CoCoA$^+$: Adding vs. Averaging in Distributed Optimization

Martin Takáč

joint work with

- Chenxin Ma
- Virginia Smith
- Martin Jaggi
- Michael I. Jordan
- Peter Richtrik

FMFI, UK, 25th of June, 2015

Outline

- Problem Formulation
- Serial and Parallel Coordinate Descent Method (CDM)
- Distributed CDM
- Original CoCoA Framework
- CoCoA$^+$ Framework
- Computation vs. Communication Trade-off
- Spark
- Some Numerical Experiments

The Problem - Regularized Empirical Loss Minimization

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

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

where

- $\lambda > 0$ 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 data, $x_i \in \mathbf{R}^d$ and $y_i \in \mathbf{R}$.

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

where

- $\lambda > 0$ 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

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

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

Gap function

$$G(\alpha) = P(w(\alpha)) - D(\alpha)$$

The Setting & Challenges

- The size of matrix A is huge (e.g. TBs of data)
- We want to use many nodes of computer cluster (or cloud) to speed-up the computation

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)

The Serial/Parallel/Distributed SDCA Algorithm

Serial **S**tochastic **D**ual **C**oordinate **A**scent

choose $\alpha^{(0)} \in \mathbf{R}^n$

repeat

$\alpha^{(t+1)} = \alpha^{(t)}$

pick a random coordinate $i \in \{1, \ldots, n\}$

compute the update: $h_t^i(\alpha^{(t)}) := \arg\max_h D(\alpha^{(t)} + he_i)$

apply the update: $\alpha_i^{(t+1)} = \alpha_i^{(t+1)} + h_t^i(\alpha^{(t)})e_i$

The Serial/Parallel/Distributed SDCA Algorithm

Parallel Stochastic Dual Coordinate Ascent

choose $\alpha^{(0)} \in \mathbf{R}^n$

repeat

$\alpha^{(t+1)} = \alpha^{(t)}$

pick a random coordinate $i \in \{1, \ldots, n\}$

pick a random subset $S \subset \{1, \ldots, n\}$ with $|S| = H$

for each $i \in S$ in **parallel** do

compute the update: $h_t^i(\alpha^{(t)}) := \arg\max_h D(\alpha^{(t)} + he_i)$

apply the update: $\alpha_i^{(t+1)} = \alpha_i^{(t+1)} + h_t^i(\alpha^{(t)})e_i$

apply the update: $\alpha_i^{(t+1)} = \alpha_i^{(t+1)} + \frac{1}{H} \sum_{i \in S} h_t^i(\alpha^{(t)})e_i$

The Serial/Parallel/Distributed SDCA Algorithm

- assume we have K nodes (computers) each with parallel processing power
- we **partition** the coordinates $\{1, 2, \ldots, n\}$ into K **balanced** sets $\mathcal{P}_1, \ldots, \mathcal{P}_K$ $\forall k \in \{1, \ldots, K\}$ we have $|\mathcal{P}_k| = \frac{n}{K}$

Distributed Stochastic Dual Coordinate Ascent

choose $\alpha^{(0)} \in \mathbf{R}^n$

repeat

$\alpha^{(t+1)} = \alpha^{(t)}$

for each **computer** $k \in \{1, \ldots, K\}$ in **parallel** do

pick a random subset $S \subset \{1, \ldots, n\}$ with $|S| = H$

pick a random subset $S_k \subset \mathcal{P}_k$ with $|S| = H \leq \frac{n}{K}$

for each $i \in S_k$ in **parallel** do

compute the update: $h_t^i(\alpha^{(t)}) := \arg\max_h D(\alpha^{(t)} + he_i)$

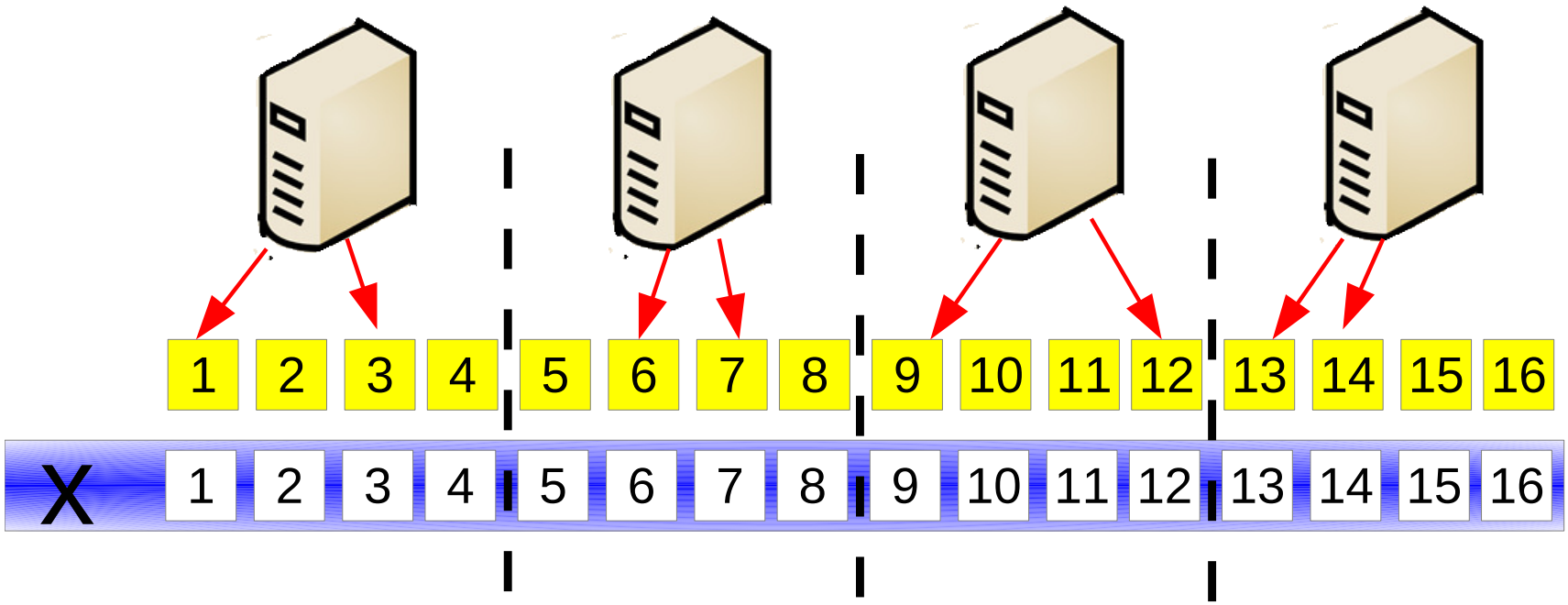
apply the update: $\alpha_i^{(t+1)} = \alpha_i^{(t+1)} + \frac{1}{H} \sum_{i \in S} h_t^i(\alpha^{(t)}) e_i$

apply the update: $\alpha_i^{(t+1)} = \alpha_i^{(t+1)} + \frac{1}{KH} \sum_{k \in \{1,\ldots,K\}} \sum_{i \in S_k} h_t^i(\alpha^{(t)}) e_i$

The distributed algorithm can need (in the worst case) the **same number of iterations as a serial one!**

