Distributed Optimization with Arbitrary Local Solvers: CoCoA+ and Beyond

Martin Takáč

joint work with

- Chenxin Ma, Lehigh University
- Martin Jaggi, ETH, Zurich
- Nathan Srebro, Toyota Technological Institute at Chicago
- Virginia Smith, UC Berkeley
- Michael I. Jordan, UC Berkeley
- Peter Richtarik, University of Edinburgh
- Jakub Konecny, University of Edinburgh

Hace 4 aos: "Hola, soy Mike!"

3rd IMA Conference on Numerical Linear Algebra and Optimisation
10 – 12 September 2012, University of Birmingham

Outline

- Problem Formulation & Motivation (5 min)
- Classical "Distributed" Methods (4.5 min)
- CoCoA$^+$ Framework (7 min)
- Computation vs. Communication Trade-off (4 min)
- Numerical Experiments (3.5 min)
- How to do it Better (ongoing work) (6 min)
- Questions (5 min)

The Problem - Regularized Empirical Loss Minimization

$$\min_{\mathbf{w} \in \mathbf{R}^d} \mathbf{E}_{(x,y) \sim X,Y} \left[\ell(\mathbf{w}^T x; y) \right] \tag{1}$$

The Problem - Regularized Empirical Loss Minimization

Let $\{(x_i, y_i)\}_{i=1}^n$ be our training data, $x_i \in \mathbf{R}^d$ and $y_i \in \mathbf{R}$.

$$\min_{\mathbf{w} \in \mathbf{R}^d} \left[P(\mathbf{w}) := \frac{1}{n} \sum_{i=1}^{n} \ell_i(\mathbf{w}^T x_i) + \frac{\lambda}{2} \|w\|^2 \right] \tag{P}$$

- $\lambda > 0$ is a regularization parameter
- $\ell_i(\cdot)$ is convex loss function which can depend on the label y_i
  Examples:
  - Logistic loss: $\ell_i(\zeta) = \log(1 + \exp(-y_i \zeta))$
  - Hinge loss: $\ell_i(\zeta) = \max\{0, 1 - y_i \zeta\}$

The Problem - Regularized Empirical Loss Minimization

Let $\{(x_i, y_i)\}_{i=1}^n$ be our training data, $x_i \in \mathbf{R}^d$ and $y_i \in \mathbf{R}$.

$$\min_{\mathbf{w} \in \mathbf{R}^d} \left[P(\mathbf{w}) := \frac{1}{n} \sum_{i=1}^{n} \ell_i(\mathbf{w}^T x_i) + \frac{\lambda}{2} \|w\|^2 \right] \tag{P}$$

- $\lambda > 0$ is a regularization parameter
- $\ell_i(\cdot)$ is convex loss function which can depend on the label y_i
  Examples:
  - Logistic loss: $\ell_i(\zeta) = \log(1 + \exp(-y_i \zeta))$
  - Hinge loss: $\ell_i(\zeta) = \max\{0, 1 - y_i \zeta\}$

The dual problem

$$\max_{\alpha \in \mathbf{R}^n} \left[D(\alpha) := -\frac{\lambda}{2} \|A\alpha\|^2 - \frac{1}{n} \sum_{i=1}^{n} \ell_i^*(-\alpha_i) \right] \tag{D}$$

where $A = \frac{1}{\lambda n} X^T$ and $X^T = [x_1, x_2, \ldots, x_n] \in \mathbf{R}^{d \times n}$

- ℓ_i^* is convex conjugate of ℓ_i
- wlog $\|x_i\| \leq 1$

Duality

Primal-Dual mapping

For any $\alpha \in \text{dom}(D)$ we can define

$$\mathbf{w}(\alpha) := A\alpha \tag{1}$$

From strong duality we have that $w^* = w(\alpha^*)$ is optimal to (P) if α^* is optimal solution to (D).

Duality

Primal-Dual mapping

For any $\alpha \in \text{dom}(D)$ we can define

$$\mathbf{w}(\alpha) := A\alpha \tag{1}$$

From strong duality we have that $w^* = w(\alpha^*)$ is optimal to (P) if α^* is optimal solution to (D).

Gap function

$$G(\alpha) = P(\mathbf{w}(\alpha)) - D(\alpha)$$

The Setting & Challenges

Recent interest in machine learning:

- the size of matrix A is **huge** (e.g. TBs of data)
- we **have to** use many nodes of computer cluster (or cloud) to speed-up the computation and leverage the problem size

Challenges

- **distributed data:** no single machine can load the whole instance
- **expensive communication:**

	latency
RAM	**100** nanoseconds
standard network connection	**250,000** nanoseconds

- **unreliable nodes:** we assume that the node can die at any point during the computation (we want to have fault tolerant solution)
- **reuse of good solvers:** we want to use highly tuned and customized single machine solvers developed over many years (SAGA, SGD, SVRG, mSDCA, mS2GD, MISO, ...)

Classical "Distributed" Optimization Alg. – Primal Space

Many algorithms works as follows

- split data $\{(x_i, y_i)\}_{i=1}^n$ across K computers (nodes)
- each node will solve some problem depending on some data stored locally
- **Question: How to combine the optimal local solutions?**

Classical "Distributed" Optimization Alg. – Primal Space

Many algorithms works as follows

- split data $\{(x_i, y_i)\}_{i=1}^n$ across K computers (nodes)
- each node will solve some problem depending on some data stored locally
- **Question: How to combine the optimal local solutions?**

Consensus based approach

- impose some constraint that the local solutions from each node should be the same

Classical "Distributed" Optimization Alg. – Primal Space

Many algorithms works as follows

- split data $\{(x_i, y_i)\}_{i=1}^n$ across K computers (nodes)
- each node will solve some problem depending on some data stored locally
- **Question: How to combine the optimal local solutions?**

Consensus based approach

- impose some constraint that the local solutions from each node should be the same

Parameter server approach

- one node (master) has the vector w
- other nodes (workers) have the data
- workers ask master about coordinates of w and also tells to master which changes should be made

Classical "Distributed" Optimization Alg. – Dual Space

- objective function is $D(\alpha)$
- $\alpha \in \mathbf{R}^n$
- we can split the coordinates of α across K nodes
- we split the data matrix accordingly (coordinates corresponds to samples)
- each node **k** can find some direction $\mathbf{\Delta}\alpha_{\mathbf{k}}$ how to decrease $D(\cdot)$ by changing the local coordinates
- **Question: How to combine the optimal local solutions?**

Classical "Distributed" Optimization Alg. – Dual Space

- objective function is $D(\alpha)$
- $\alpha \in \mathbf{R}^n$
- we can split the coordinates of α across K nodes
- we split the data matrix accordingly (coordinates corresponds to samples)
- each node **k** can find some direction $\mathbf{\Delta}\alpha_{\mathbf{k}}$ how to decrease $D(\cdot)$ by changing the local coordinates
- **Question: How to combine the optimal local solutions?**
- Easy solution: define a new iterate as the old one + **average** of locally computed solutions

Runtime Analysis - From Signle node to Distributed

Classical Single Node

- iterative optimization algorithm $\mathcal{A}$
- $\mathcal{T}_{\mathcal{A}}$ – time it takes to perform a single iteration of algorithm $\mathcal{A}$
- $\mathcal{I}_{\mathcal{A}}(\epsilon)$ is the number of iterations $\mathcal{A}$ needs to attain an ϵ-accurate objective

$$\text{TIME}(\mathcal{A}) = \mathcal{I}_{\mathcal{A}}(\epsilon) \times \mathcal{T}_{\mathcal{A}}\,.$$

Runtime Analysis - From Signle node to Distributed

Classical Single Node

- iterative optimization algorithm $\mathcal{A}$
- $\mathcal{T}_{\mathcal{A}}$ – time it takes to perform a single iteration of algorithm $\mathcal{A}$
- $\mathcal{I}_{\mathcal{A}}(\epsilon)$ is the number of iterations $\mathcal{A}$ needs to attain an ϵ-accurate objective

$$\text{TIME}(\mathcal{A}) = \mathcal{I}_{\mathcal{A}}(\epsilon) \times \mathcal{T}_{\mathcal{A}}\,.$$

Distributed Algorithm

- c – the time required to perform one round of communication

$$\text{TIME}(\mathcal{A}) = \mathcal{I}_{\mathcal{A}}(\epsilon) \times (\mathbf{c} + \mathcal{T}_{\mathcal{A}})\,.$$

Runtime Analysis - From Signle node to Distributed

Distributed Algorithm

- c – the time required to perform one round of communication

$$\text{TIME}(\mathcal{A}) = \mathcal{I}_{\mathcal{A}}(\epsilon) \times (\mathbf{c} + \mathcal{T}_{\mathcal{A}}) .$$

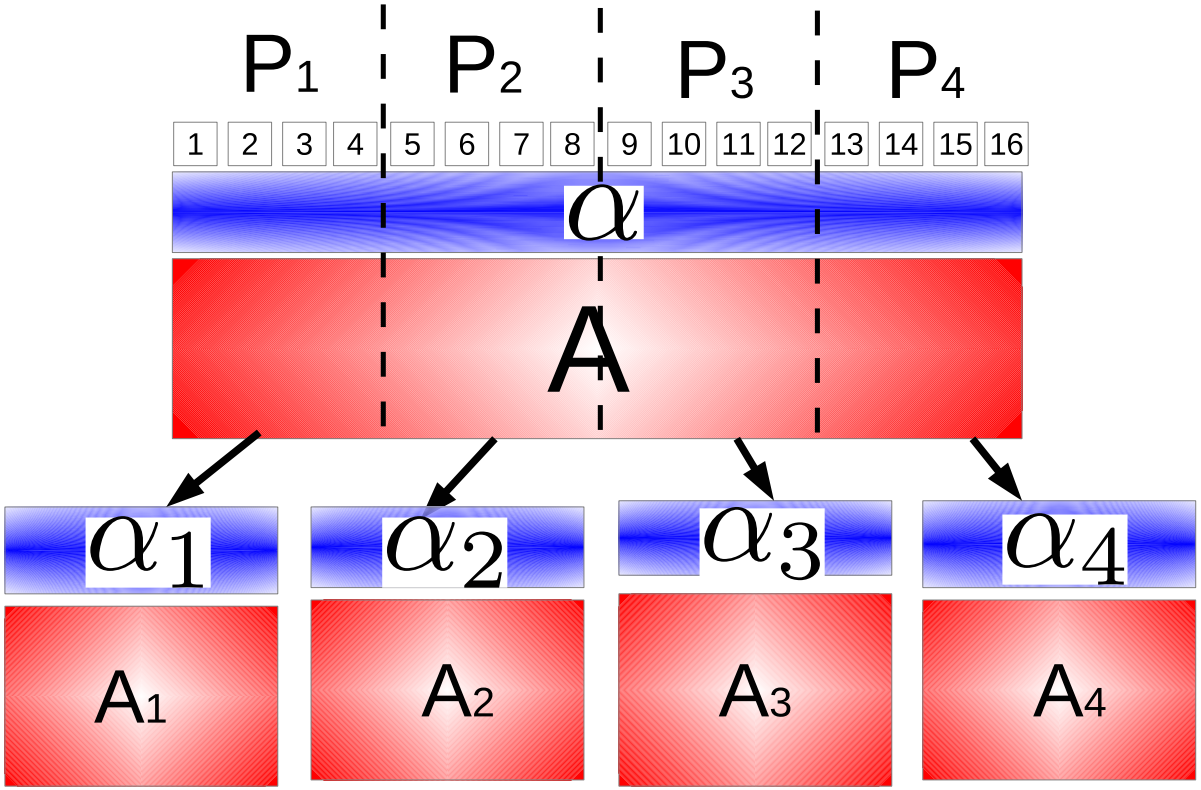
Distributed Algorithm with Weak Local Solutions

- $\mathcal{T}_{\mathcal{A}}(\Theta)$ – the time the local algorithm $\mathcal{A}$ needs to obtain accuracy Θ on the local subproblem
- $\mathcal{I}(\epsilon, \Theta)$ – the number of outer iterations
- c – the time required to perform one round of communication

$$\text{TIME}(\mathcal{A}, \Theta) = \mathcal{I}(\epsilon, \Theta) \times (\mathbf{c} + \mathcal{T}_{\mathcal{A}}(\Theta)) .$$

Data Distribution

Vector α and columns of matrix A are partitioned according $\{\mathcal{P}_k\}_{k=1}^K$.



