSSR for General Decomposition Approaches

- **Extended WSS1:** select $q/2$ most violating pairs
  - Asymptotic convergence
  - Finite termination

WSSR for General Decomposition Approaches

- **Extended WSS1:** select $q/2$ most violating pairs
  - Asymptotic convergence
  - Finite termination
- **MVP (or WSS2) + other variables**
  - finite termination (if subproblems were solved exactly)

WSSR for General Decomposition Approaches

- **Extended WSS1:** select $q/2$ most violating pairs
  - Asymptotic convergence
  - Finite termination
- **MVP (or WSS2) + other variables**
  - finite termination (if subproblems were solved exactly)
- **Additional variables** can be selected according to heuristic to exploit cached data (see works from L. Zanni, 2005-2006)

Reviving Non-SMO Approaches

- non-SMO solvers dismissed for decades...

Reviving Non-SMO Approaches

- non-SMO solvers dismissed for decades...
- ...although supposed to possess potentially useful features;

Reviving Non-SMO Approaches

- non-SMO solvers dismissed for decades...
- ...although supposed to possess potentially useful features;
- elements for an efficient non-SMO solver:
  - very efficient solver for subproblems;
  - smart selection of variables.

Reviving Non-SMO Approaches

- non-SMO solvers dismissed for decades...
- ...although supposed to possess potentially useful features;
- elements for an efficient non-SMO solver:
  - very efficient solver for subproblems;
  - smart selection of variables.
- Both requirements addressed in the following.

Solving subproblems by SMO

- Subproblem form:

$$\begin{aligned} \min_{\alpha_w} \ & \frac{1}{2}\alpha_w^T Q_{ww}\alpha_w + p_w^T \alpha_w \\ \text{s.t.} \ & y_w^T \alpha_w = -y_{\overline{w}}^T \alpha_{\overline{w}}^k, \\ & 0 \leq \alpha_w \leq C. \end{aligned}$$

Solving subproblems by SMO

- Subproblem form:

$$\begin{aligned} \min_{\alpha_w} \ & \frac{1}{2}\alpha_w^T Q_{ww}\alpha_w + p_w^T \alpha_w \\ \text{s.t.} \ & y_w^T \alpha_w = -y_{\overline{w}}^T \alpha_{\overline{w}}^k, \\ & 0 \leq \alpha_w \leq C. \end{aligned}$$

- Same form as the original problem!

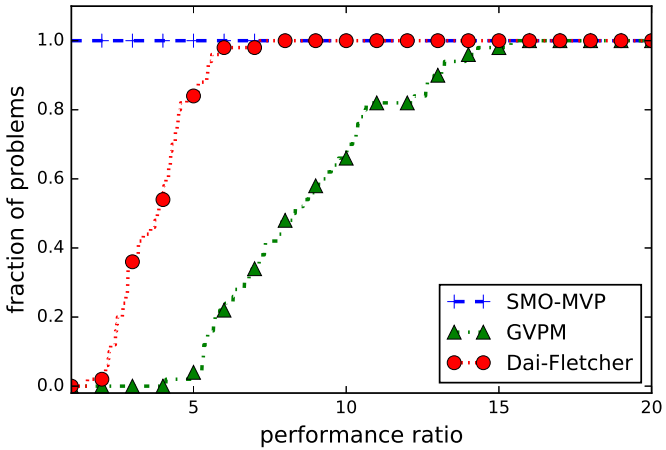
Solving subproblems by SMO

- Subproblem form:

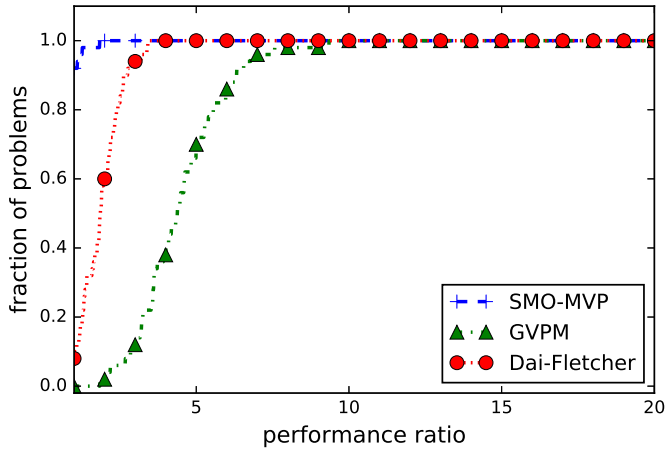
$$\begin{aligned} \min_{\alpha_W} \ & \frac{1}{2}\alpha_W^T Q_{WW}\alpha_W + p_W^T \alpha_W \\ \text{s.t.} \ & y_W^T \alpha_W = -y_{\overline{W}}^T \alpha_{\overline{W}}^k, \\ & 0 \leq \alpha_W \leq C. \end{aligned}$$

- Same form as the original problem!
- Decomposition can be applied recursively:
  - use SMO (WSS1) to solve the subproblem.