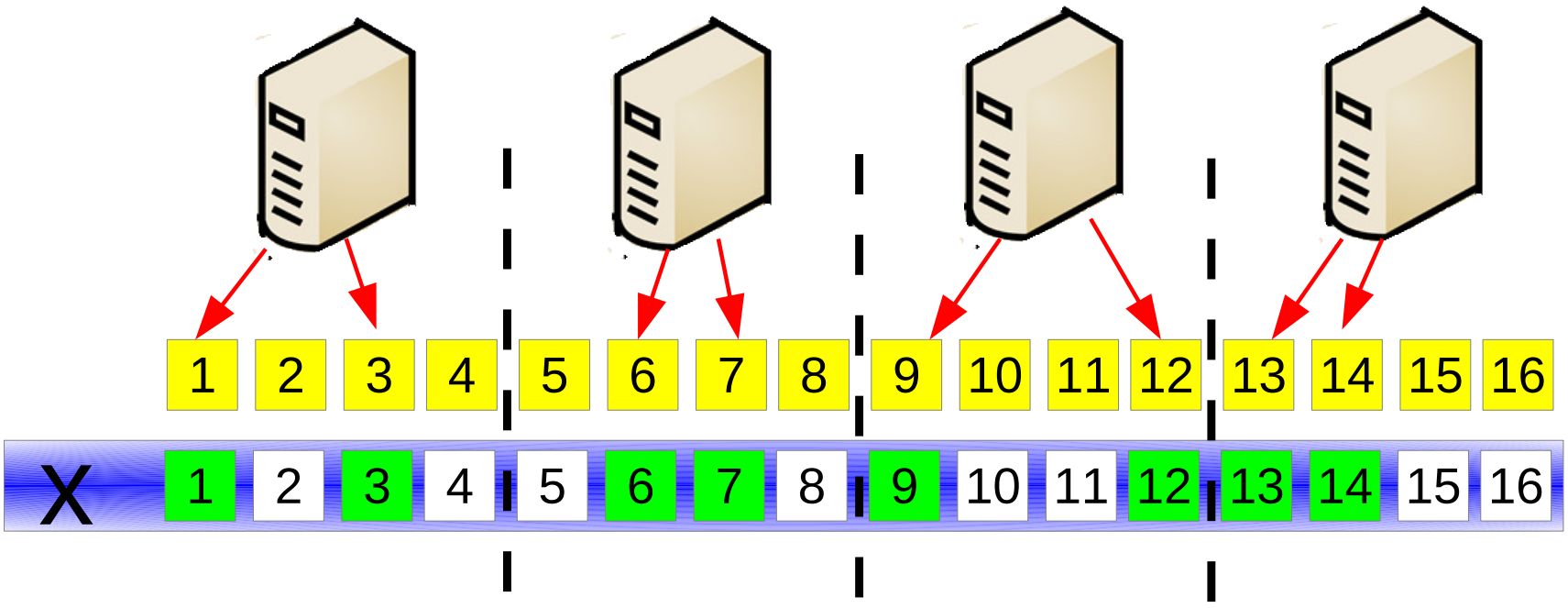
Distributed CDM

Illustration: $K = 4$ and $H = 2$



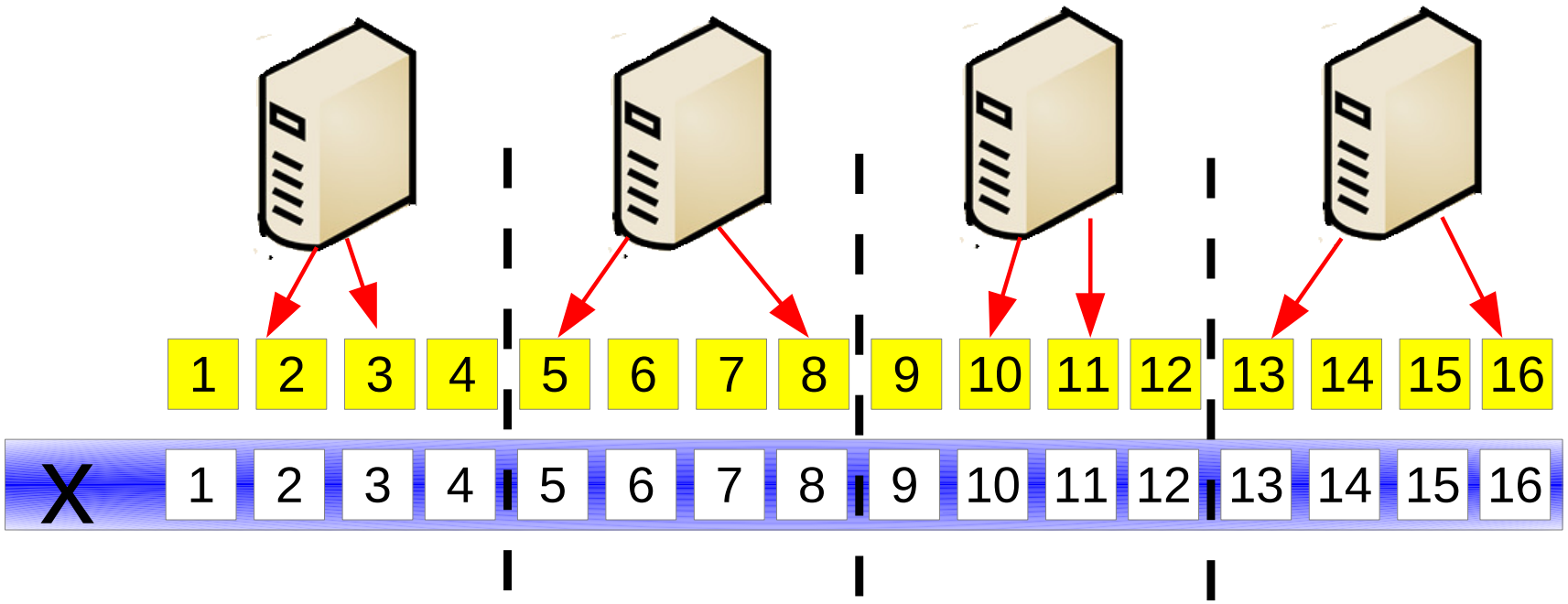
Distributed CDM

Illustration: $K = 4$ and $H = 2$



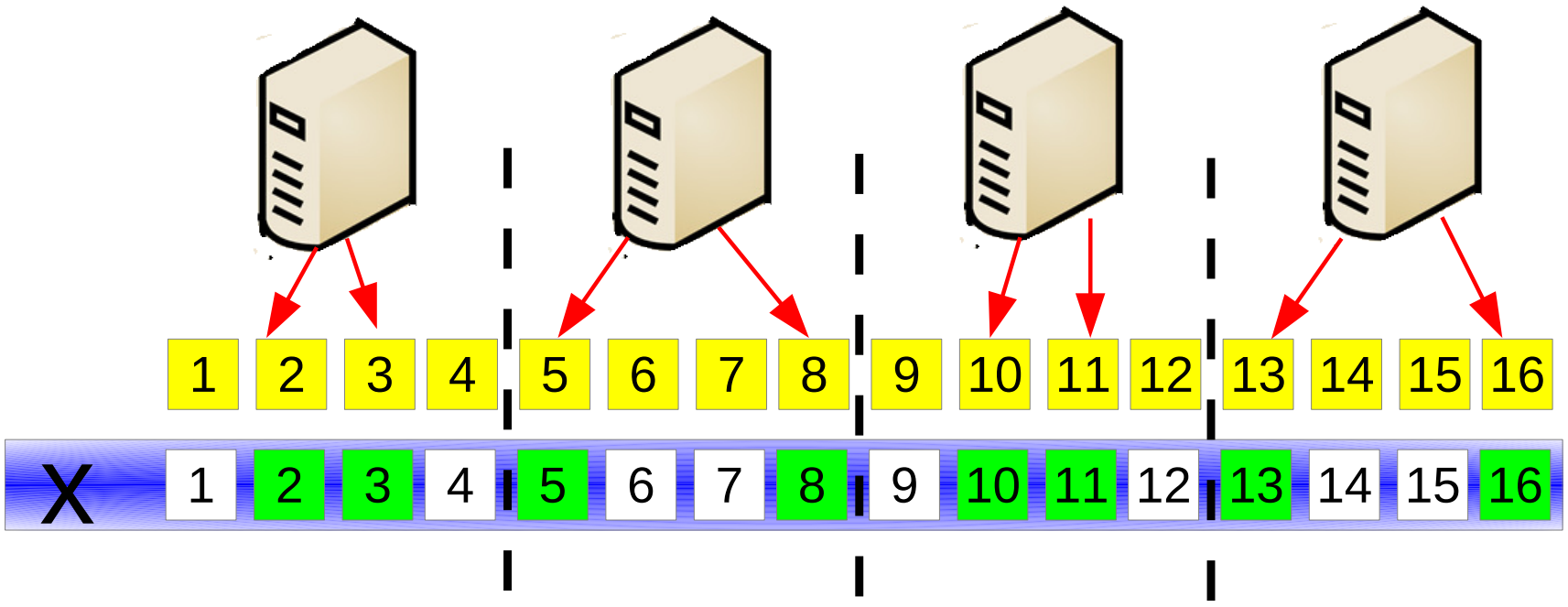
Distributed CDM

Illustration: $K = 4$ and $H = 2$



Distributed CDM

Illustration: $K = 4$ and $H = 2$

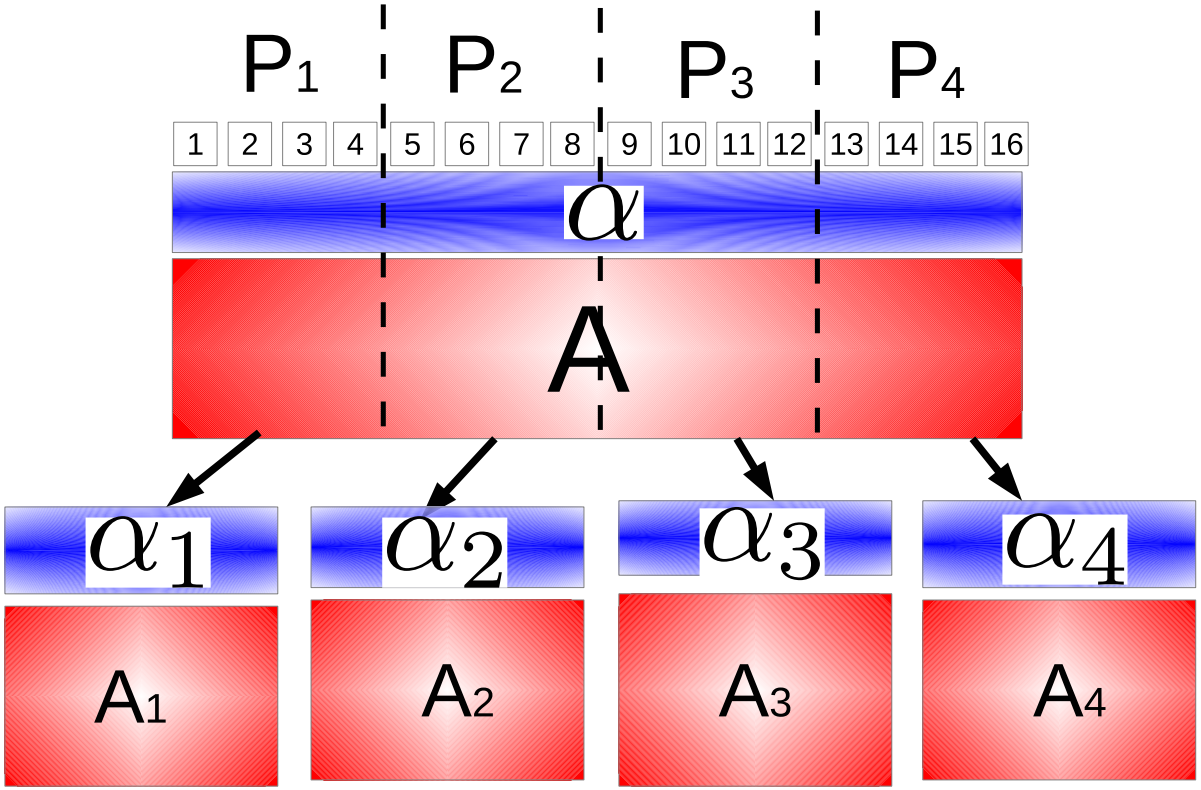


Disadvantages of Distributed CDM

- we cannot choose $H > |\mathcal{P}_k|$!
- the computation of step is very easy (usually close form or a bit complicated 1D problem)
- after taking H steps, usually the objective function doesn't change much
- it is almost **impossible** to balance computation and communication

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 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)?
- Can we prove it will be better?

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

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

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