Notation: For $k \in \{1, 2, \ldots, K\}$ we use $\alpha_k \in \mathbf{R}^{|\mathcal{P}_k|}$ is a subvector of α.
Vector $\alpha_{[k]} \in \mathbf{R}^n$ is a vector obtained from vector α by setting all coordinates $\notin \mathcal{P}_k$ to zero.
Example: $\alpha_1 = (*, *, *, *)^T$, $\alpha_{[1]} = (*, *, *, *, 0, 0, \ldots, 0)^T$.

Local Problem

CoCoA subproblem

At iteration t at node k

$$(\Delta\alpha^*)_{[k]}^{(t)} = \arg\max_{\Delta\alpha_{[k]}\in\mathbf{R}^n} D(\alpha^{(t)} + \Delta\alpha_{[k]})$$

$$= \arg\max_{\Delta\alpha_{[k]}\in\mathbf{R}^n} \left(-\frac{\lambda}{2}\|A(\alpha^{(t)} + \Delta\alpha_{[k]})\|^2 - \frac{1}{n}\sum_{i=1}^{n} \ell_i^*(-(\alpha^{(t)} + \Delta\alpha_{[k]})_i) \right)$$

- we cannot solve the subproblem as it depends on $\alpha^{(t)}$ and A

Local Problem

CoCoA subproblem

At iteration t at node k

$$(\Delta\alpha^*)_{[k]}^{(t)} = \arg\max_{\Delta\alpha_{[k]}\in\mathbf{R}^n} D(\alpha^{(t)} + \Delta\alpha_{[k]})$$

$$= \arg\max_{\Delta\alpha_{[k]}\in\mathbf{R}^n} \left(-\frac{\lambda}{2}\|A(\alpha^{(t)} + \Delta\alpha_{[k]})\|^2 - \frac{1}{n}\sum_{i=1}^{n} \ell_i^*(-(\alpha^{(t)} + \Delta\alpha_{[k]})_i) \right)$$

- we cannot solve the subproblem as it depends on $\alpha^{(t)}$ and A
- if we know $w^{(t)} = A\alpha^{(t)}$ then

$$(\Delta\alpha^*)_{[k]}^{(t)} = \arg\max_{\Delta\alpha_{[k]}\in\mathbf{R}^n} \left(-\frac{\lambda}{2}\|w^{(t)} + A\Delta\alpha_{[k]}\|^2 - \frac{1}{n}\sum_{i\in\mathcal{P}_k} \ell_i^*(-(\alpha^{(t)} + \Delta\alpha_{[k]})_i) \right)$$

- if we know $w^{(t)}$ we can compute $(\Delta\alpha^*)_{[k]}^{(t)}$

The CoCoA Framework

Communication-Efficient Distributed Dual Coordinate Ascent

Input: $T \geq 1$

Data: $\{(x_i, y_i)\}_{i=1}^n$ distributed over K machines

Initialize: $\alpha_{[k]}^{(0)} \leftarrow 0$ for all machines k, and $w^{(0)} \leftarrow 0$

for $t = 1, 2, \ldots, T$

  for all machines $k = 1, 2, \ldots, K$ **in parallel**

    Solve local problem approximately to obtain $\Delta\alpha_{[k]}$ computation

    $\alpha_{[k]}^{(t)} \leftarrow \alpha_{[k]}^{(t-1)} + \frac{1}{K}\Delta\alpha_{[k]}$

    $\Delta w_k \leftarrow \frac{1}{K} A\Delta\alpha_{[k]}$

  reduce $w^{(t)} \leftarrow w^{(t-1)} + \sum_{k=1}^K \Delta w_k$ communication

The CoCoA Framework

Communication-Efficient Distributed Dual Coordinate Ascent

Input: $T \geq 1$

Data: $\{(x_i, y_i)\}_{i=1}^n$ distributed over K machines

Initialize: $\alpha_{[k]}^{(0)} \leftarrow 0$ for all machines k, and $w^{(0)} \leftarrow 0$

for $t = 1, 2, \ldots, T$

  for all machines $k = 1, 2, \ldots, K$ **in parallel**

    Solve local problem approximately to obtain $\Delta\alpha_{[k]}$ computation

    $\alpha_{[k]}^{(t)} \leftarrow \alpha_{[k]}^{(t-1)} + \frac{1}{K}\Delta\alpha_{[k]}$

    $\Delta w_k \leftarrow \frac{1}{K} A\Delta\alpha_{[k]}$

  reduce $w^{(t)} \leftarrow w^{(t-1)} + \sum_{k=1}^K \Delta w_k$ communication

Few comments

- The performance of this methods (in worst case) can be the same as if we **randomly pick** k and solve corresponding subproblem and replace $\frac{1}{K}$ by 1
- How accurately do we need to solve the local sub-problem?
- How to change the local problem to avoid averaging (e.g. just to add local solutions)?

Smarter Subproblem

Local Subproblem for CoCoA$^+$

$$\max_{\Delta\alpha_{[k]}\in\mathbf{R}^n} \mathcal{G}_k^{\sigma'}(\Delta\alpha_{[k]}; w^{(t)}) \tag{2}$$

where

$$\mathcal{G}_k^{\sigma'}(\Delta\alpha_{[k]}; w^{(t)}) = -\frac{\lambda}{2}\|w^{(t)} + A\Delta\alpha_{[k]}\|^2 - \frac{1}{n}\sum_{i\in\mathcal{P}_k}\ell_i^*(-(\alpha_{[k]}^{(t)} + \Delta\alpha_{[k]})_i) - \frac{1}{K}\frac{\lambda}{2}\|w^{(t)}\|^2 - \frac{\lambda}{2}(\sigma' - 1)\left\|A\Delta\alpha_{[k]}\right\|^2.$$

and $\sigma' \geq 1$ will be explained soon

Compare with:

$$\max_{\Delta\alpha_{[k]}\in\mathbf{R}^n}\left(-\frac{\lambda}{2}\|w^{(t)} + A\Delta\alpha_{[k]}\|^2 - \frac{1}{n}\sum_{i\in\mathcal{P}_k}\ell_i^*(-(\alpha^{(t)} + \Delta\alpha_{[k]})_i)\right) \tag{3}$$

If $\sigma' = 1$ then the optimal solutions of (2) and (3) coincides.

The CoCoA$^+$ Framework

Communication-Efficient Distributed Dual Coordinate Ascent

Input: $T \geq 1$, $\gamma \in [\frac{1}{K}, 1]$, $\sigma' \in [1, \infty)$
Data: $\{(x_i, y_i)\}_{i=1}^n$ distributed over K machines
Initialize: $\alpha_{[k]}^{(0)} \leftarrow 0$ for all machines k, and $w^{(0)} \leftarrow 0$
for $t = 1, 2, \ldots, T$
  for all machines $k = 1, 2, \ldots, K$ **in parallel**
    approximately $\max \mathcal{G}^{\sigma'}(\Delta\alpha_{[k]}; w^{(t)})$ to obtain $\Delta\alpha_{[k]}$ computation
    $\alpha_{[k]}^{(t)} \leftarrow \alpha_{[k]}^{(t-1)} + \gamma\Delta\alpha_{[k]}$
    $\Delta w_k \leftarrow \gamma A \Delta\alpha_{[k]}$
  reduce $w^{(t)} \leftarrow w^{(t-1)} + \sum_{k=1}^K \Delta w_k$ communication

- If $\gamma = \frac{1}{K}$ we obtain CoCoA
- If $\gamma = \frac{1}{K}$ then $\sigma' = 1$ is "safe" value
- What about another values of γ? (we want $\gamma = 1$)

CoCoA$^+$ Parameters - σ' and γ

- σ' measures the difficulty of the given data partition
- it must be chosen not smaller than

$$\sigma' \geq \sigma'_{min} \stackrel{\text{def}}{=} \gamma \max_{\alpha \in \mathbf{R}^n} \frac{\|\mathbf{A}\alpha\|^2}{\sum_{k=1}^{K} \|\mathbf{A}\alpha_{[k]}\|^2} \tag{4}$$

Lemma

For any $\alpha \in \mathbf{R}^n$ $(\alpha \neq \mathbf{0})$ we have

$$\frac{\|\mathbf{A}\alpha\|^2}{\sum_{k=1}^{K} \|\mathbf{A}\alpha_{[k]}\|^2} \leq K$$

- We can take the safe value $\sigma' = K \cdot \gamma$
  Again: if $\gamma = \frac{1}{K}$ then $\sigma' = K \cdot \frac{1}{K} = 1$ is a safe value
- New: if $\gamma = 1$ then $\sigma' = K \cdot 1 = K$ is a safe value

CoCoA$^+$ Parameters - σ' and γ

- σ' measures the difficulty of the given data partition
- it must be chosen not smaller than

$$\sigma' \geq \sigma'_{min} \stackrel{\text{def}}{=} \gamma \max_{\alpha \in \mathbf{R}^n} \frac{\|\mathbf{A}\alpha\|^2}{\sum_{k=1}^{K} \|\mathbf{A}\alpha_{[k]}\|^2} \tag{4}$$

Lemma

For any $\alpha \in \mathbf{R}^n$ $(\alpha \neq \mathbf{0})$ we have

$$\frac{\|\mathbf{A}\alpha\|^2}{\sum_{k=1}^{K} \|\mathbf{A}\alpha_{[k]}\|^2} \leq K$$

- We can take the safe value $\sigma' = K \cdot \gamma$
  Again: if $\gamma = \frac{1}{K}$ then $\sigma' = K \cdot \frac{1}{K} = 1$ is a safe value
- New: if $\gamma = 1$ then $\sigma' = K \cdot 1 = K$ is a safe value
- **If $A^T A$ is block diagonal, then $\sigma'_{\min} = \gamma$**

How Accurately?

Assumption: Θ-approximate solution

We assume that there exists $\Theta \in [0, 1)$ such that $\forall k \in [K]$, the local solver at any iteration t produces a **(possibly) randomized** approximate solution $\Delta\alpha_{[k]}$, which satisfies

$$\mathbf{E}\left[\mathcal{G}_k^{\sigma'}(\Delta\alpha_{[k]}^*, w) - \mathcal{G}_k^{\sigma'}(\Delta\alpha_{[k]}, w)\right] \leq \Theta\left(\mathcal{G}_k^{\sigma'}(\Delta\alpha_{[k]}^*, w) - \mathcal{G}_k^{\sigma'}(\mathbf{0}, w)\right), \tag{5}$$

where

$$\Delta\alpha^* \in \arg\min_{\Delta\alpha \in \mathbf{R}^n} \sum_{k=1}^{K} \mathcal{G}_k^{\sigma'}(\Delta\alpha_{[k]}, w). \tag{6}$$

- because the subproblem is **not really** what one wants to solve, therefore in practise $\Theta \approx 0.9$ (depending on the cluster and problem)
- what about convergence guarantees?
- how to get Θ approximate solution?