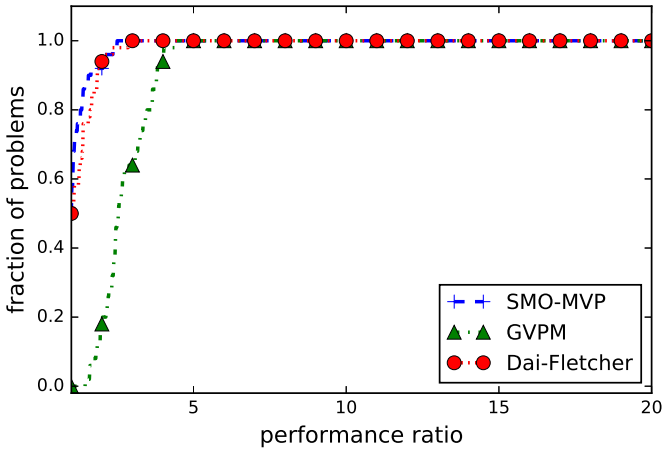
Numerical Experiments - Subproblems Solution



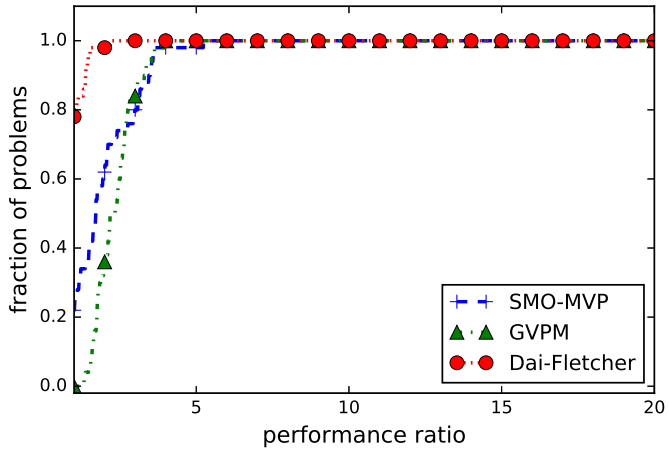
(a) $q = 4$ - runtime



(b) $q = 10$ - runtime