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

Lemma

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

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

- 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

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

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

we have the expected duality gap

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

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

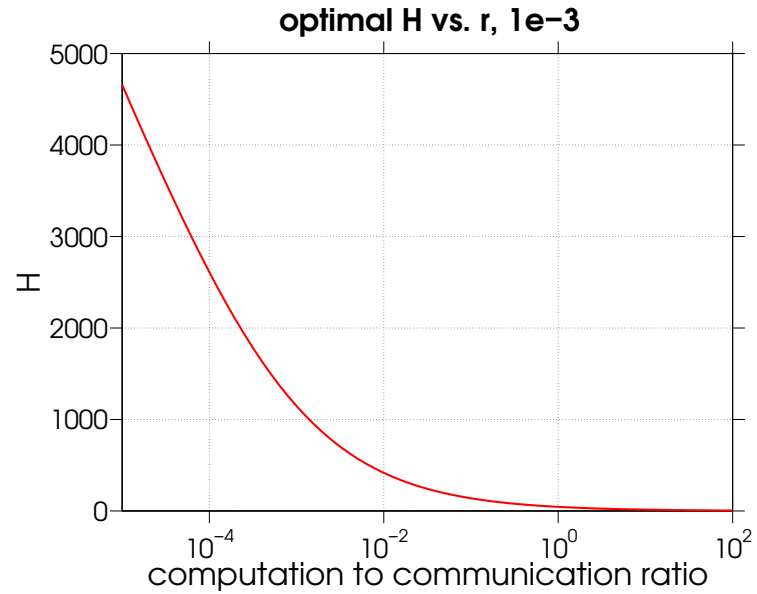
Let

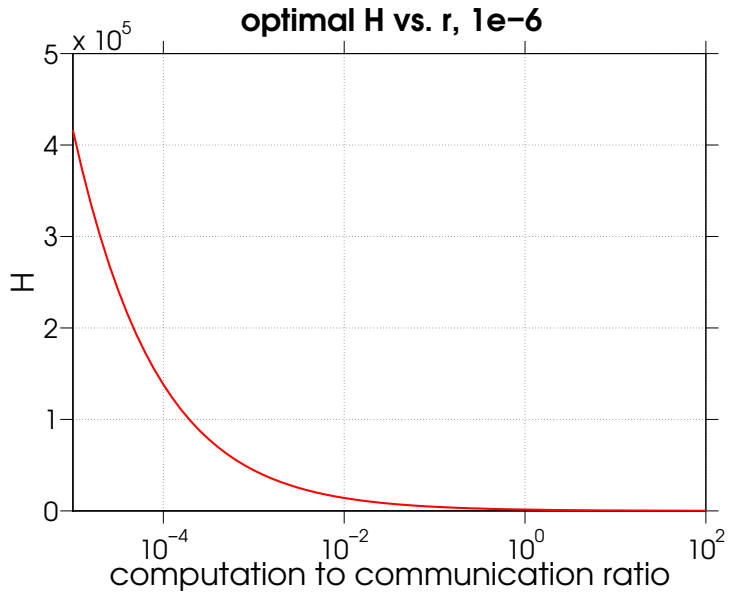
- τ_o be the duration of communication per iteration
- τ_c be the duration of **ONE** coordinate update during the inner iteration

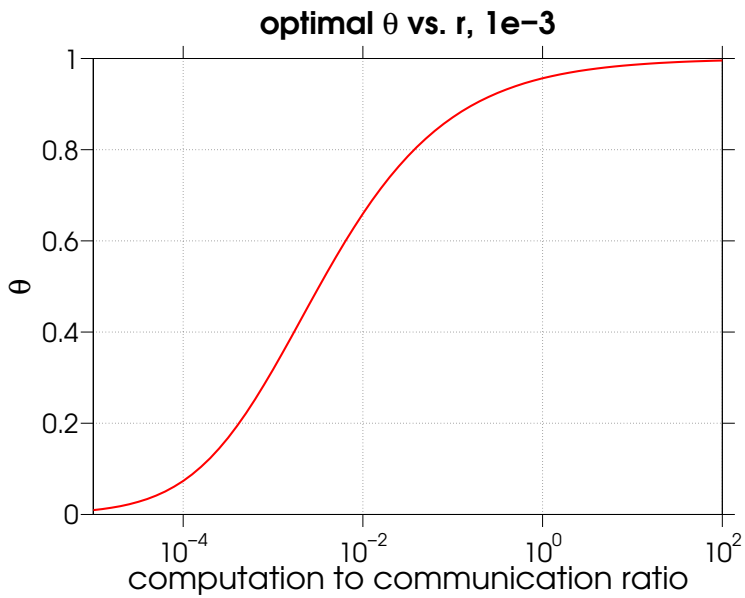
Total runtime

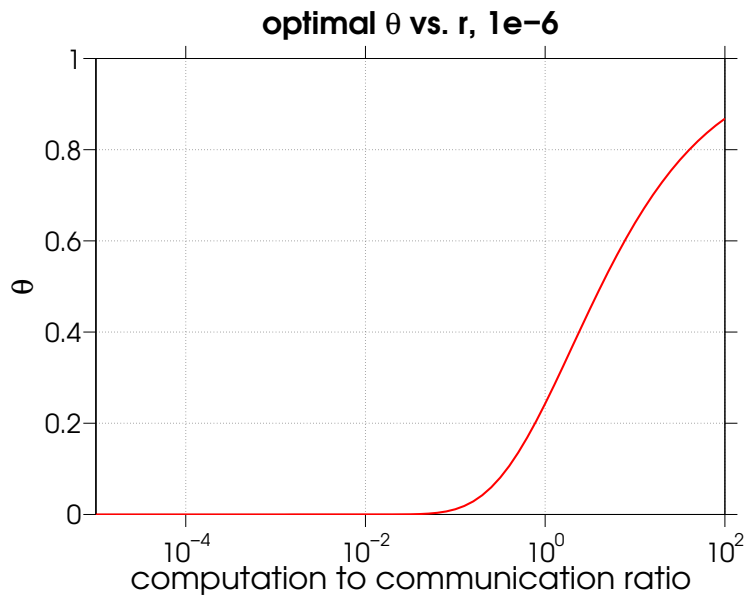
$$\mathcal{O}\left(\frac{1}{1-\Theta}\left(\tau_O + H\tau_c\right)\right) = \mathcal{O}\left(\frac{1}{1-\Theta}\left(1 + H\underbrace{\frac{\tau_c}{\tau_o}}_{r_{c/o}}\right)\right)$$