Iteration Complexity - Smooth Loss

Theorem

Assume the loss functions functions ℓ_i are $(1/\mu)$-smooth, for $i \in \{1, 2, \ldots, n\}$. We define

$$\sigma_k \stackrel{def}{=} \max_{\alpha_{[k]} \in \mathbf{R}^n} \frac{\|A\alpha_{[k]}\|^2}{\|\alpha_{[k]}\|^2} \leq |\mathcal{P}_k| \tag{7}$$

and $\sigma_{\max} = \max_{k \in [K]} \sigma_k$.
Then after T iterations of CoCoA$^+$, with

$$T \geq \frac{1}{\gamma(1-\Theta)} \frac{\lambda\mu n + \sigma_{\max}\sigma'}{\lambda\mu n} \log \frac{1}{\epsilon},$$

it holds that $\mathbf{E}[D(\alpha^*) - D(\alpha^T)] \leq \epsilon$.
Furthermore, after T iterations with

$$T \geq \frac{\mathbf{1}}{\gamma(\mathbf{1}-\mathbf{\Theta})} \frac{\lambda\mu\mathbf{n} + \sigma_{\max}\sigma'}{\lambda\mu\mathbf{n}} \log \left(\frac{1}{\gamma(1-\Theta)} \frac{\lambda\mu n + \sigma_{\max}\sigma'}{\lambda\mu n} \frac{1}{\epsilon} \right),$$

we have the expected duality gap

$$\mathbf{E}[\mathbf{P}(\mathbf{w}(\alpha^{(\mathbf{T})})) - \mathbf{D}(\alpha^{(\mathbf{T})})] \leq \epsilon.$$

Averaging vs. Adding

The leading term is $\frac{1}{\gamma(1-\Theta)} \frac{\lambda\mu n+\sigma_{\max}\sigma'}{\lambda\mu n}$. Let us assume that $\forall k : |\mathcal{P}_k| = \frac{n}{K}$

Averaging

$\gamma = \frac{1}{K}$
$\sigma' = 1$

$$\frac{K}{1-\Theta} \frac{\lambda\mu n+\frac{n}{K}}{\lambda\mu n}$$

$$\frac{1}{1-\Theta} \frac{\lambda\mu K+1}{\lambda\mu}$$

Adding

$\gamma = 1$
$\sigma' = K$

$$\frac{1}{1-\Theta} \frac{\lambda\mu n+\frac{n}{K}K}{\lambda\mu n}$$

$$\frac{1}{1-\Theta} \frac{\lambda\mu+1}{\lambda\mu}$$

Note: this is in the worst case (for the worst case example)

Iteration Complexity - General Convex Loss

Theorem

Consider CoCoA$^+$ starting with $\alpha^0 = \mathbf{0} \in \mathbf{R}^n$ and $\forall i \in \{1, 2, \ldots, n\} : \ell_i(\cdot)$ be L-Lipschitz continuous and $\epsilon > 0$ be the desired duality gap. Then after T iterations, where

$$T \geq T_0 + \max\{\left\lceil \frac{1}{\gamma(1-\Theta)} \right\rceil, \frac{4L^2\sigma\sigma'}{\lambda n^2 \epsilon \gamma(1-\Theta)}\},$$

$$T_0 \geq t_0 + \left(\frac{2}{\gamma(1-\Theta)} \left(\frac{8L^2\sigma\sigma'}{\lambda n^2 \epsilon} - 1 \right) \right)_+,$$

$$t_0 \geq \max(0, \left\lceil \tfrac{1}{\gamma(1-\Theta)} \log(\tfrac{2\lambda n^2 (D(\alpha^*) - D(\alpha^0))}{4L^2\sigma\sigma'}) \right\rceil),$$

we have that the expected duality gap satisfies $\mathbf{E}[P(w(\overline{\alpha})) - D(\overline{\alpha})] \leq \epsilon$, at the averaged iterate

$$\overline{\alpha} := \tfrac{1}{T-T_0} \sum_{t=T_0+1}^{T-1} \alpha^{(t)},$$

where $\sigma = \sum_{k=1}^{K} |\mathcal{P}_k| \sigma_k$.

SDCA as a Local Solver

SDCA

1: **Input:** $\alpha_{[k]}, w = w(\alpha)$
2: **Data:** Local $\{(x_i, y_i)\}_{i \in \mathcal{P}_k}$
3: **Initialize:** $\Delta\alpha^0_{[k]} = 0 \in \mathbb{R}^n$
4: **for** $h = 0, 1, \ldots, H-1$ **do**
5: choose $i \in \mathcal{P}_k$ uniformly at random
6: $\delta_i^* = \arg\max_{\delta_i \in \mathbf{R}} \mathcal{G}_k^{\sigma'}(\Delta\alpha^h_{[k]} + \delta_i e_i, w)$
7: $\Delta\alpha^{(h+1)}_{[k]} = \Delta\alpha^{(h)}_{[k]} + \delta_i^* e_i$
8: **end for**
9: **Output:** $\Delta\alpha^{(H)}_{[k]}$

Theorem

Assume the functions ℓ_i are $(1/\mu)-$smooth for $i \in \{1, 2, \ldots, n\}$. If

$$H \geq n_k \frac{\sigma' + \lambda n \mu}{\lambda n \mu} \log \frac{1}{\Theta} \tag{8}$$

then SDCA will produce a Θ-approximate solution.

Total Runtime

- To get ϵ accuracy we need

$$\mathcal{O}\left(\frac{1}{1-\Theta}\log\frac{1}{\epsilon}\right)$$

- Recall $\Theta = \left(1 - \frac{\lambda n \gamma}{1+\lambda n \gamma}\frac{K}{n}\right)^H$

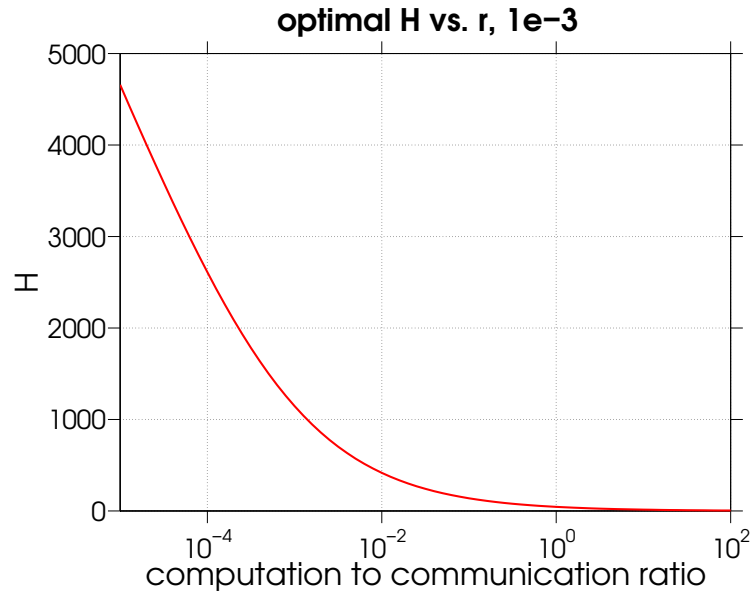
Let

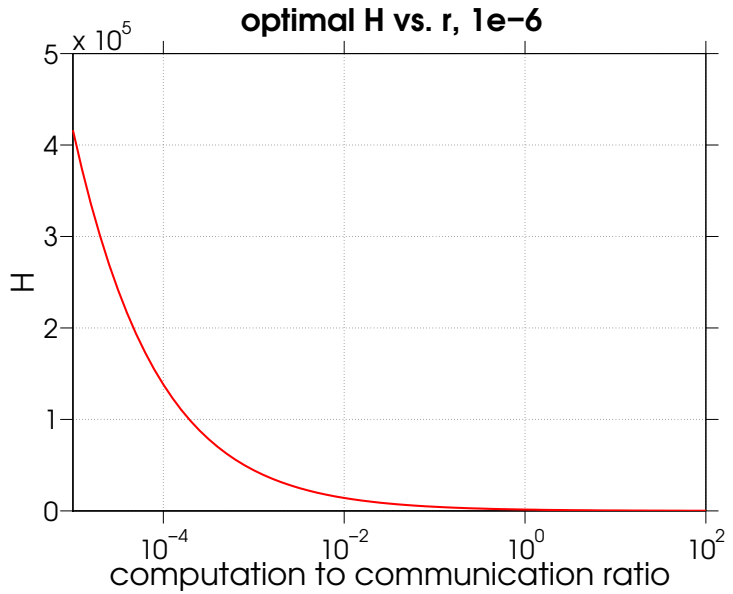
- τ_o be the duration of communication per iteration
- τ_c be the duration of **ONE** coordinate update during the inner iteration

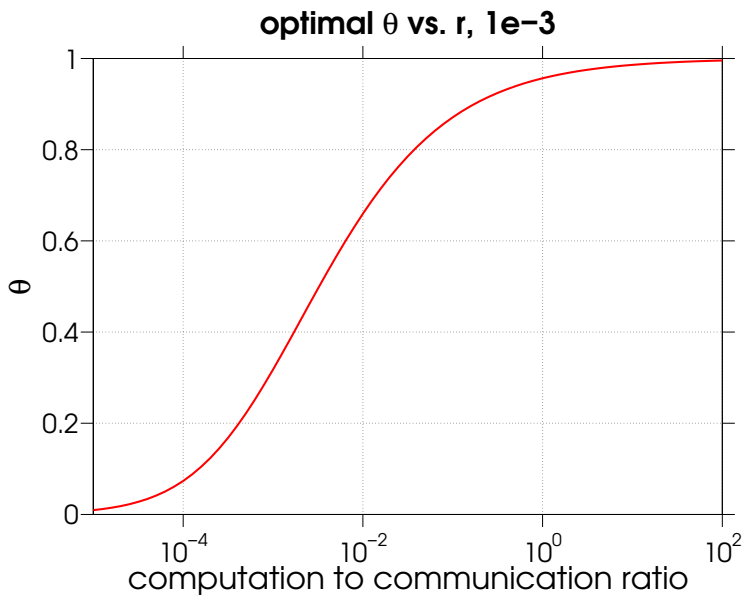
Total runtime

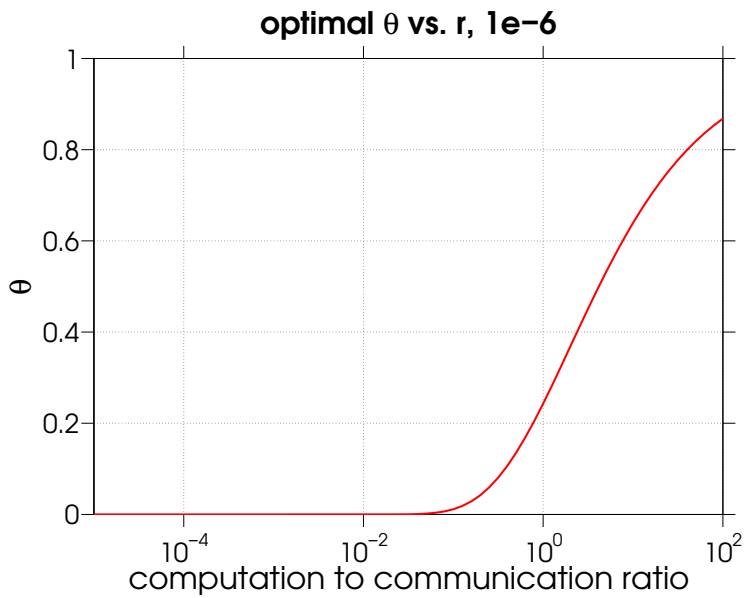
$$\mathcal{O}\left(\frac{1}{1-\Theta}(\tau_O + H\tau_c)\right) = \mathcal{O}\left(\frac{1}{1-\Theta}\left(1 + H\underbrace{\frac{\tau_c}{\tau_o}}_{r_{c/o}}\right)\right)$$