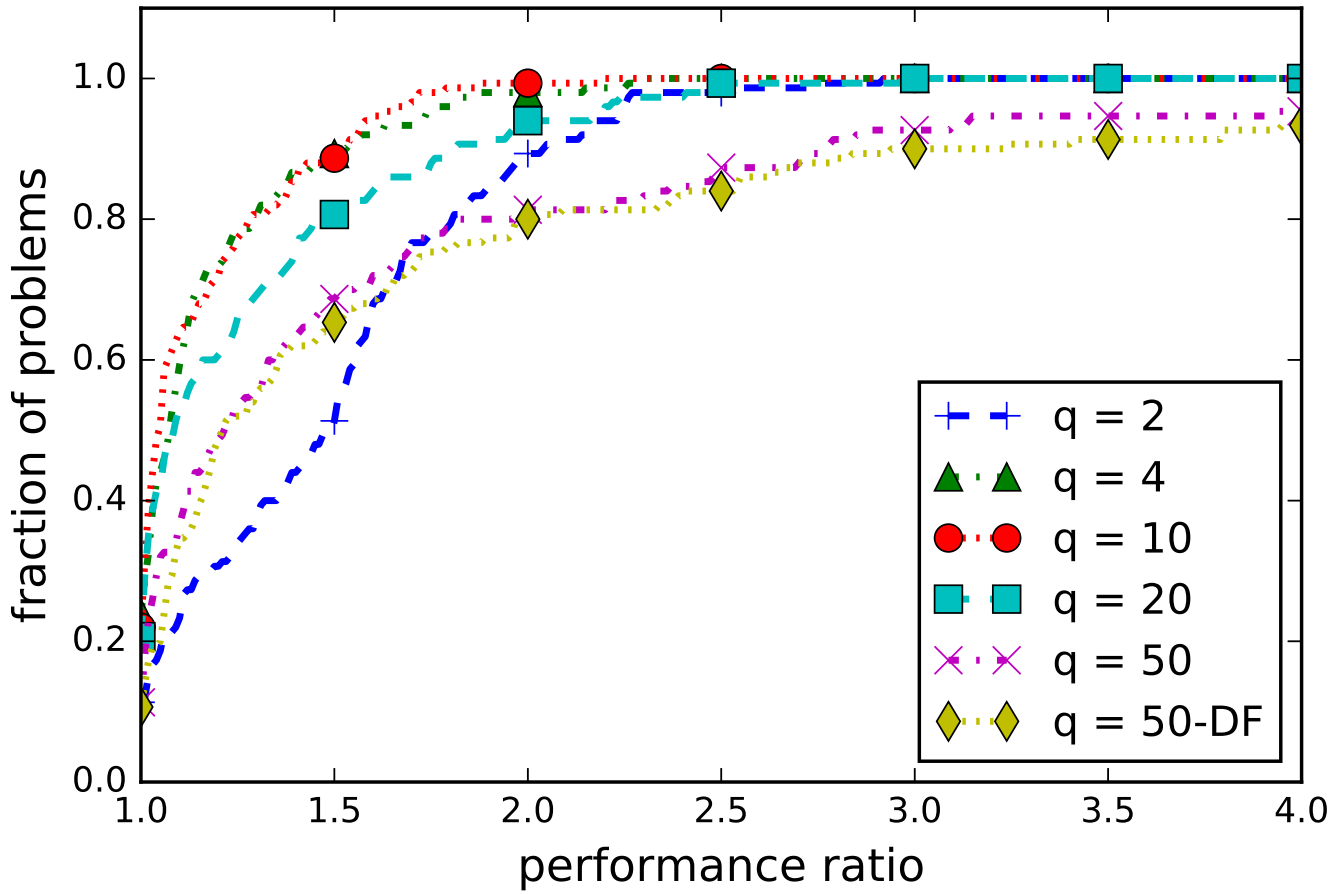


(c) $q = 20$ - runtime



(d) $q = 50$ - runtime

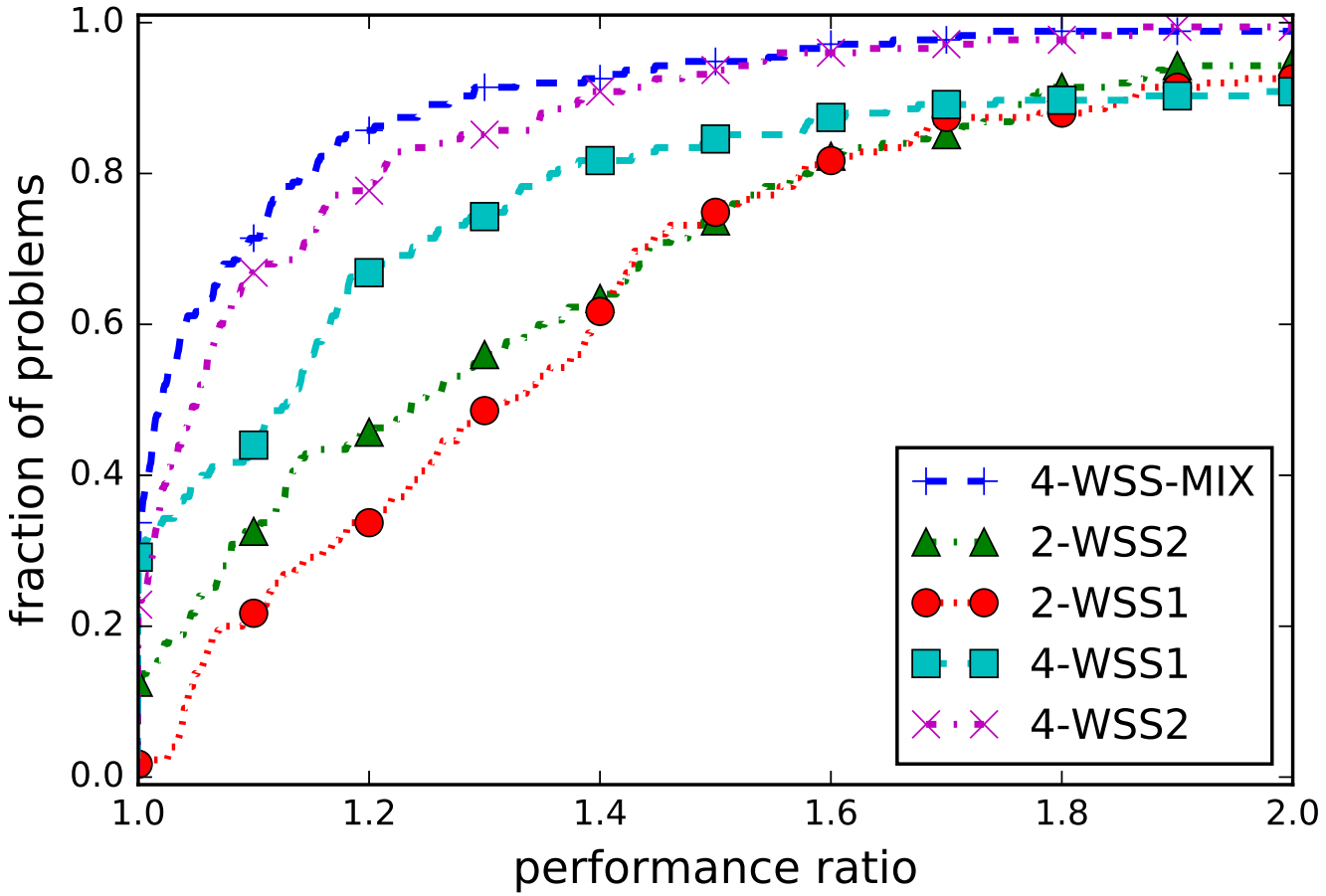
Numerical Experiments - Working Set Size