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









Apache Spark

Apache Spark

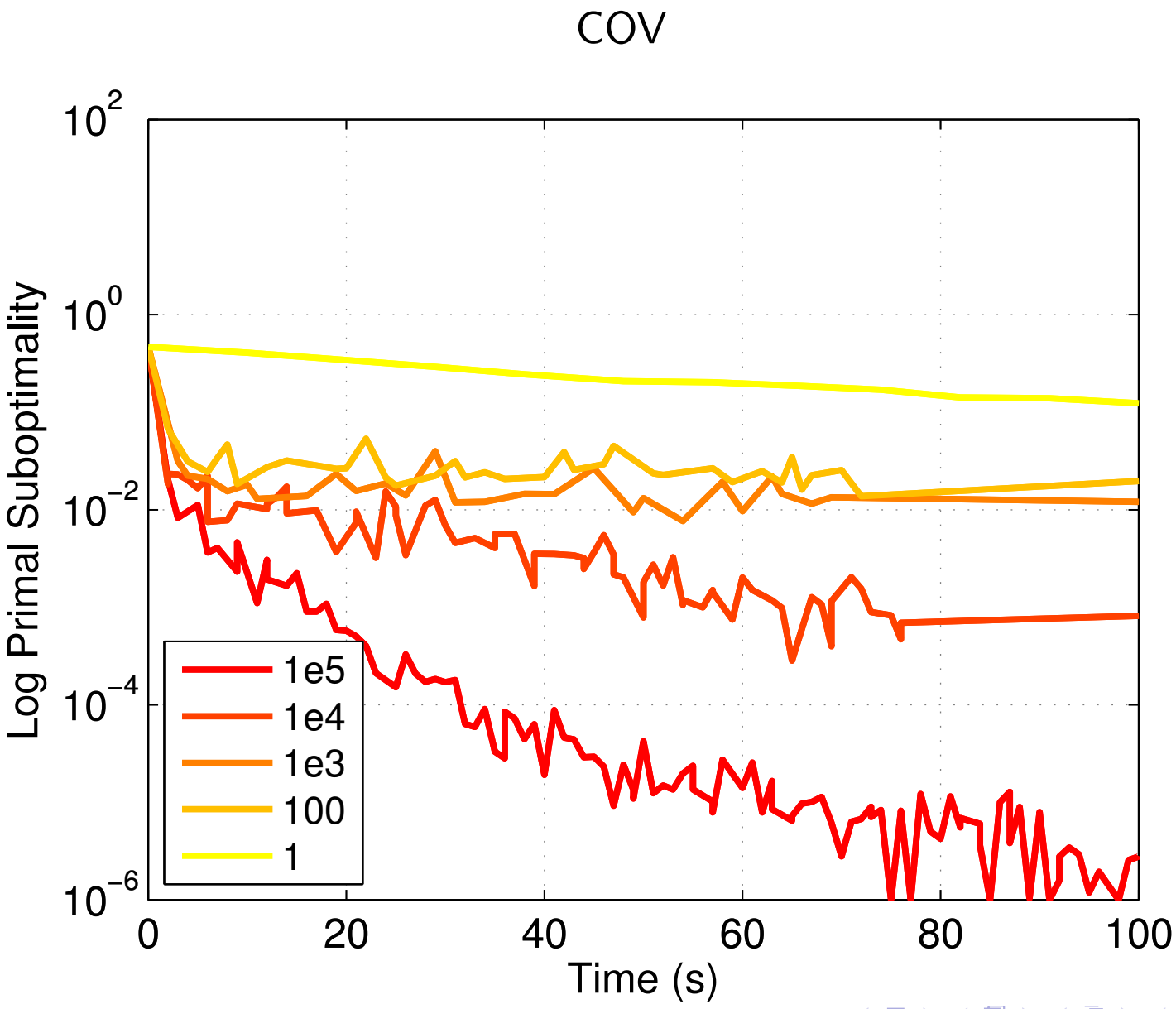
- is a fast and general engine for large-scale data processing
- runs on Hadoop, Mesos, standalone, or in the cloud. It can access diverse data sources including HDFS, Cassandra, HBase, S3
- **is slower than our C++ code** for CoCoA$^+$
- we run it on Amazon Elastic Compute Cloud (Amazon EC2)

Numerical Experiments

Datasets

Dataset	**Training** (n)	**Features** (d)	$nnz/(n \cdot d)$	λ	**Workers** (K)
cov	522,911	54	22.22%	1*e*-6	4
rcv1	677,399	47,236	0.16%	1*e*-6	8
imagenet	32,751	160,000	100%	1*e*-5	32

Dependence of Primal Suboptimality on H



Comparison with Different Algorithms

- COCOA

Comparison with Different Algorithms

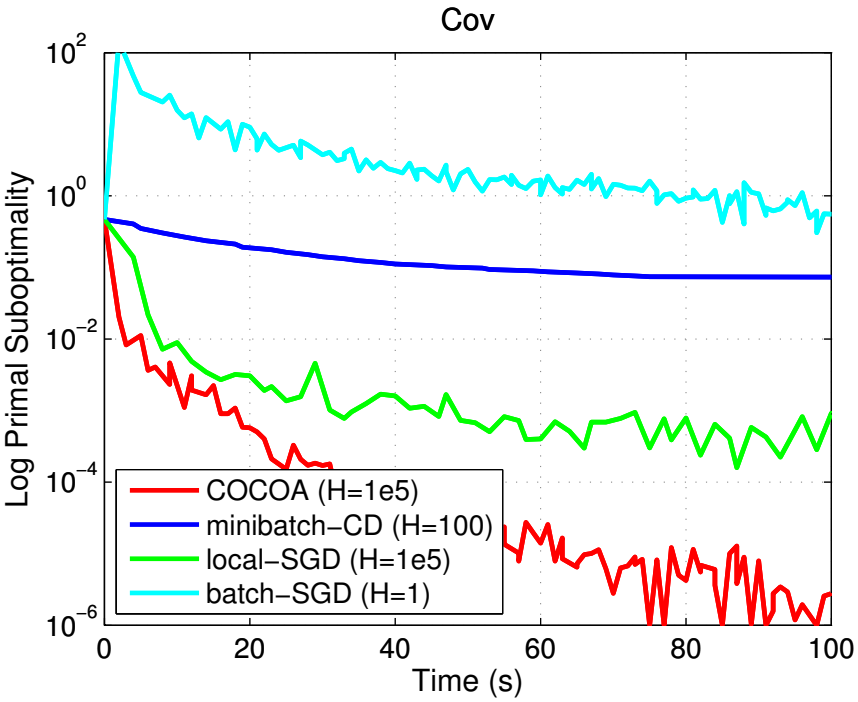
- COCOA
- minibatch-CD
  Tianbao Yang. *Trading Computation for Communication: Distributed Stochastic Dual Coordinate Ascent.* In NIPS 2013.
  Martin Takáč, Avleen Bijral, Peter Richtárik, and Nathan Srebro. *Mini-Batch Primal and Dual Methods for SVMs.* In ICML, March 2013.