$H(\tau_c/\tau_o)$, $\Theta(\tau_c/\tau_o)$









Numerical Experiments

Datasets

Dataset	n	d	size(GB)
rcv1test	677,399	47,236	1.2
epsilon	400,000	2,000	3.1
splice-site.t	4,627,840	11,725,480	273.4

Numerical Experiments

Datasets

Dataset	n	d	size(GB)
rcv1test	677,399	47,236	1.2
epsilon	400,000	2,000	3.1
splice-site.t	4,627,840	11,725,480	273.4

Local Solvers

RCDM	Randomized Coordinate Descent
APPROX	Accelerated, Parallel and Proximal Coordinate Descent
GD	Gradient Descent with Backtracking Line Search
CG	Conjugate Gradient Method
L-BFGS	Quasi-Newton with Limited-Memory BFGS Updating
BB	Barzilai-Borwein Gradient Method
FISTA	Fast Iterative Shrinkage-Thresholding Algorithm

Numerical Experiments

Datasets

Dataset	n	d	size(GB)
rcv1test	677,399	47,236	1.2
epsilon	400,000	2,000	3.1
splice-site.t	4,627,840	11,725,480	273.4

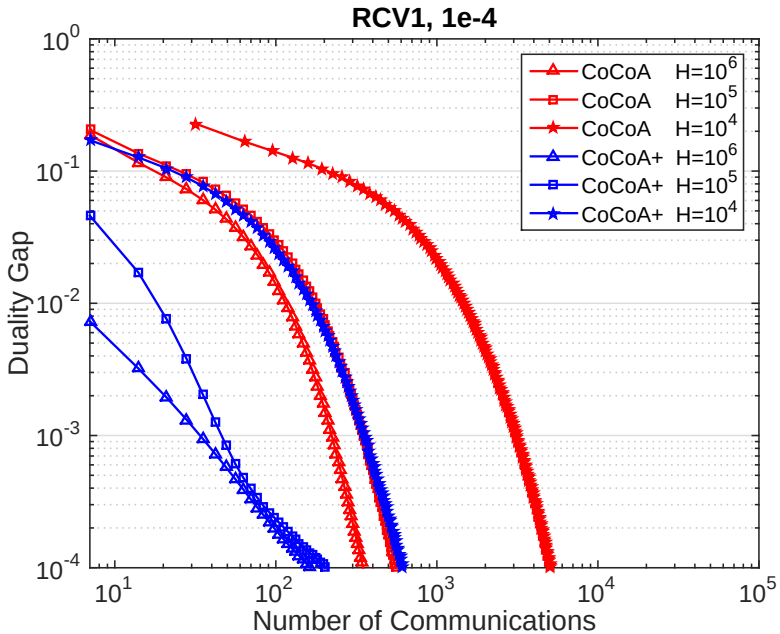
Local Solvers

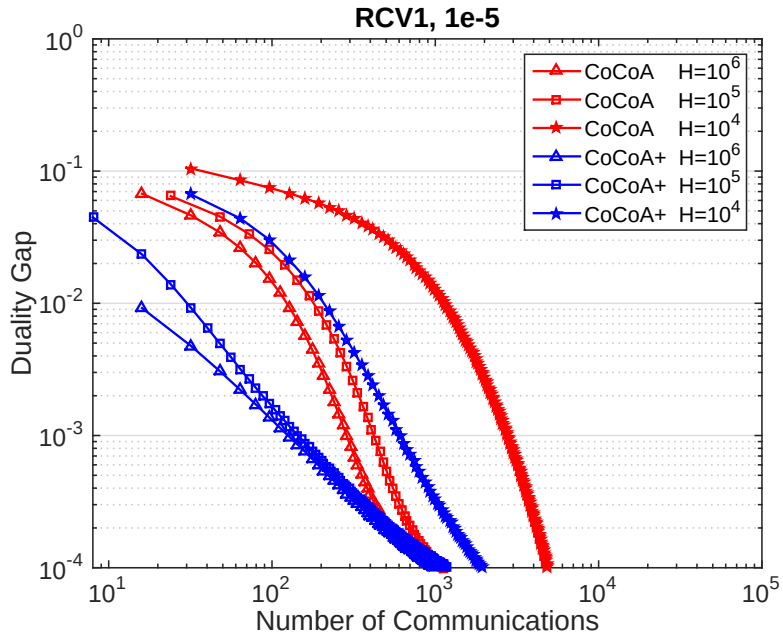
RCDM	Randomized Coordinate Descent
APPROX	Accelerated, Parallel and Proximal Coordinate Descent
GD	Gradient Descent with Backtracking Line Search
CG	Conjugate Gradient Method
L-BFGS	Quasi-Newton with Limited-Memory BFGS Updating
BB	Barzilai-Borwein Gradient Method
FISTA	Fast Iterative Shrinkage-Thresholding Algorithm

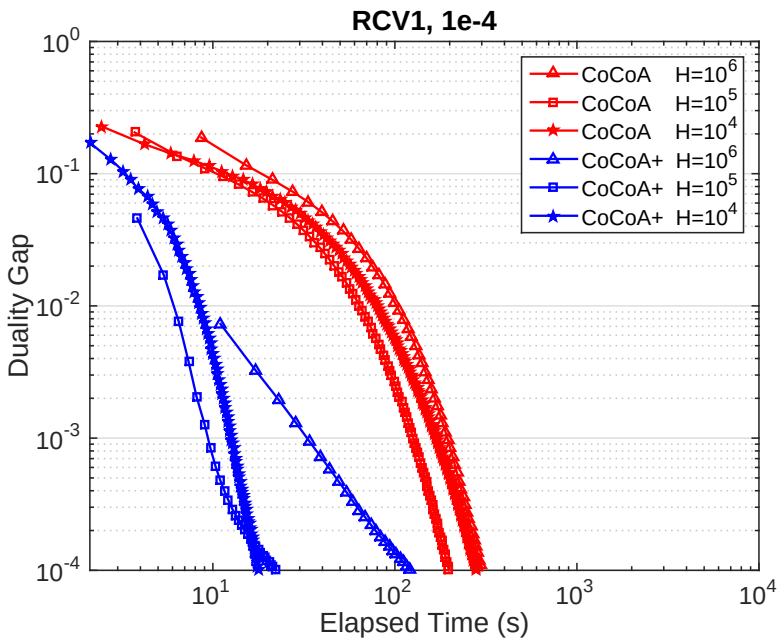
PS: we tried different parameters for local solvers. The best parameters are here:

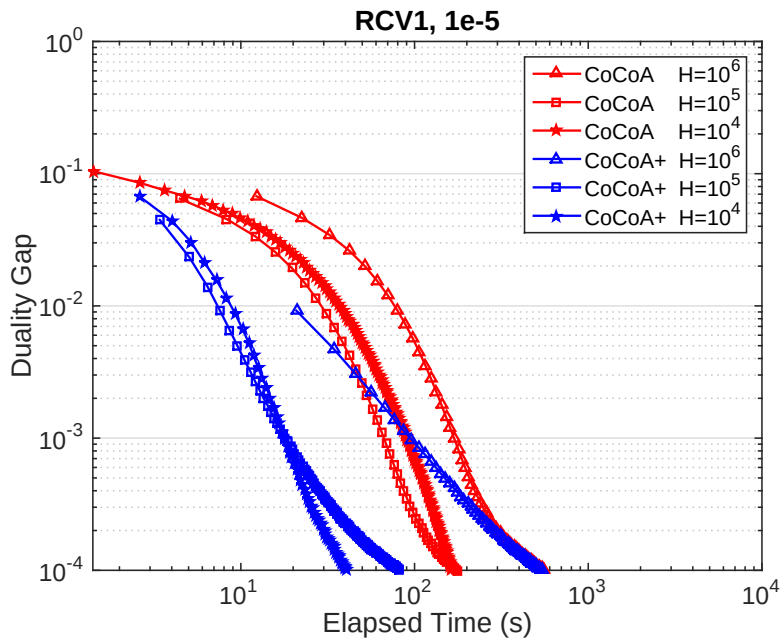
Local Solver	RCDM	APPROX	GD	CG	L-BFGS	BB	FISTA
H	40,000	40,000	20	5	10	15	20