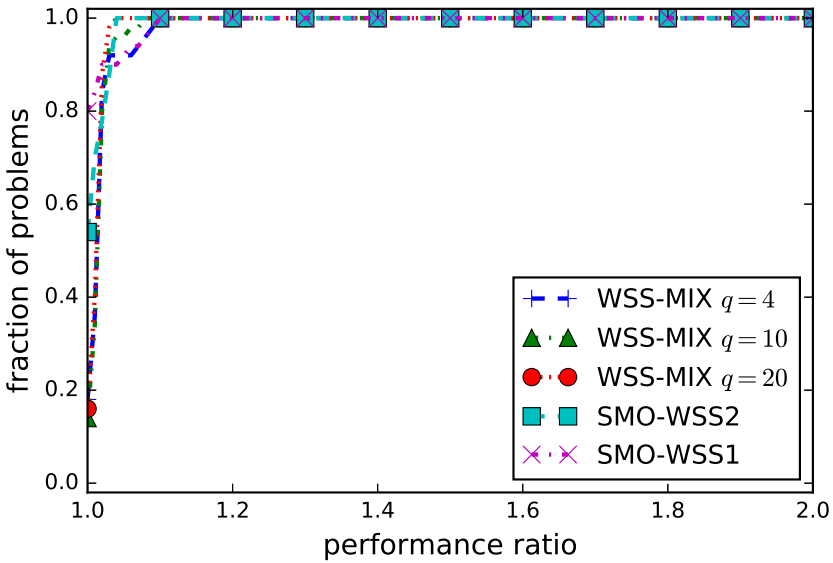
A composite WSSR

- q variables can be selected as follows:
  - Select one pair of variables by WSS2
    - for speed in difficult cases
  - Select one pair of variables by WSS1
    - useful variables, no additional cost for selection
  - Fill the WS up to q variables with the cache-oriented heuristic
    - use variables with no kernel computation cost

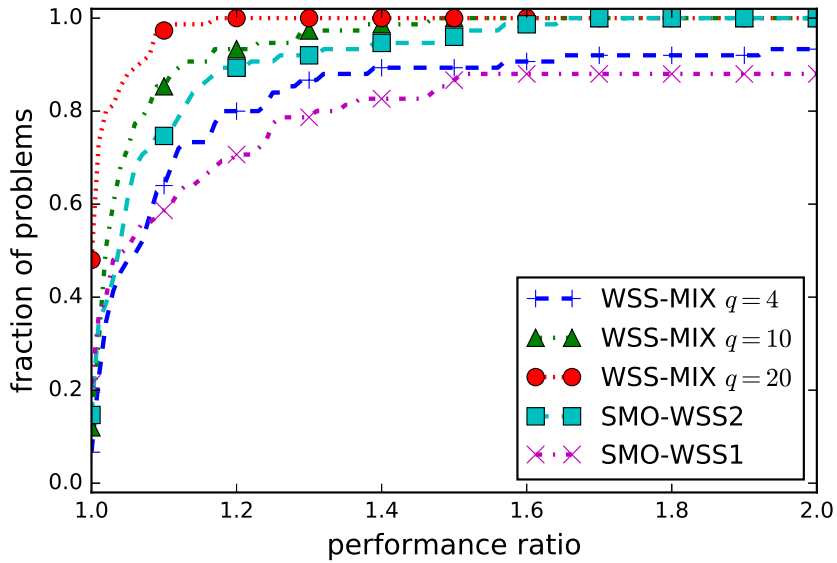
Numerical Expeiments - WS Selection Rules