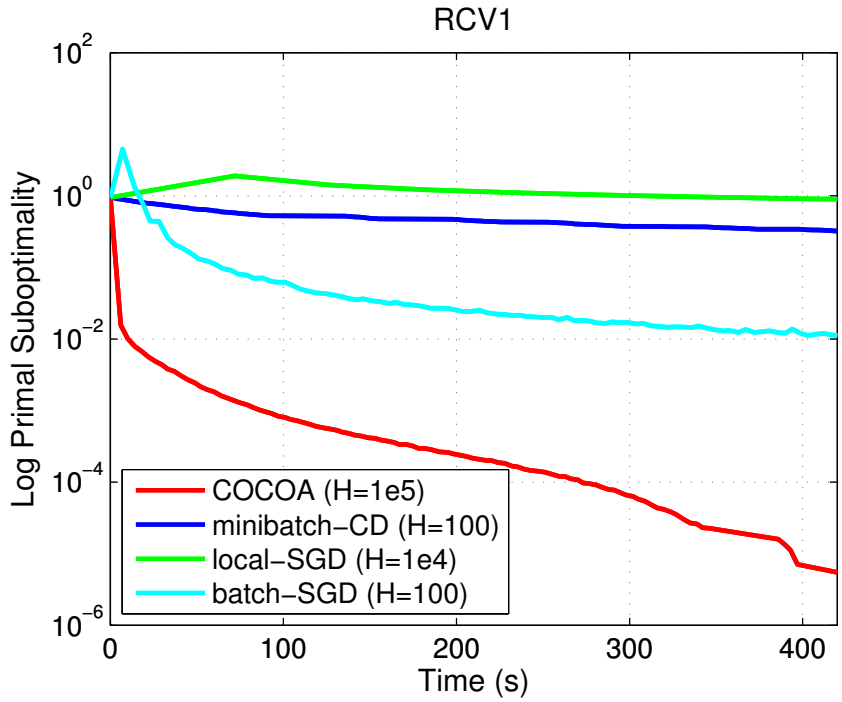
Comparison with Different Algorithms

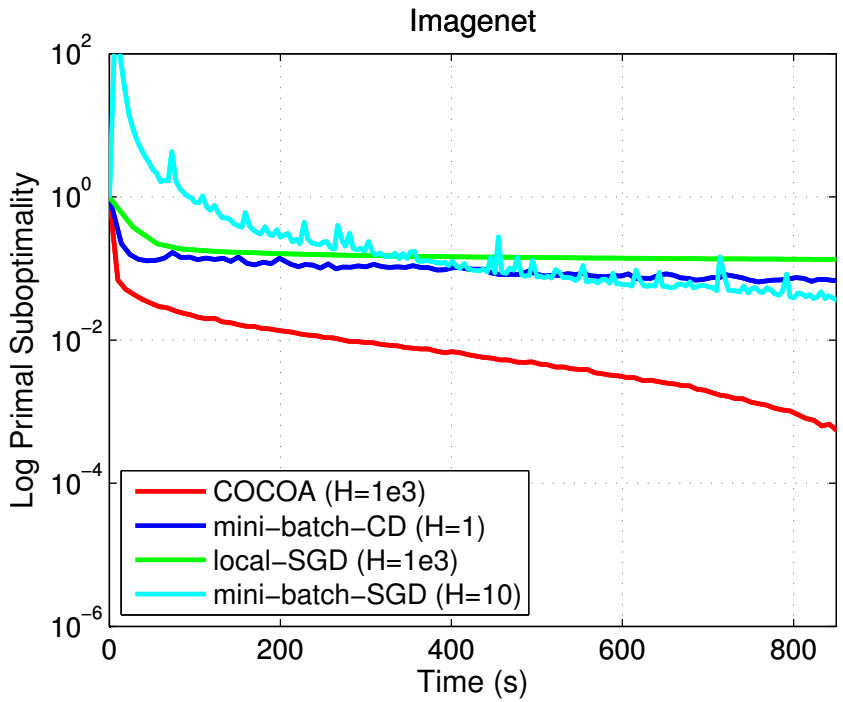
- COCOA
- minibatch-CD
  Tianbao Yang. *Trading Computation for Communication: Distributed Stochastic Dual Coordinate Ascent.* In NIPS 2013.
  Martin Takáč, Avleen Bijral, Peter Richtárik, and Nathan Srebro. *Mini-Batch Primal and Dual Methods for SVMs.* In ICML, March 2013.
- local-SGD
  Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. *Pegasos: Primal Estimated Sub-Gradient Solver for SVM.* Mathematical Programming, 127(1):3–30, October 2010.

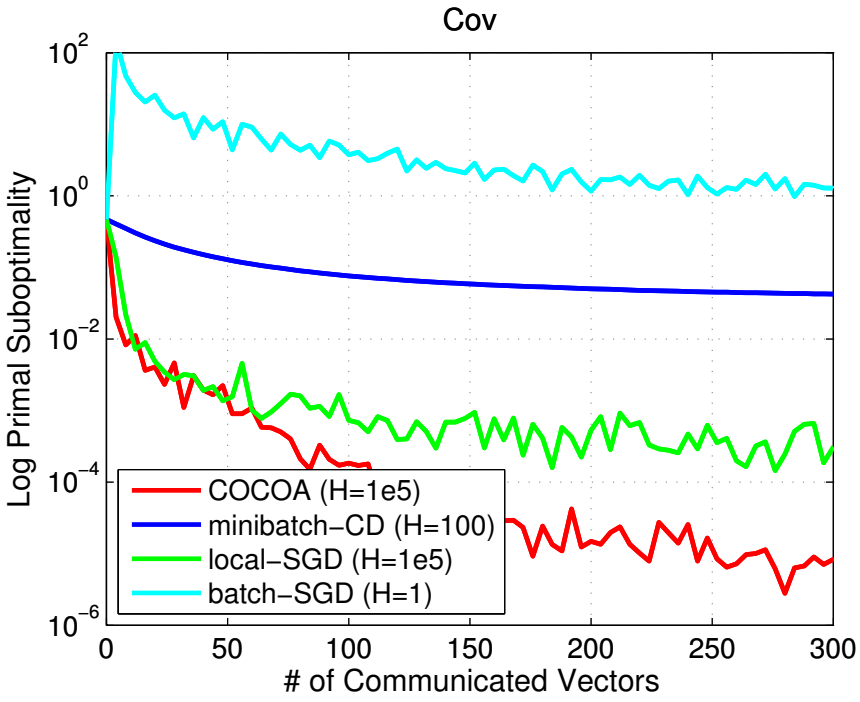
Comparison with Different Algorithms

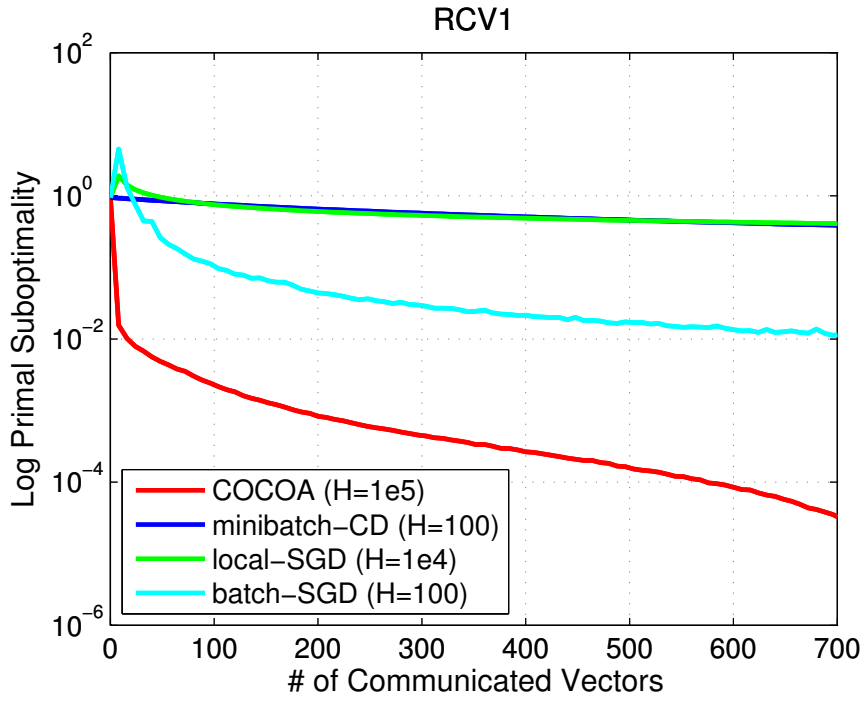
- COCOA
- minibatch-CD
  Tianbao Yang. *Trading Computation for Communication: Distributed Stochastic Dual Coordinate Ascent.* In NIPS 2013.
  Martin Takáč, Avleen Bijral, Peter Richtárik, and Nathan Srebro. *Mini-Batch Primal and Dual Methods for SVMs.* In ICML, March 2013.
- local-SGD
  Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. *Pegasos: Primal Estimated Sub-Gradient Solver for SVM.* Mathematical Programming, 127(1):3–30, October 2010.
- batch-SGD
  Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. *Pegasos: Primal Estimated Sub-Gradient Solver for SVM.* Mathematical Programming, 127(1):3–30, October 2010.