CoCoA vs. CoCoA$^+$ - SDCA as a Local Solver

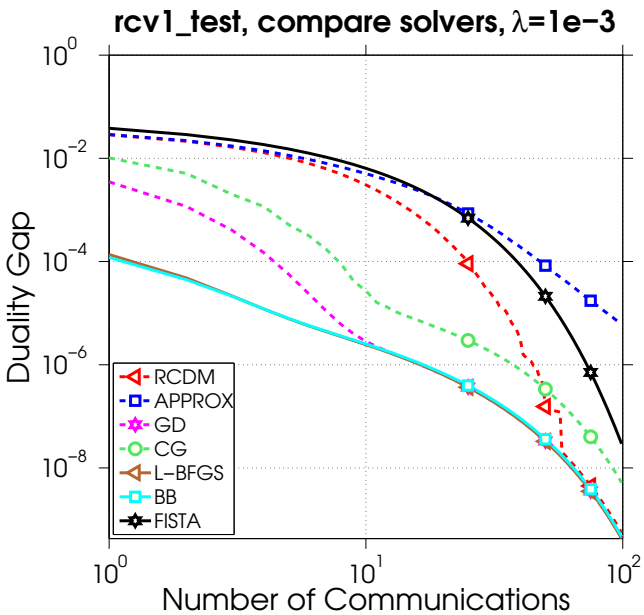


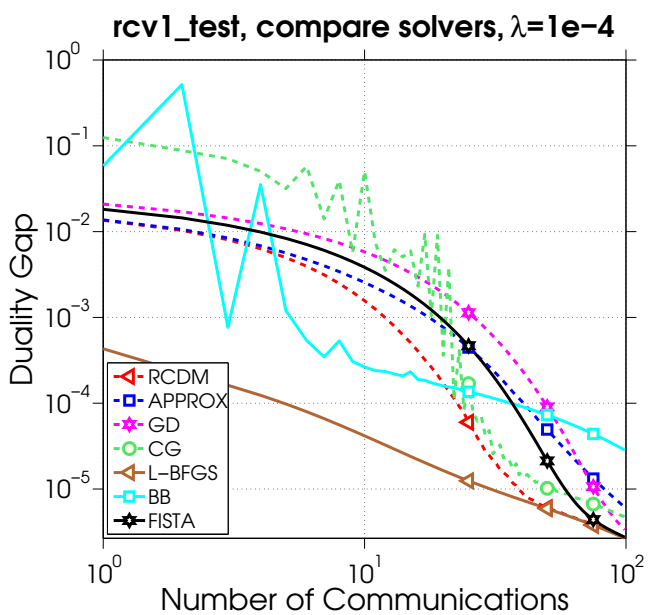


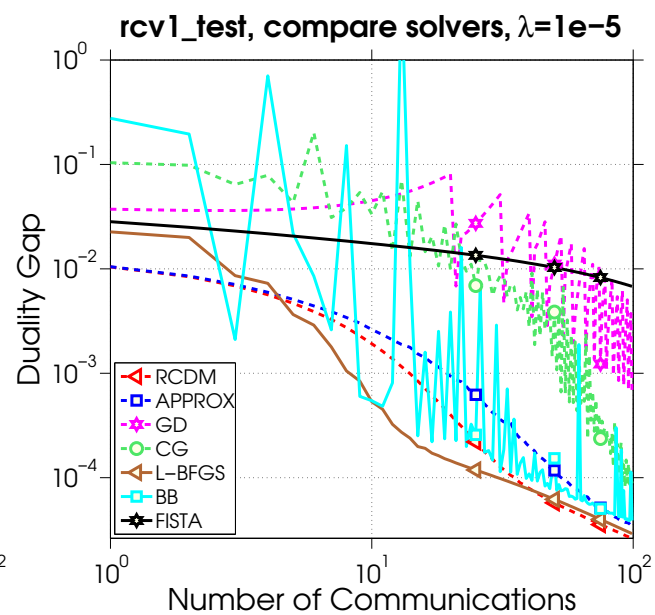




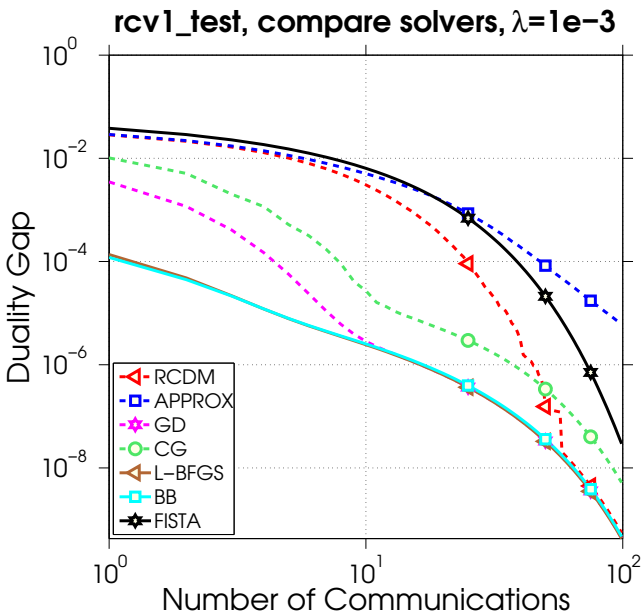
CoCoA$^+$: Various Solvers

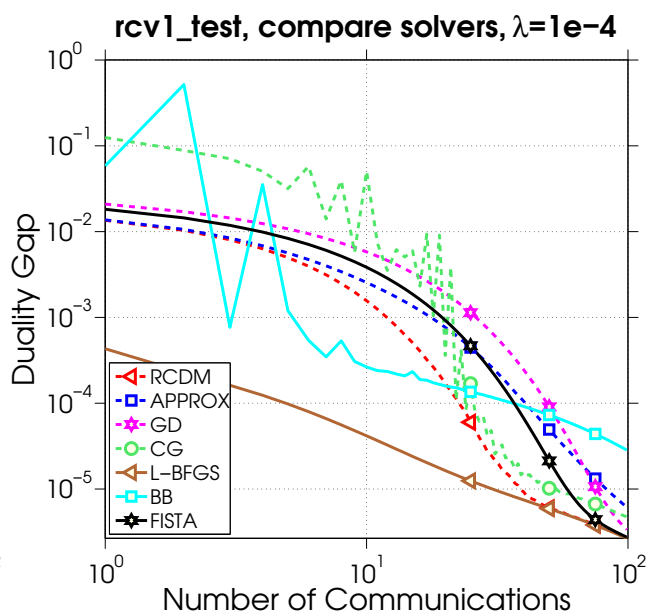


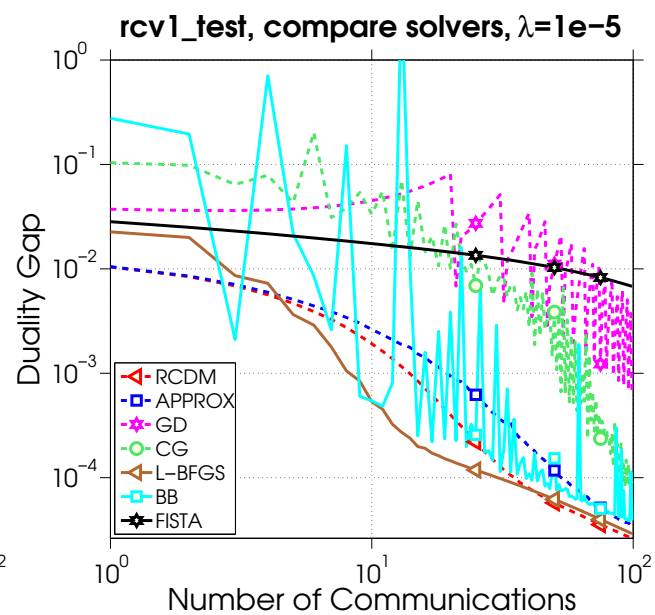


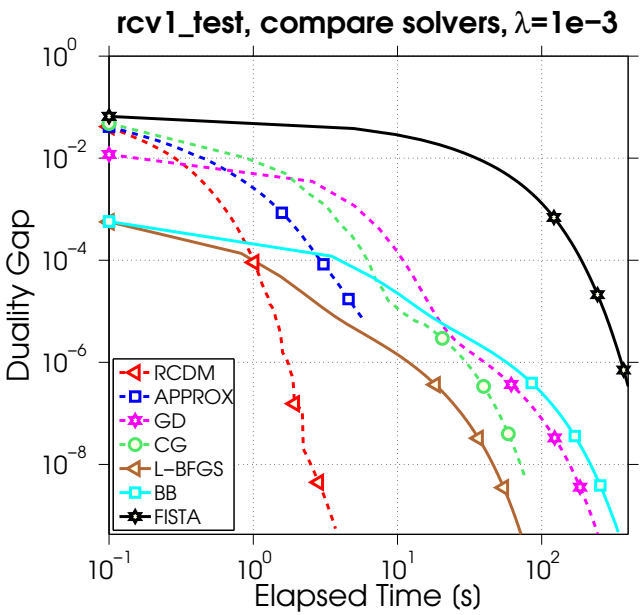


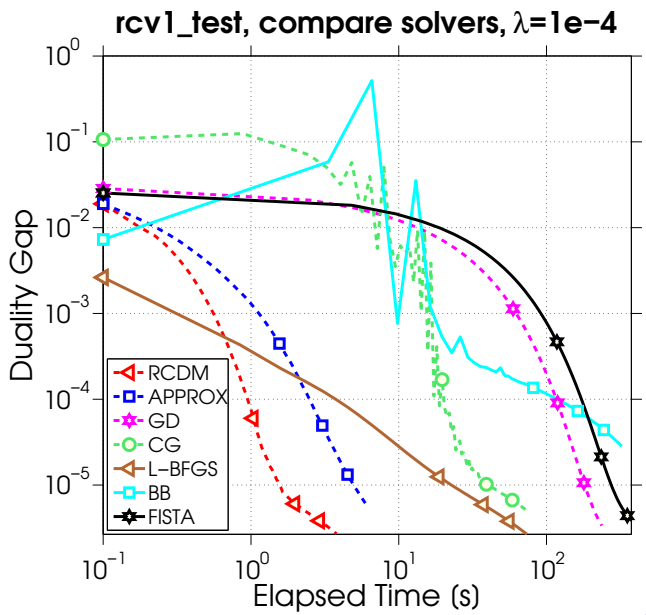
CoCoA$^+$: Various Solvers

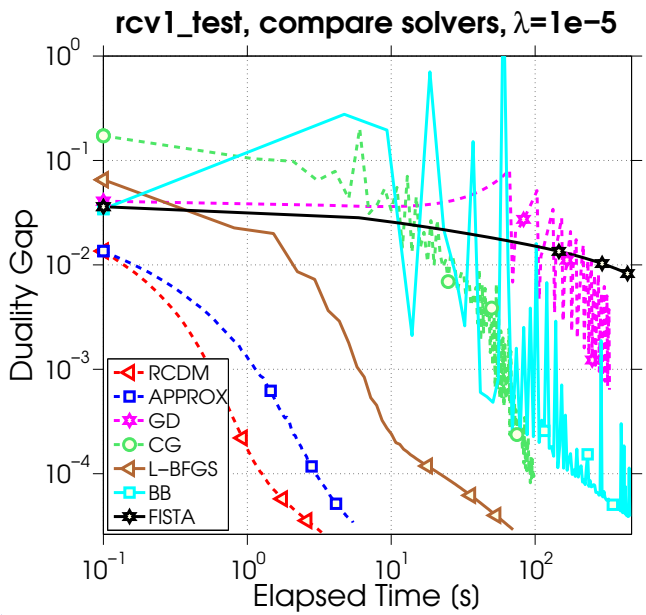




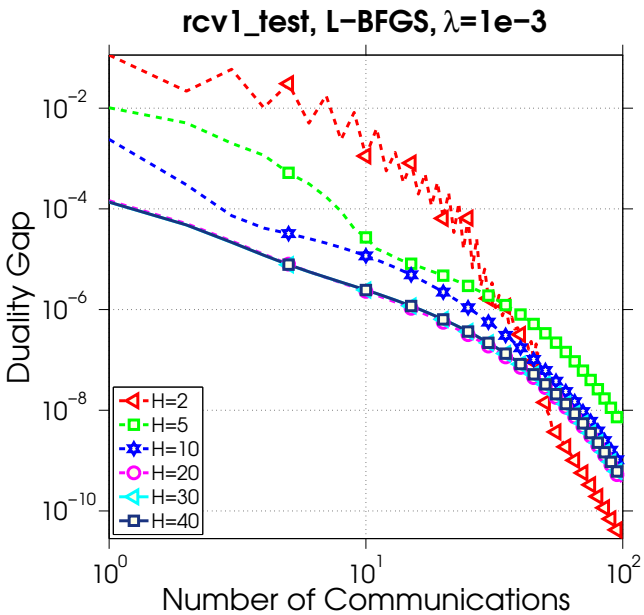


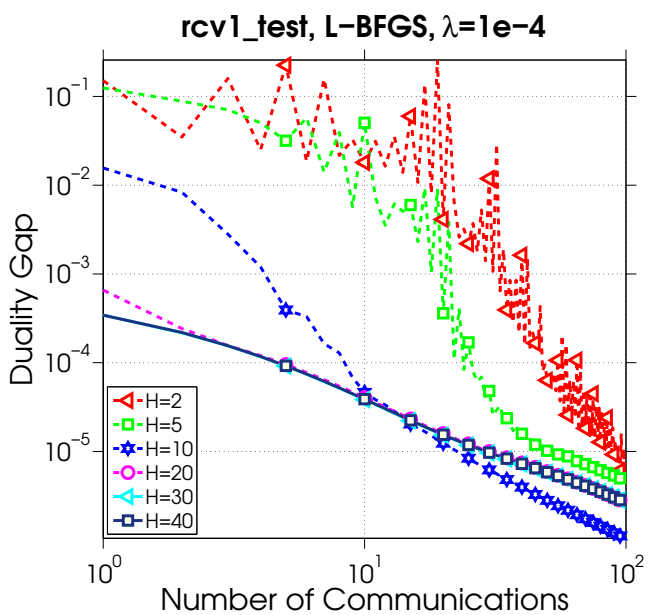


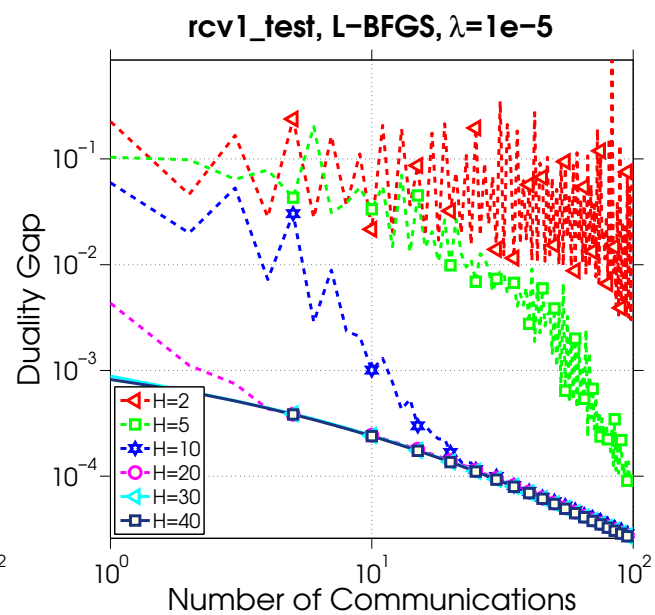




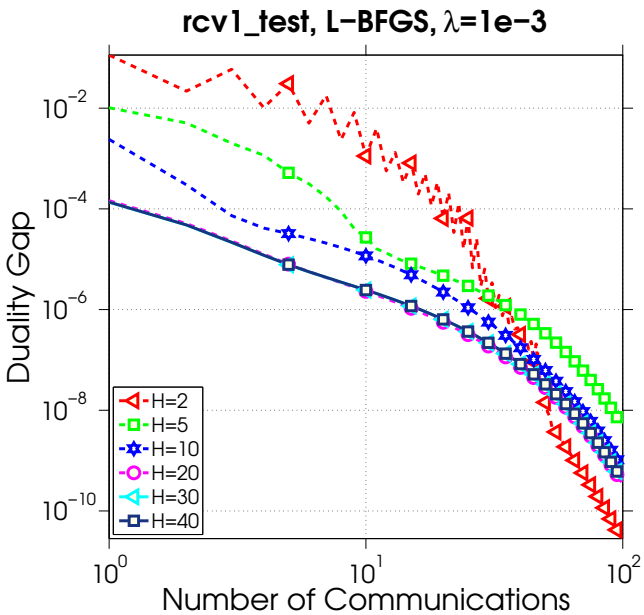
CoCoA$^+$: L-BFGS

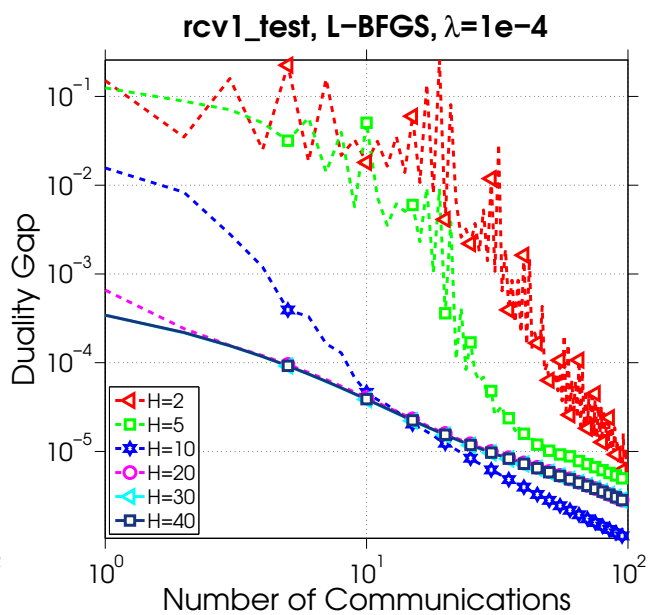

rcv1_test, L−BFGS, λ=1e−3