Numerical Experiments - Cached Variables



(e) small problems



(f) large problems

The Proposed Algorithm

Algorithm 2: Two-Level Decomposition Method for SVM Training

Input: $\alpha^0 = 0$, $\nabla f(\alpha^0) = -e$, $k = 0$, $q \geq 4$

1: **while** KKT violation $> \epsilon$ **do**
2: select 4 variables to define the working set $W \subset \{1, \ldots, n\}$ according to WSS2 and WSS1
3: eventually add $q - 4$ variables to W, corresponding to cached columns of the Hessian matrix.
4: compute α_W^{k+1} by applying SMO to $\min_{\alpha_W \in \mathcal{F}(\alpha_{\overline{W}}^k)} f(\alpha_W; \alpha_{\overline{W}}^k)$
5: $\alpha_{\overline{W}}^{k+1} = \alpha_{\overline{W}}^k$
6: $\nabla f(\alpha^{k+1}) = \nabla f(\alpha^k) + \sum_{h \in W} Q_h(\alpha_h^{k+1} - \alpha_h^k)$
7: set $k = k + 1$
8: **end while**
9: **return** α^k

(Few) Theoretical Guarantees

- The inner SMO procedure terminates with an approximate KKT point for the subproblem in finite time;

(Few) Theoretical Guarantees

- The inner SMO procedure terminates with an approximate KKT point for the subproblem in finite time;
- The objective function is monotonically non increasing;