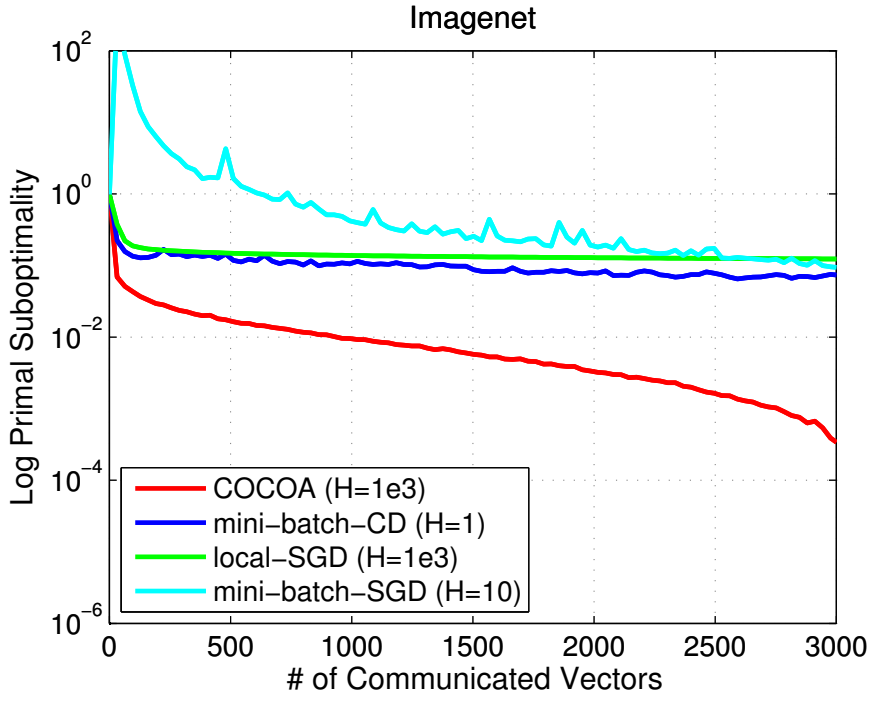

Cov
Log Primal Suboptimality
Time (s)
COCOA (H=1e5)
minibatch−CD (H=100)
local−SGD (H=1e5)
batch−SGD (H=1)


RCV1
Log Primal Suboptimality
Time (s)
COCOA (H=1e5)
minibatch−CD (H=100)
local−SGD (H=1e4)
batch−SGD (H=100)

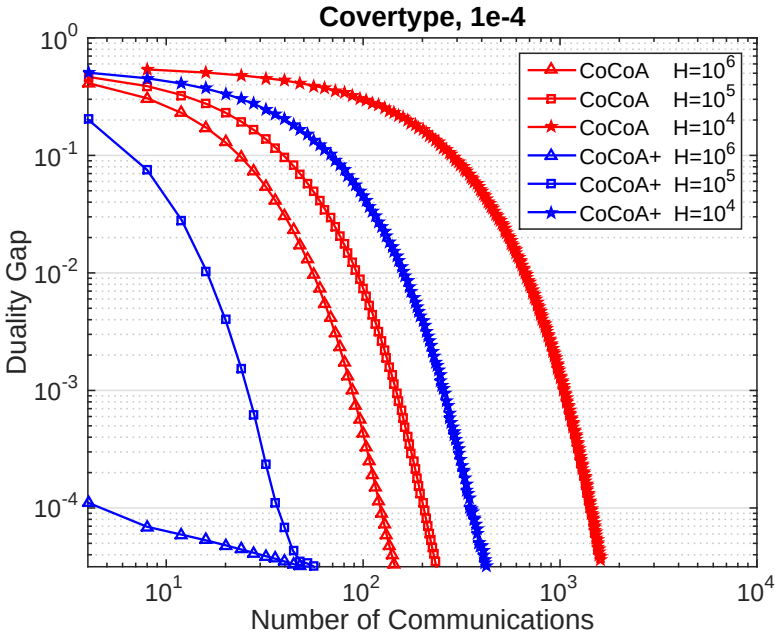

Imagenet
Log Primal Suboptimality
Time (s)
COCOA (H=1e3)
mini−batch−CD (H=1)
local−SGD (H=1e3)
mini−batch−SGD (H=10)

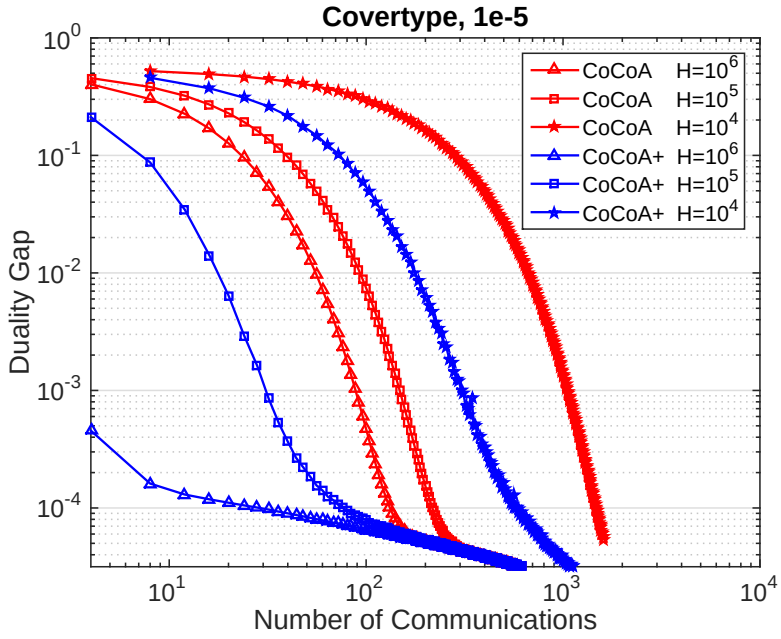

Cov
Log Primal Suboptimality
# of Communicated Vectors
COCOA (H=1e5)
minibatch−CD (H=100)
local−SGD (H=1e5)
batch−SGD (H=1)

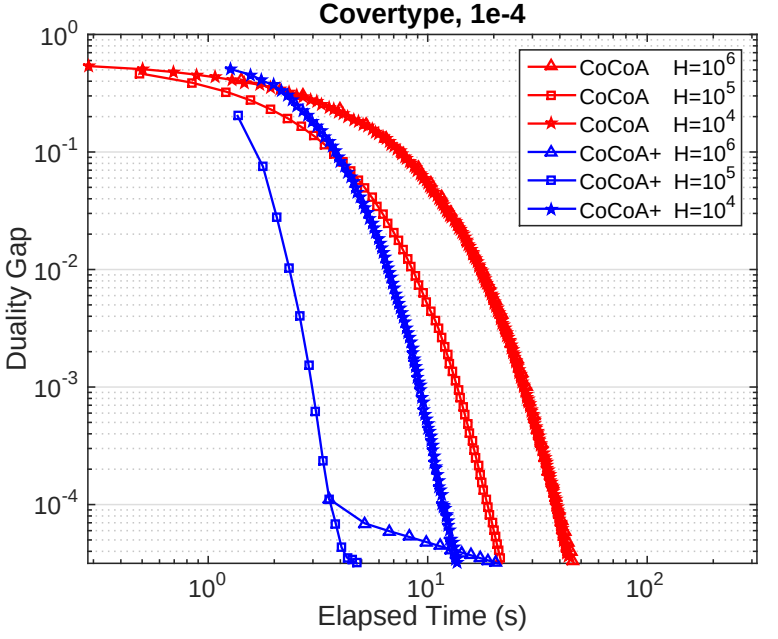

RCV1
Log Primal Suboptimality
# of Communicated Vectors
COCOA (H=1e5)
minibatch−CD (H=100)
local−SGD (H=1e4)
batch−SGD (H=100)

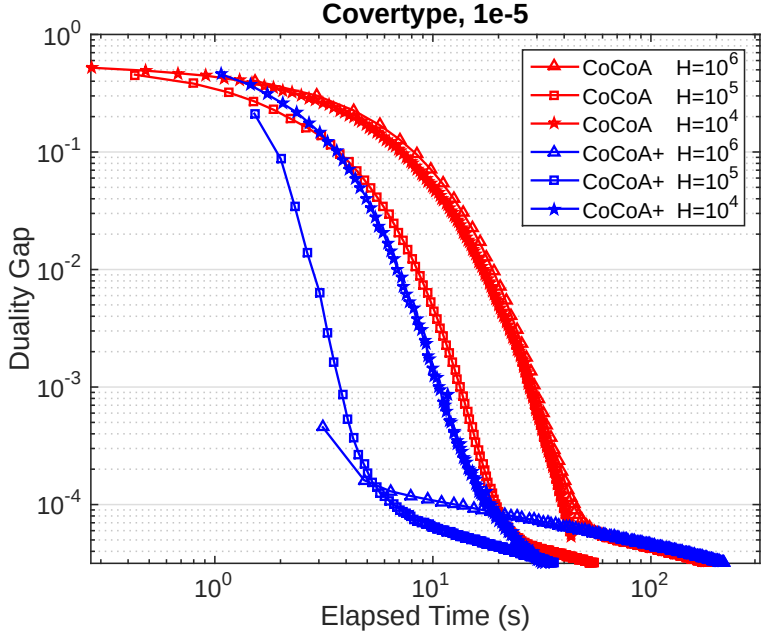

Imagenet
Log Primal Suboptimality
# of Communicated Vectors
COCOA (H=1e3)
mini−batch−CD (H=1)
local−SGD (H=1e3)
mini−batch−SGD (H=10)