Duality Gap
Number of Communications
H=2
H=5
H=10
H=20
H=30
H=40

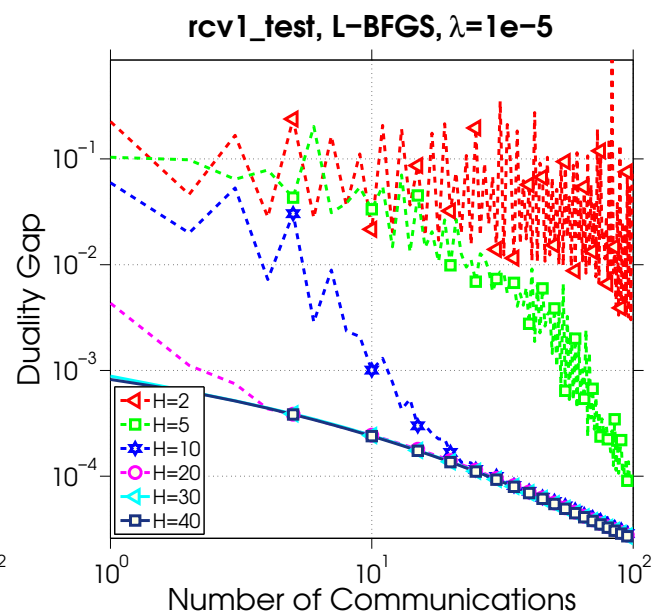

rcv1_test, L−BFGS, λ=1e−4
Duality Gap
Number of Communications
H=2
H=5
H=10
H=20
H=30
H=40

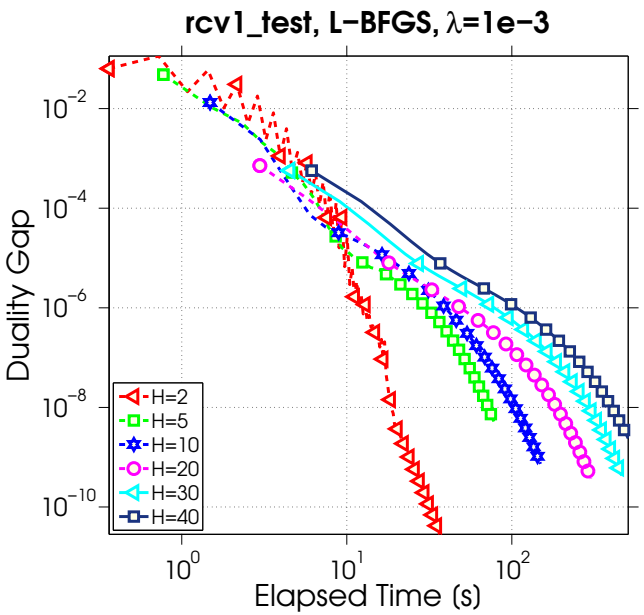

rcv1_test, L−BFGS, λ=1e−5
Duality Gap
Number of Communications
H=2
H=5
H=10
H=20
H=30
H=40

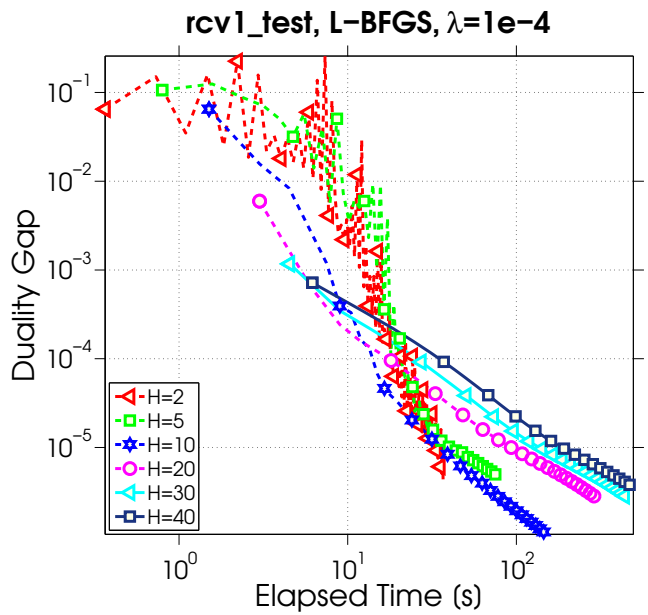
CoCoA$^+$: L-BFGS

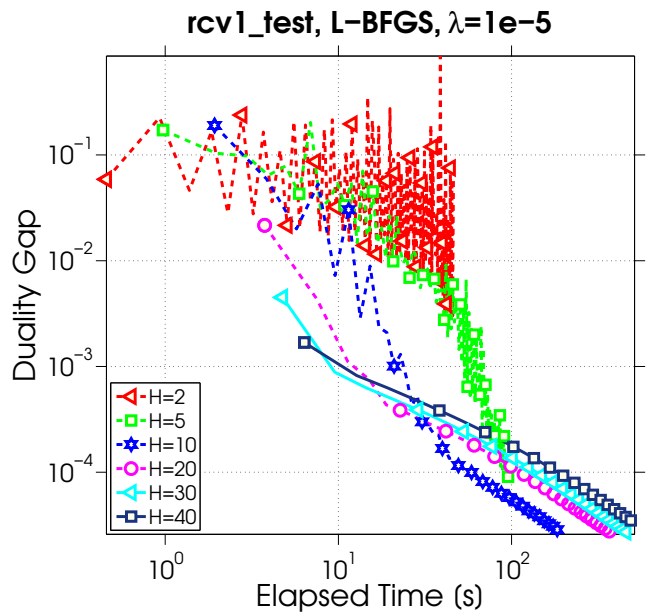

rcv1_test, L−BFGS, λ=1e−3
Duality Gap
Number of Communications
H=2
H=5
H=10
H=20
H=30
H=40


rcv1_test, L−BFGS, λ=1e−4
Duality Gap
Number of Communications
H=2
H=5
H=10
H=20
H=30
H=40

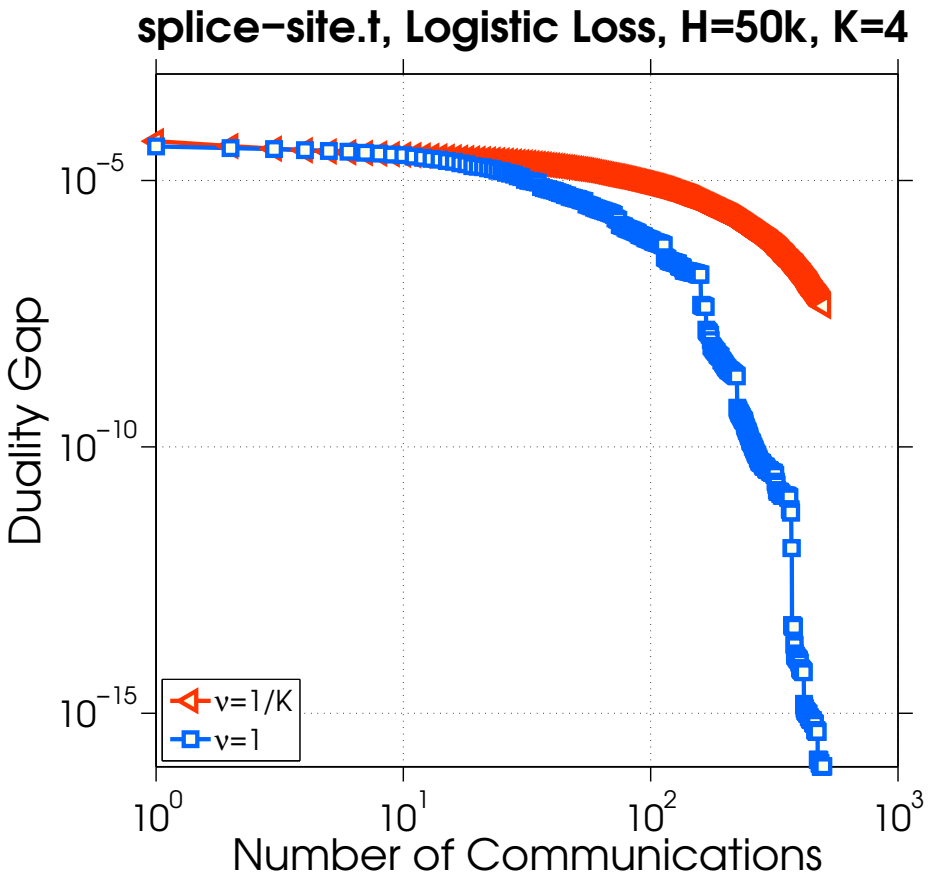

rcv1_test, L−BFGS, λ=1e−5
Duality Gap
Number of Communications
H=2
H=5
H=10
H=20
H=30
H=40