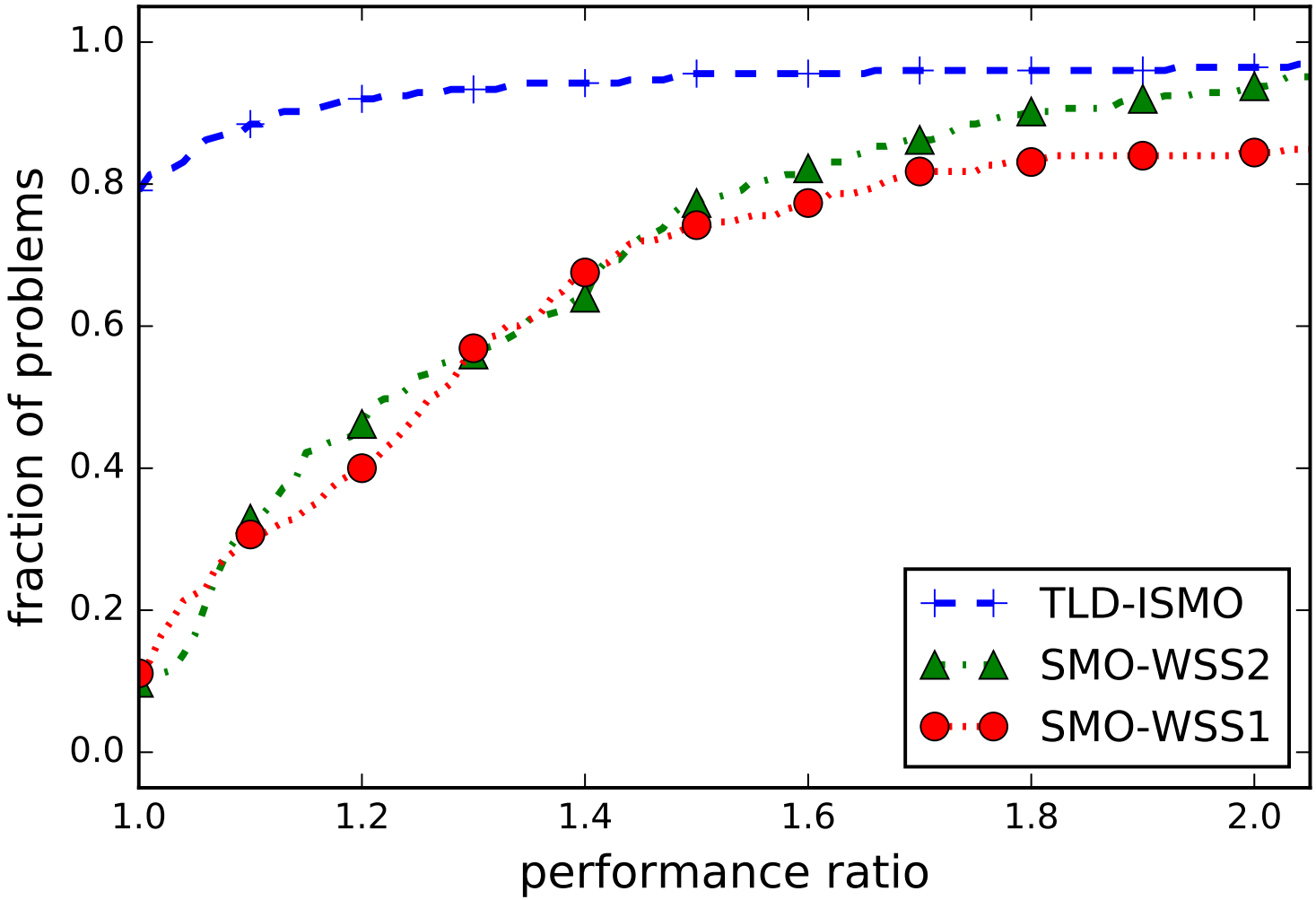
(Few) Theoretical Guarantees

- The inner SMO procedure terminates with an approximate KKT point for the subproblem in finite time;
- The objective function is monotonically non increasing;
- At each iteration the variables are updated;

(Few) Theoretical Guarantees

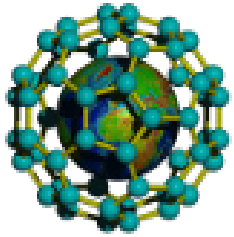
- The inner SMO procedure terminates with an approximate KKT point for the subproblem in finite time;
- The objective function is monotonically non increasing;
- At each iteration the variables are updated;
- The working set changes at every iteration, so that it cannot happen that the algorithm infinitely loops on the same sub-problem.

Numerical Experiments - Overall



Conclusions

- Non-SMO Decomposition has potential to speed up nonlinear SVM training;
- Decomposition can be applied recursively to solve subproblems
  - Open question: are other additional levels of decomposition useful?
- Large working sets allow to exploit variables selected by different rationale.
  - Optimal trade-off between iterations, kernel computations and time spent selecting variables.
- See *"A Two-Level Decomposition Framework Exploiting First and Second Order Information for SVM Training Problems.", JMLR 22 (2021): 23-1* for further discussions and deep numerical analyses.



GOL

A Two-Level Decomposition Framework Exploiting First and Second Order Information for SVM Training Problems

SIMAI Congress 2020+2021, 31st August 2021

Giulio Galvan, **Matteo Lapucci**, Chih-Jen Lin, Marco Sciandrone

DINFO, Univerisity of Florence

Numerical Experiments - Setup

- Benchmark

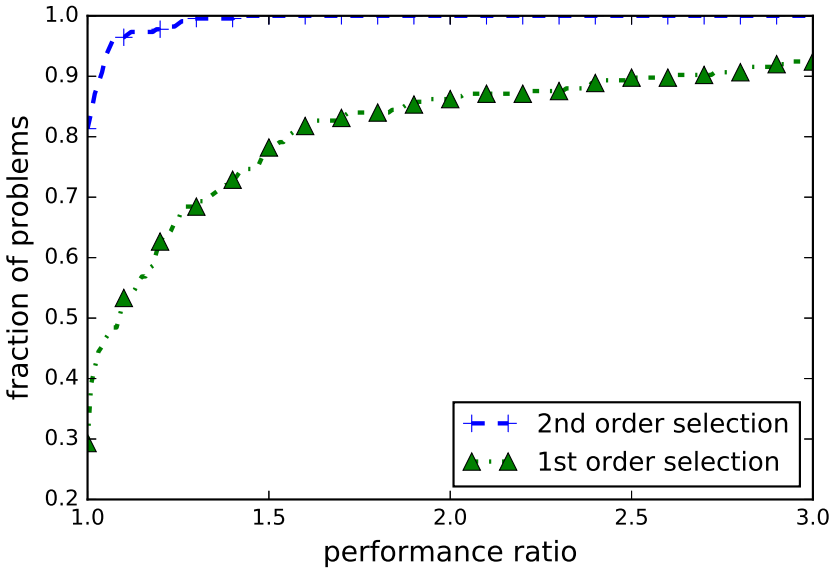
Data set	Training Size	Features	Class
splice	1,000	60	small
a1a	1,605	123	small
leukemia	38	7,129	small
a9a	32,561	123	medium size
w8a	49,749	300	medium-size
ijcnn1	49,990	22	medium-size
rcv1.binary	20,242	47,236	large
real-sim	72,309	20,958	large
covtype.binary	581,012	54	huge

- 25 RBF kernel problems for each data set (different γ and C)
- 225 test problems
- Comparison by Performance Profiles