CoCoA vs. CoCoA$^+$

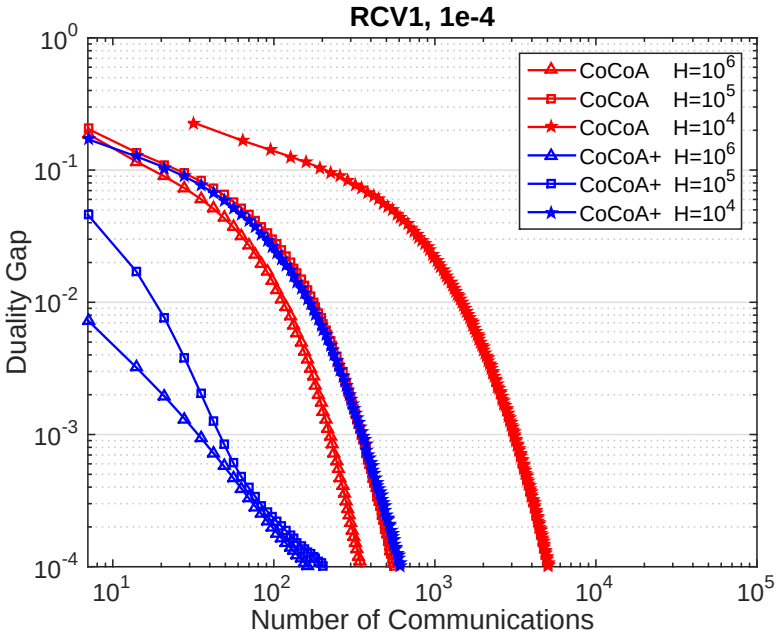


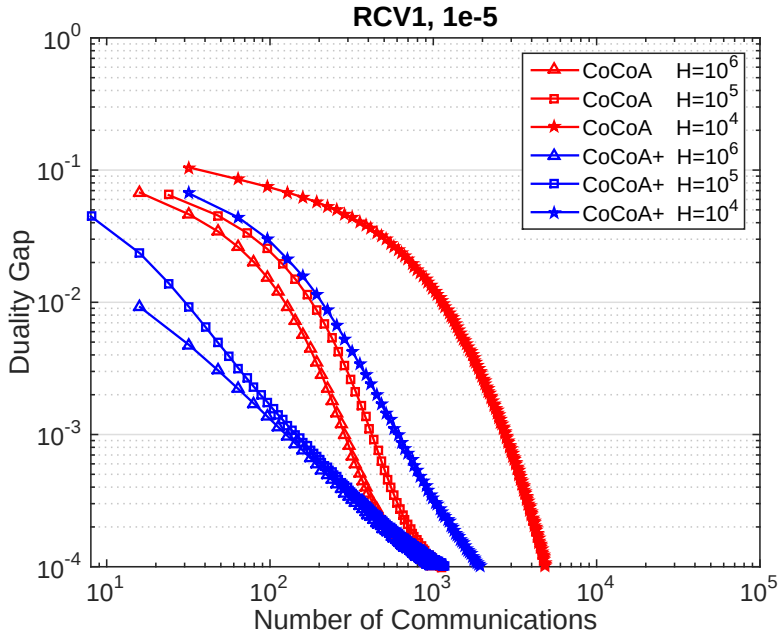


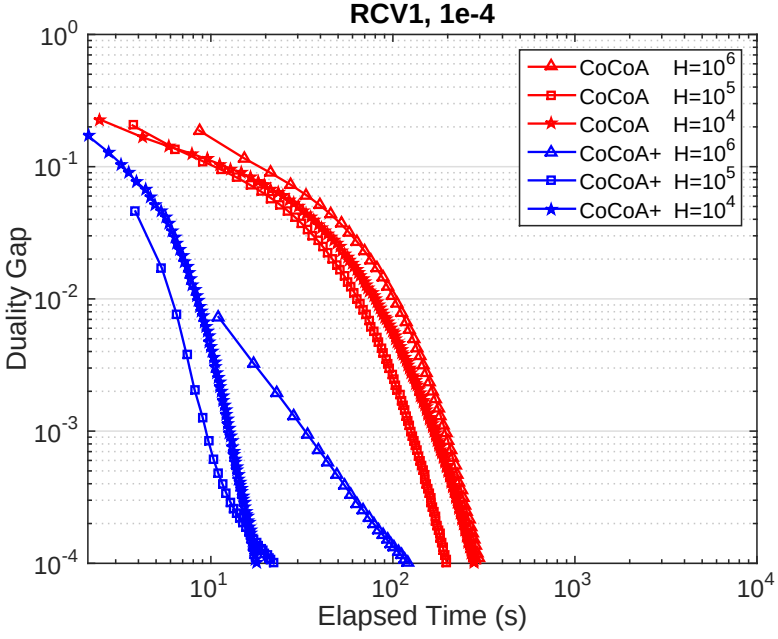


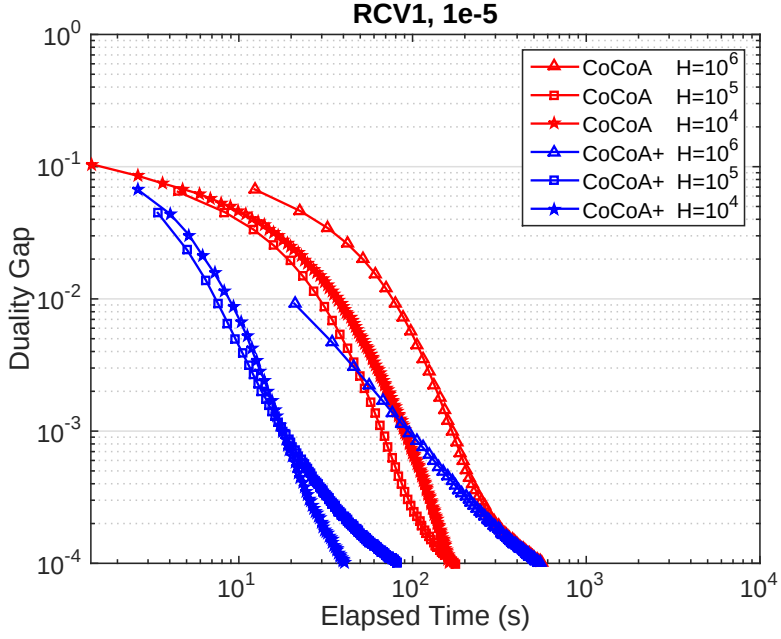


CoCoA vs. CoCoA$^+$

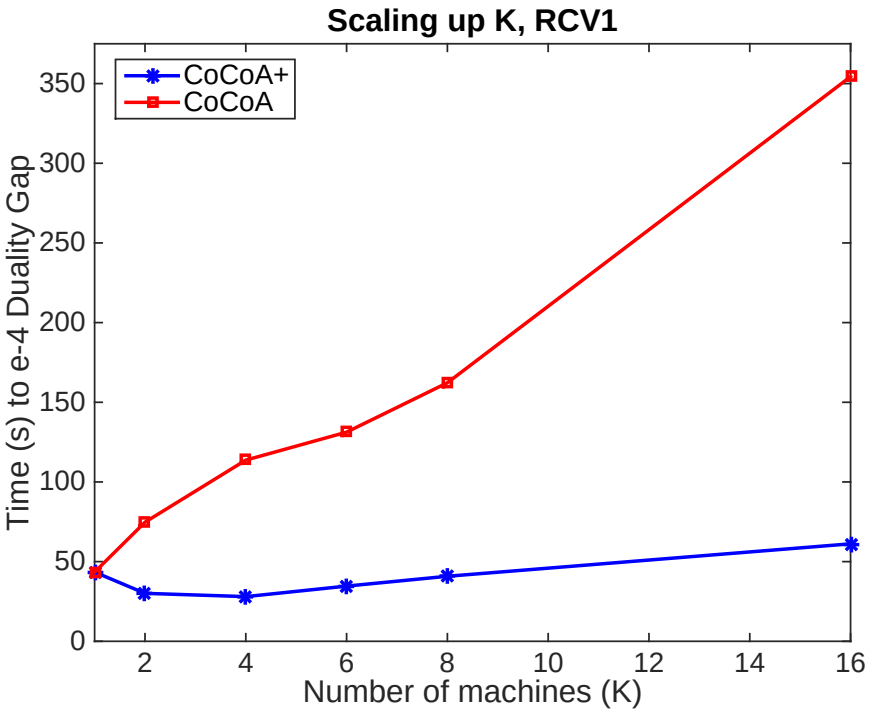


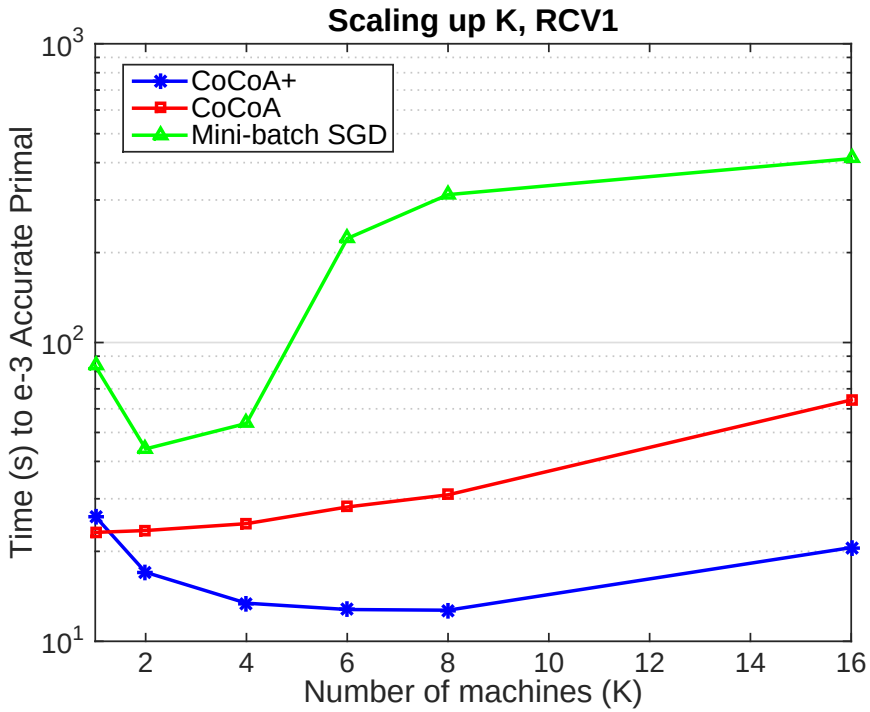


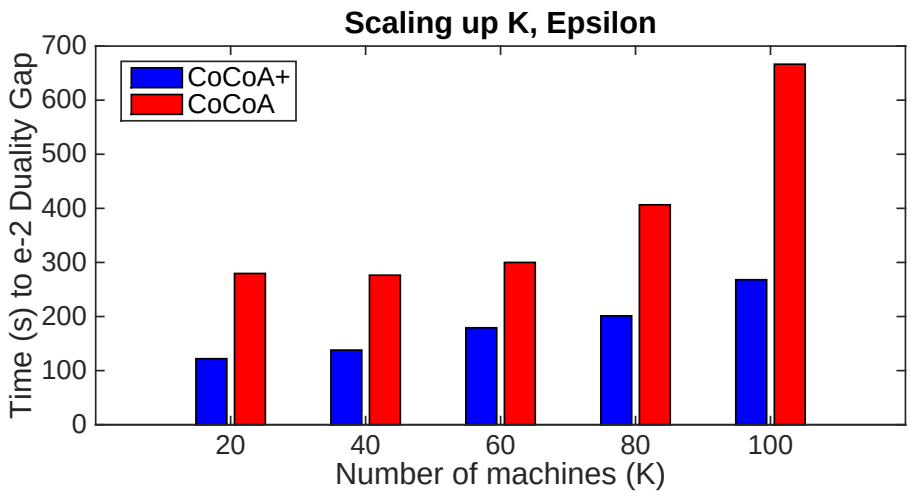




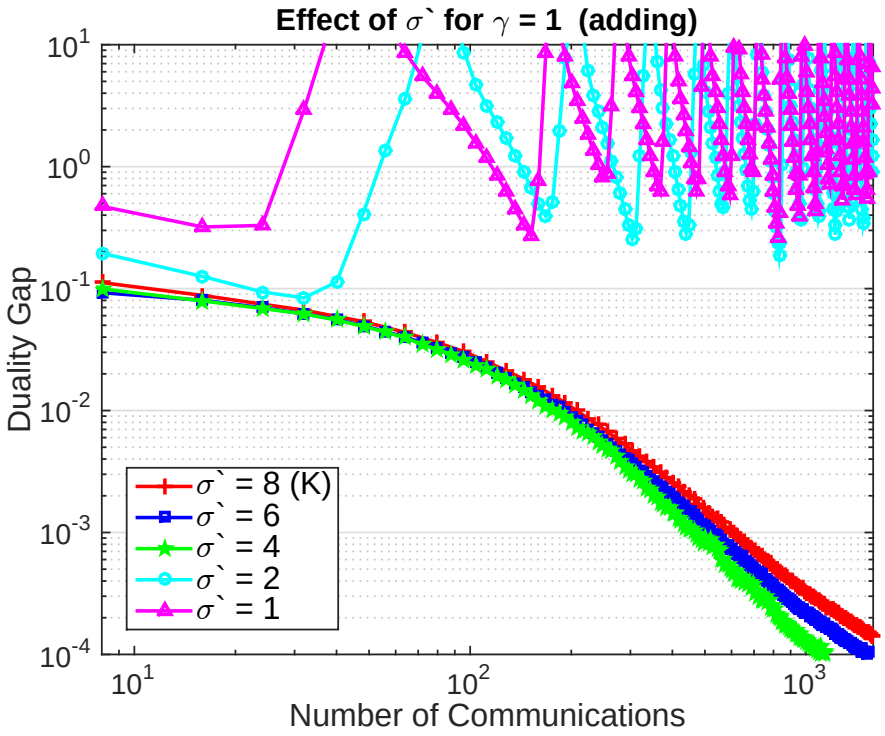
Scaling up

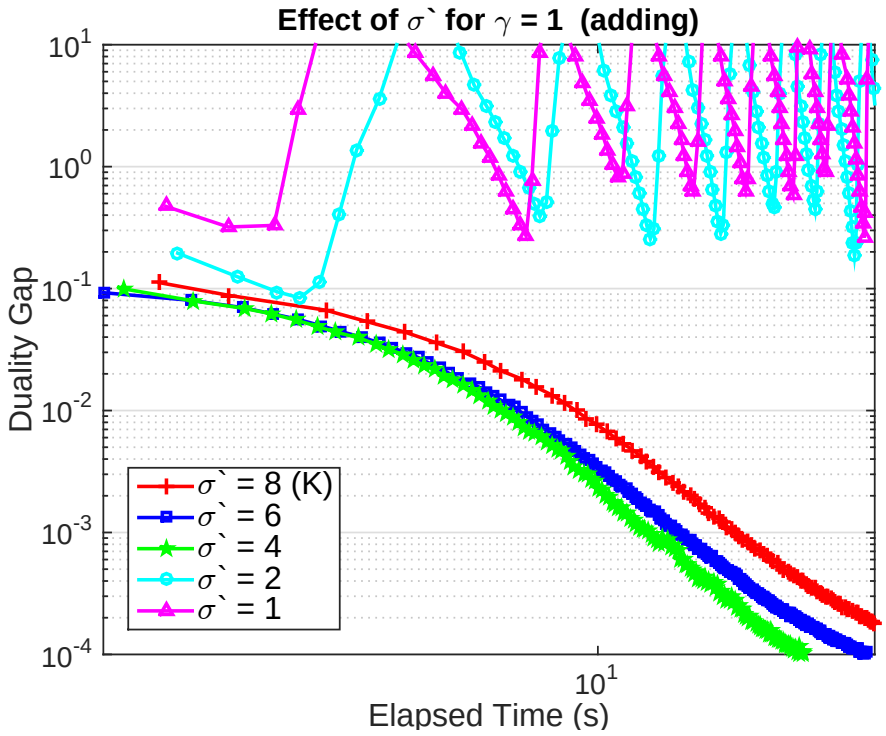

Scaling up K, RCV1
CoCoA+
CoCoA
Time (s) to e-4 Duality Gap
Number of machines (K)
0
50
100
150
200
250
300
350
2
4
6
8
10
12
14
16


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

Effect of σ'





References

1. Chenxin Ma, Virginia Smith, Martin Jaggi, Michael I. Jordan, Peter Richtárik and Martin Takáč: *Adding vs. Averaging in Distributed Primal-Dual Optimization*, ICML 2015.
2. Martin Jaggi, Virginia Smith, Martin Takáč, Jonathan Terhorst, Thomas Hofmann and Michael I. Jordan: *Communication-Efficient Distributed Dual Coordinate Ascent*, NIPS 2014.
3. Richtárik, P. and Takáč, M.: *On optimal probabilities in stochastic coordinate descent methods*, arXiv:1310.3438, 2013.
4. Richtárik, P. and Takáč, M.: *Parallel coordinate descent methods for big data optimization*, arXiv:1212.0873, 2012.
5. Richtárik, P. and Takáč, M.: *Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function*, Mathematical Programming, 2012.
6. Takáč, M., Bijral, A., Richtárik, P. and Srebro, N.: *Mini-batch primal and dual methods for SVMs*, In ICML, 2013.
7. Qu, Z., Richtárik, P. and Zhang, T.: *Randomized dual coordinate ascent with arbitrary sampling*, arXiv:1411.5873, 2014.
8. Qu, Z., Richtárik, P., Takáč, M. and Fercoq, O.: *SDNA: Stochastic Dual Newton Ascent for Empirical Risk Minimization*, arXiv:1502.02268, 2015.
9. Tappenden, R., Takáč, M. and Richtárik, P., *On the Complexity of Parallel Coordinate Descent*, arXiv: 1503.03033, 2015.