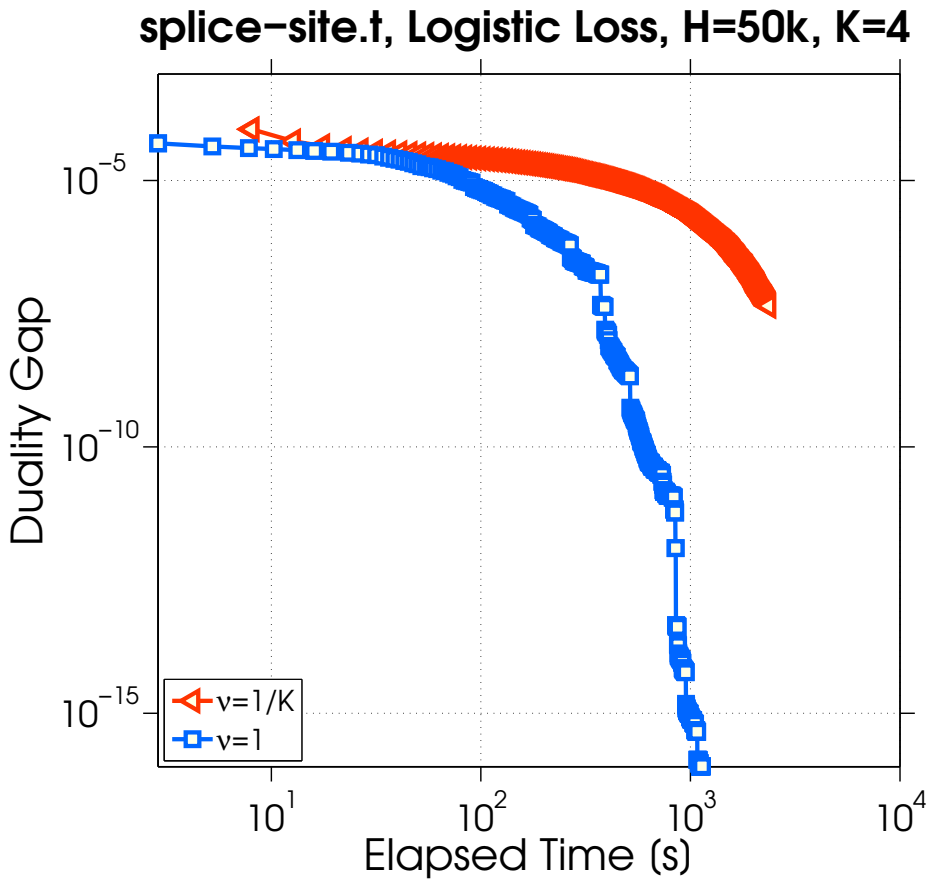

rcv1_test, L−BFGS, λ=1e−3
Duality Gap
Elapsed Time (s)
H=2
H=5
H=10
H=20
H=30
H=40


rcv1_test, L−BFGS, λ=1e−4
Duality Gap
Elapsed Time (s)
H=2
H=5
H=10
H=20
H=30
H=40

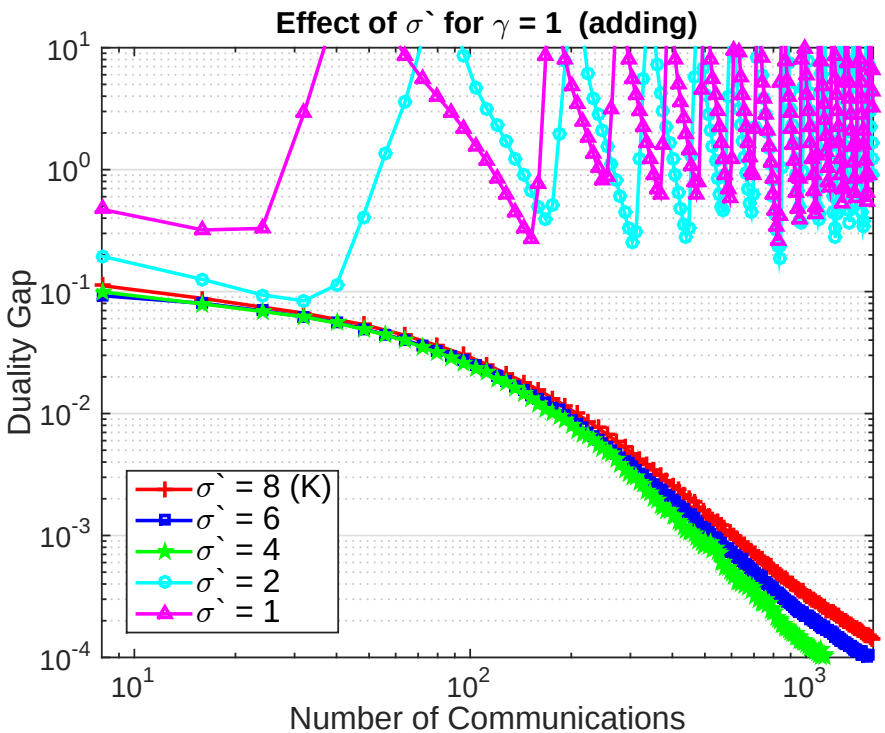

rcv1_test, L−BFGS, λ=1e−5
Duality Gap
Elapsed Time (s)
H=2
H=5
H=10
H=20
H=30
H=40

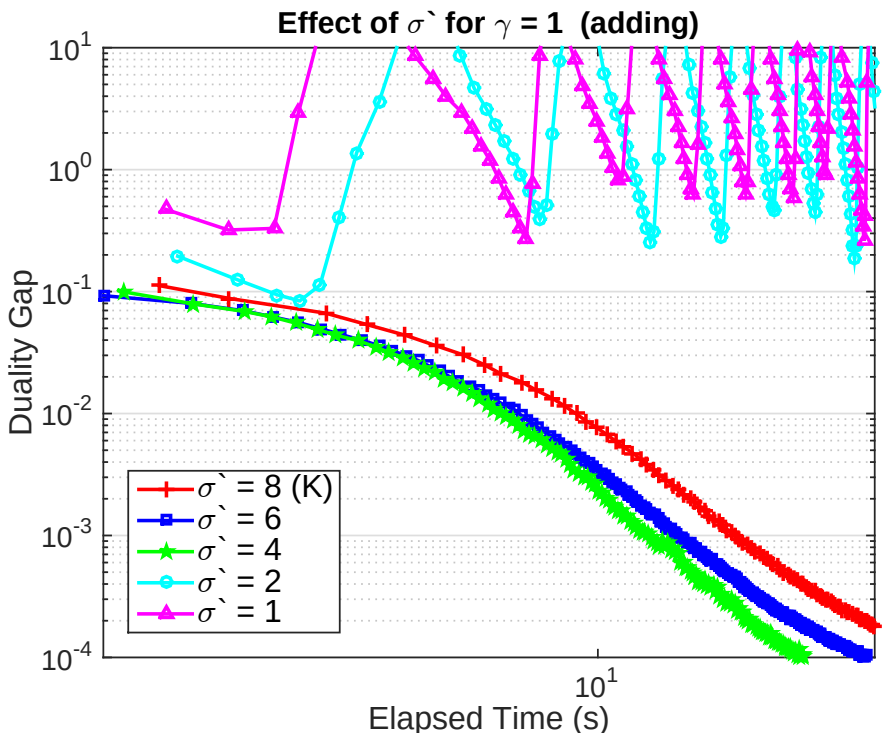
CoCoA$^+$ with RCDM - Large Scale Problem



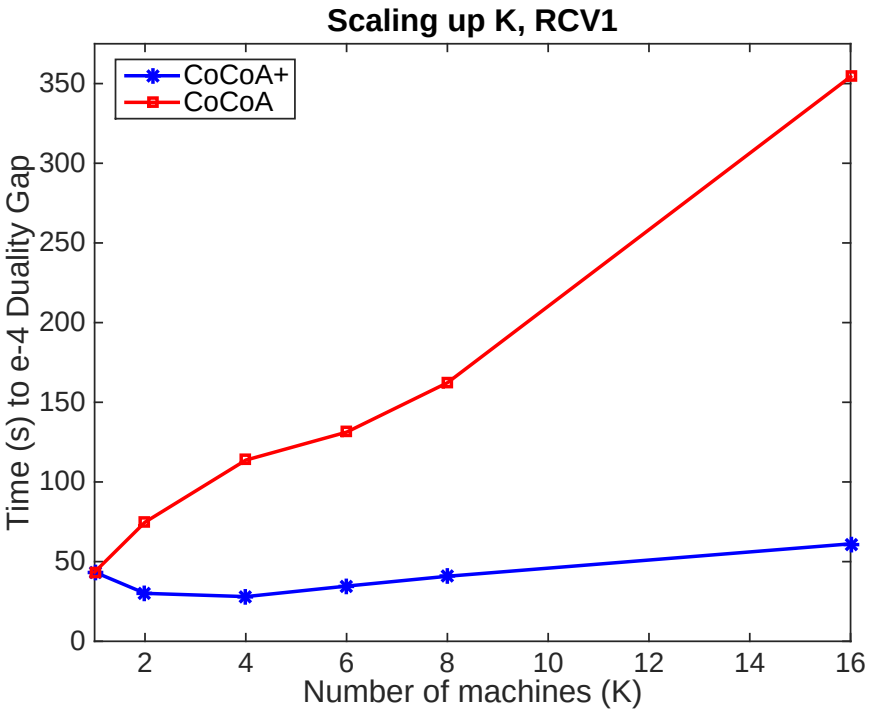


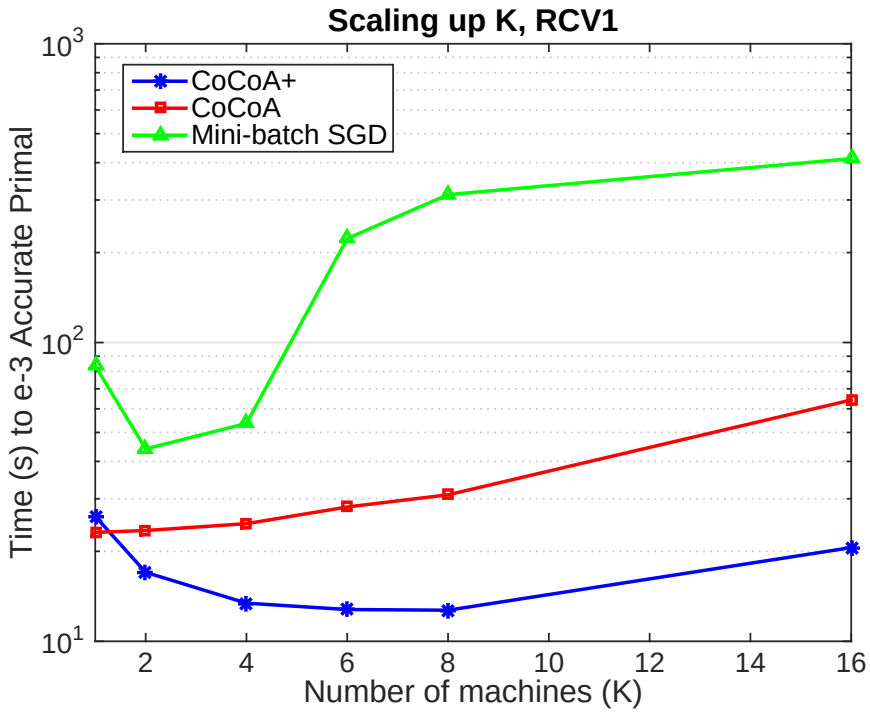
Effect of σ'