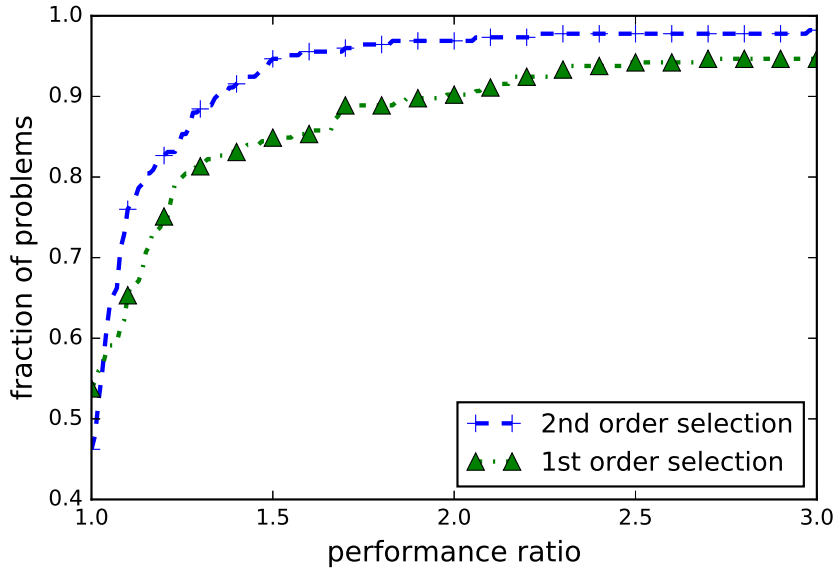
Numerical Experiments - Overall

dataset	TLD-ISMO	SMO-WSS1	SMO-WSS2
a1a	**2.20**	2.79	2.64
splice	**1.43**	1.68	1.78
leukemia	**0.51**	0.52	0.56
w8a	**1808.37**	2213.93	2275.07
ijcnn1	**2023.05**	5109.01	2541.04
a9a	**3003.09**	7708.22	4203.10
rcv1	**2432.89**	2585.81	2509.33
real-sim	**22780.93**	24116.95	24495.85
covtype	**518847.84**	2213224.58	651239.04
epsilon-subset	**182127.99**	217016.02	191457.61
phishing	**64.64**	81.12	80.64
cod-rna	**1999.17**	26490.30	4698.16
skin_nonskin	**10465.35**	12810.54	13364.90
cifar-resnetv2-emb	**25876.53**	35630.19	29719.72
news20	**10704.85**	11263.44	10980.03
SUSY-subset	**715706 (3)**	918867 (5)	746725 (3)
HIGGS-subset	**1011667 (3)**	1166775 (5)	1119047 (5)

WS Selection Rules

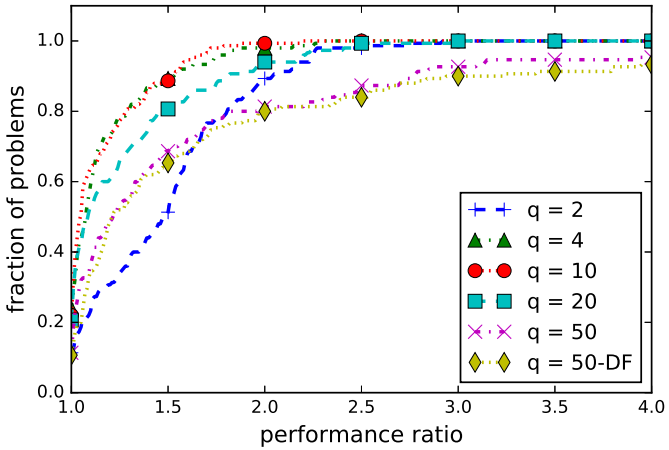


(g) Number of iterations.

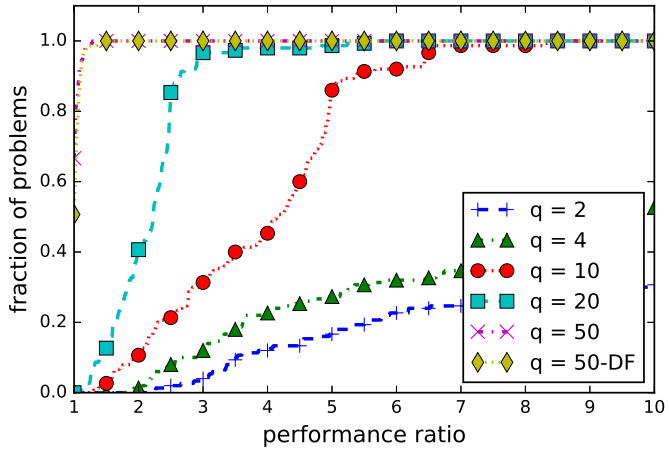


(h) Runtime.

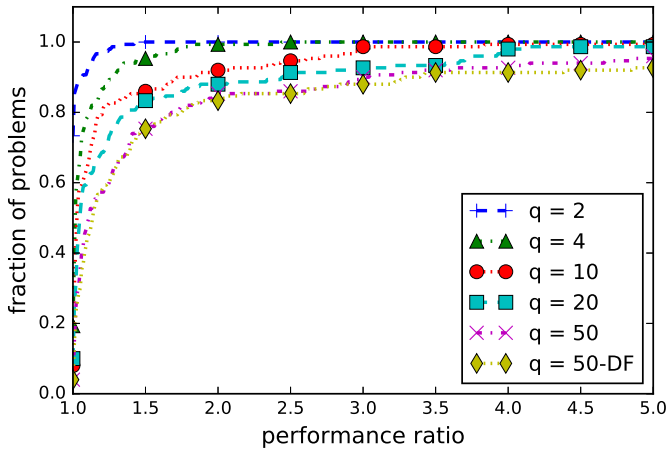
Numerical Experiments - Working Set Size



(i) Runtime

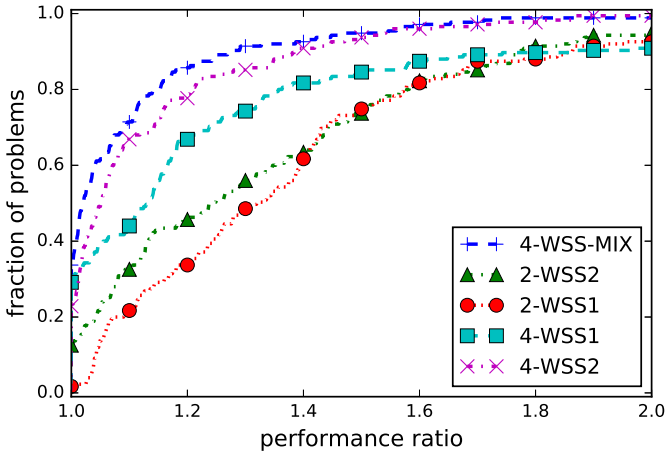


(j) (Outer) iterations

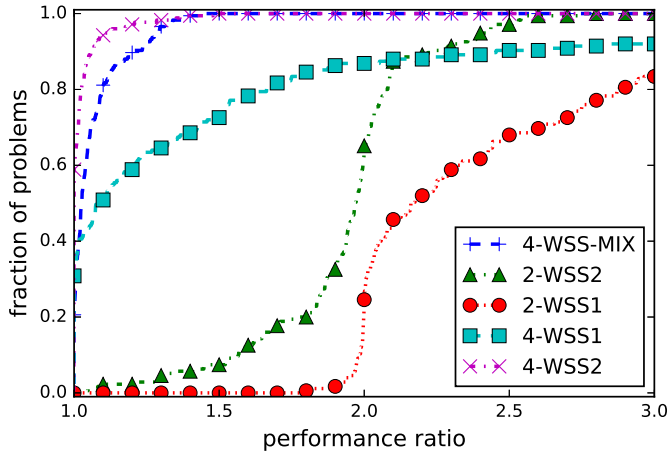


(k) Kernel evaluations.

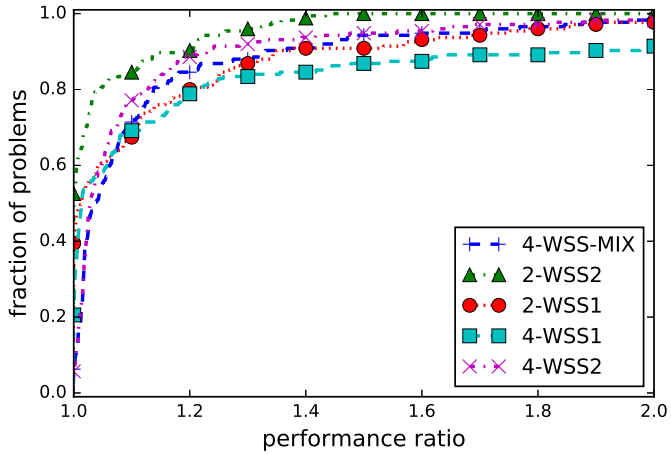
Numerical Expeiments - WS Selection Rules



(l) runtime



(m) iterations



(n) kernel evaluations