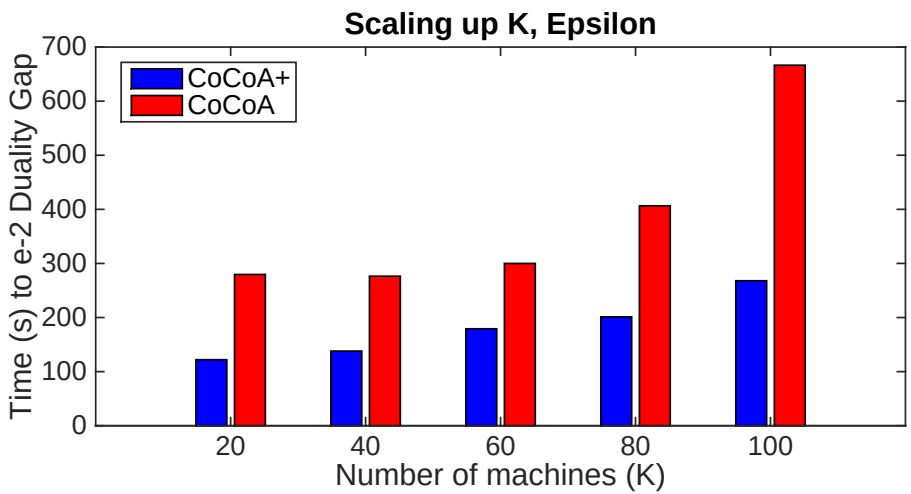




Scaling up


Scaling up K, RCV1
CoCoA+
CoCoA
Time (s) to e-4 Duality Gap
Number of machines (K)
0
50
100
150
200
250
300
350
2
4
6
8
10
12
14
16


Scaling up K, RCV1
CoCoA+
CoCoA
Mini-batch SGD
Time (s) to e-3 Accurate Primal
Number of machines (K)
10^3
10^2
10^1
2
4
6
8
10
12
14
16


Scaling up K, Epsilon
CoCoA+
CoCoA
Time (s) to e-2 Duality Gap
Number of machines (K)
0
100
200
300
400
500
600
700
20
40
60
80
100

How to do it Better?

Props

- the subproblem are very similar to original problems $\Rightarrow$ hope that optimized solvers will work still fine
- we can use duality to write the primal to local dual problem (we can also use optimized primal solvers – SAGA, SGD, SVRG, ...)

How to do it Better?

Props

- the subproblem are very similar to original problems $\Rightarrow$ hope that optimized solvers will work still fine
- we can use duality to write the primal to local dual problem (we can also use optimized primal solvers – SAGA, SGD, SVRG, ...)

Cons

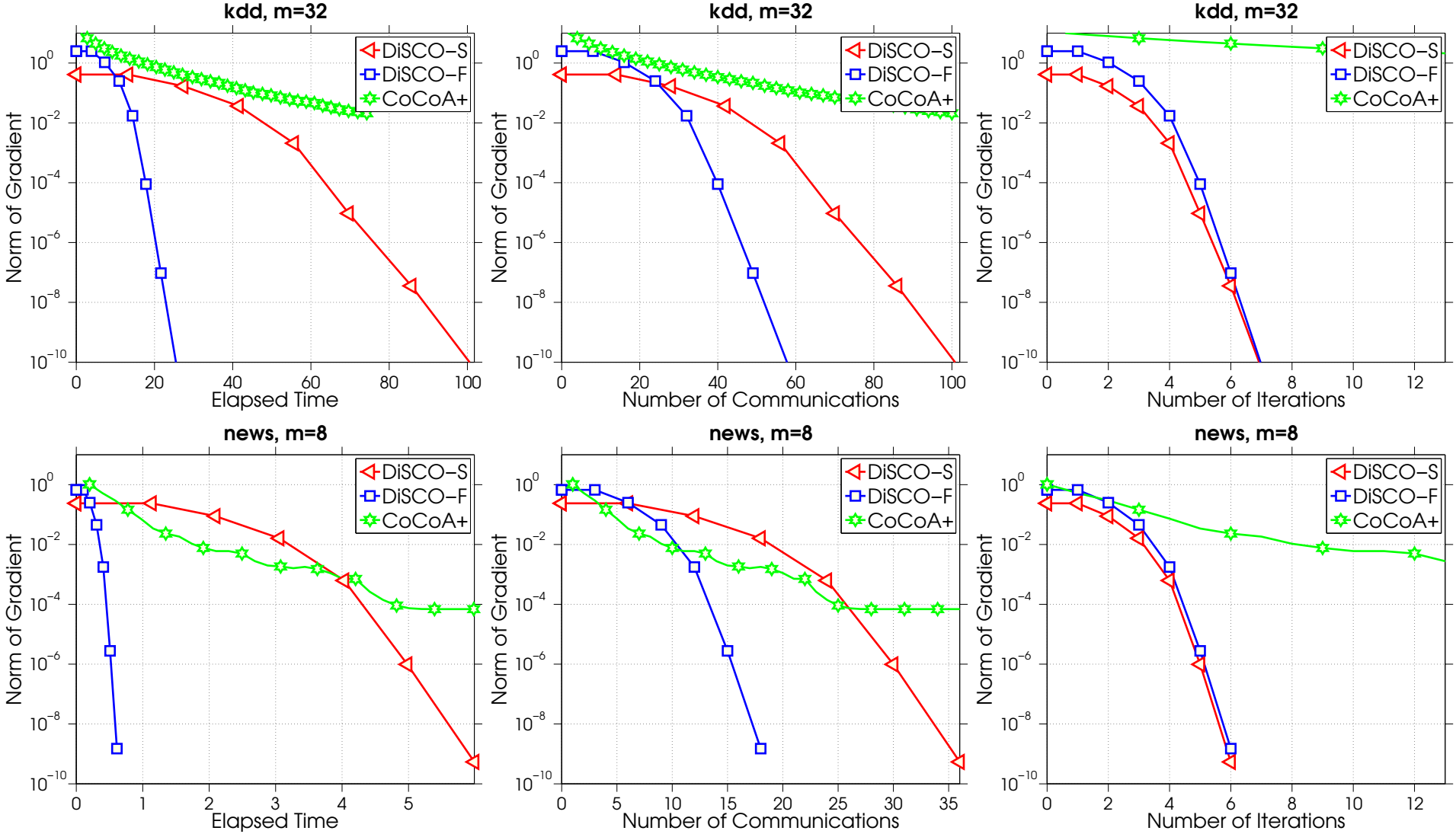
- **Theory** shows no advantage when compared with steepest descent
- ...this is just theory, in practise it works much better ...(actually, this can be seen as some **fully parallel inexact** block coordinate descent **but with primal-dual analysis**
- the main reason is that the algorithm is using **ONLY** the block diagonal part of Hessian

Distributed PCG

Partitioning by samples was done in Yuchen Zhang and Lin Xiao: *Communication-efficient distributed optimization of self-concordant empirical loss*, arXiv:1501.00263, 2015.

We redesign the algorithm to utilize cores better and minimize communication

			part. by samp.	**part. by feat.**
comp.	master	matrix-vector multiplication	$1(\mathbf{R}^{d\times d} \times \mathbf{R}^{d})$	$1(\mathbf{R}^{d_1\times d_1} \times \mathbf{R}^{d_1})$
		back solving linear system	$1\ (\mathbf{R}^{d})$	$1\ (\mathbf{R}^{d_1})$
		sum of vectors	$4\ (\mathbf{R}^{d})$	$4\ (\mathbf{R}^{d_1})$
		inner product of vectors	$4\ (\mathbf{R}^{d})$	$4\ (\mathbf{R}^{d_1})$
	nodes	matrix-vector multiplication	$1\ (\mathbf{R}^{d\times d} \times \mathbf{R}^{d})$	$1(\mathbf{R}^{d_1\times d_i} \times \mathbf{R}^{d_i})$
		back solving linear system	0	$1\ (\mathbf{R}^{d_i})$
		sum of vectors	0	$4\ (\mathbf{R}^{d_i})$
		inner product of vectors	0	$4\ (\mathbf{R}^{d_i})$
comn.	Broadcast		one $\mathbf{R}^{d}$ vector	0
	ReduceAll		1x $\mathbf{R}^{d}$	1x $\mathbf{R}^{n}$, 3 scalars



- we need the same number of iterations (the "same algorithm")
- we save almost 50% of communication
- we are much faster (we utilize nodes better)
- **future work:** searching for better preconditioning; distributed L-BFGS implementation

Joke at the end – what have I learnt in US so far

"How shall I torture you today?
Put you on the rack? Boil you in oil?
Make you call a technical support line?"

References

1. Chenxin Ma and Martin Takáč: *Partitioning Data on Features or Samples in Communication-Efficient Distributed Optimization?*, OptML@NIPS 2015.
2. Chenxin Ma, Virginia Smith, Martin Jaggi, Michael I. Jordan, Peter Richtárik and Martin Takáč: *Adding vs. Averaging in Distributed Primal-Dual Optimization*, ICML 2015.
3. Martin Jaggi, Virginia Smith, Martin Takáč, Jonathan Terhorst, Thomas Hofmann and Michael I. Jordan: *Communication-Efficient Distributed Dual Coordinate Ascent*, NIPS 2014.
4. Richtárik, P. and Takáč, M.: *Distributed coordinate descent method for learning with big data*, Journal Paper Journal of Machine Learning Research (to appear), 2016
5. Richtárik, P. and Takáč, M.: *On optimal probabilities in stochastic coordinate descent methods*, Optimization Letters, 2015.
6. Richtárik, P. and Takáč, M.: *Parallel coordinate descent methods for big data optimization*, Mathematical Programming, 2015.
7. Richtárik, P. and Takáč, M.: *Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function*, Mathematical Programming, 2012.
8. Takáč, M., Bijral, A., Richtárik, P. and Srebro, N.: *Mini-batch primal and dual methods for SVMs*, In ICML, 2013.
9. Qu, Z., Richtárik, P. and Zhang, T.: *Randomized dual coordinate ascent with arbitrary sampling*, arXiv:1411.5873, 2014.
10. Qu, Z., Richtárik, P., Takáč, M. and Fercoq, O.: *SDNA: Stochastic Dual Newton Ascent for Empirical Risk Minimization*, arXiv:1502.02268, 2015.
11. Tappenden, R., Takáč, M. and Richtárik, P., *On the Complexity of Parallel Coordinate Descent*, arXiv: 1503.03033, 2015.