An Accelerated Communication-Efficient Primal-Dual Optimization Framework for Structured Machine Learning

Chenxin Ma[a], Martin Jaggi[b], Frank E. Curtis[a], Nathan Srebro[c], and Martin Takáč[a]

[a]Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA, USA;
[b]School of Computer and Communication Sciences, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland;
[c]Toyota Technological Institute at Chicago, Chicago, IL

ARTICLE HISTORY
Compiled July 27, 2019

ABSTRACT
Distributed optimization algorithms are essential for training machine learning models on very large-scale datasets. However, they often suffer from communication bottlenecks. Confronting this issue, a communication-efficient primal-dual coordinate ascent framework (CoCoA) and its improved variant CoCoA+ have been proposed, achieving a convergence rate of $\mathcal{O}(1/t)$ for solving empirical risk minimization problems with Lipschitz continuous losses. In this paper, an accelerated variant of CoCoA+ is proposed and shown to possess a convergence rate of $\mathcal{O}(1/t^2)$ in terms of reducing suboptimality. The analysis of this rate is also notable in that the convergence rate bounds involve constants that, except in extreme cases, are significantly reduced compared to those previously provided for CoCoA+. The results of numerical experiments are provided to show that acceleration can lead to significant performance gains.

KEYWORDS
nonlinear optimization, nonsmooth optimization, distributed optimization, machine learning, accelerated methods

1. Introduction

The use of *distributed* optimization has become essential for training machine learning models on very large datasets. For example, distribution is a key ingredient when the dataset does not fit into the memory of a single machine, and rather must be stored in a distributed manner over many machines or computational agents, each having direct access only to their own (local) segment of the training data.

Efficiently training a machine learning model in such a network setting is challenging due to the cost of communicating information between machines, as compared to the relatively cheap cost of local computation on a single machine. Therefore, designing efficient distributed optimization algorithms, which are able to balance the amount of local computation with the amount of necessary communication between machines, is of crucial importance. This is especially the case since communication bottlenecks and the heterogeneity of large-scale computing systems have been increasing.

CONTACT: Martin Takáč. Email: Takac.MT@gmail.com

In this paper, we improve the existing CoCoA+ framework for communication-efficient distributed optimization [6, 10, 18] by equipping it with *acceleration* [12]. We do not simply apply a generic acceleration scheme, e.g., using the Universal Catalyst [7] (see §1.2 for more discussion), but rather develop a technique specific for our setting. We prove that our accelerated scheme possesses an improved convergence rate involving reduced constants compared to that of CoCoA+. Our theory holds for arbitrary local solvers used on the worker machines. We also provide the results of extensive numerical experiments, where we demonstrate that the new accelerated scheme leads to significantly improved performance in real application settings.

In the remainder of this introduction, we formally state our optimization problems of interest, provide further motivation for the design of sophisticated distributed optimization algorithms, discuss relevant work that has appeared in the literature, summarize our contributions, and outline the remainder of the paper.

1.1. Problem Statements

We present our algorithmic framework in the context of solving the following pair of optimization problems, which are dual to each other:

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^n} \Big[\ \mathcal{O}_A(\boldsymbol{\alpha}) := f(A\boldsymbol{\alpha}) \ + \textstyle\sum_{i\in\mathcal{I}} g_i(\alpha_i)\Big] \quad \text{and} \tag{A}$$

$$\min_{\mathbf{w}\in\mathbb{R}^d} \Big[\ \mathcal{O}_B(\mathbf{w}) := f^*(\mathbf{w}) \ + \textstyle\sum_{i\in\mathcal{I}} g_i^*(-A_i^\top \mathbf{w})\Big]. \tag{B}$$

Here, h^* denotes the *convex conjugate* of a function h. The two problems are defined by the functions $f : \mathbb{R}^d \to \mathbb{R}$, the functions $g_i : \mathbb{R} \to \mathbb{R}$ for all $i \in \mathcal{I}$, and the data matrix $A \in \mathbb{R}^{d\times n}$, about which we make the following assumption.

Assumption 1. *The functions f and g_i for all $i \in \mathcal{I}$ are closed and convex. In addition, the gradient function ∇f is Lipschitz continuous with constant λ and the function g_i^* for each $i \in \mathcal{I}$ is Lipschitz continuous with constant L. Finally, without loss of generality, the columns of the data matrix satisfy $\|A_i\| \le 1$.*

Observe that problem formulation (A) includes many important problems with nonsmooth regularizers, such as Lasso with ℓ_1-norm regularization, i.e.,

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^n} \underbrace{\tfrac{1}{2}\|A\boldsymbol{\alpha}-\mathbf{b}\|_2^2}_{f(A\boldsymbol{\alpha})} + \underbrace{\lambda_1\|\boldsymbol{\alpha}\|_1}_{\sum_{i\in\mathcal{I}} g_i(\alpha_i)}. \tag{1}$$

(It is important to note this regularization term $\|\cdot\|_1$ does not satisfy the requirements in Assumption 1; however, we address how this can be handled in the next paragraph.) More generally, many other smooth data-fitting functions can be used in place of f in our context of problem (A). Problem formulation (B) includes many regularized empirical loss minimization problems with strongly convex regularizers of the form

$$\min_{\mathbf{w}\in\mathbb{R}^d} \sum_{i=1}^{n} \underbrace{\tfrac{1}{n}\ell_i(A_i^\top \mathbf{w})}_{g_i^*(-A_i^T\mathbf{w})} + \underbrace{\tfrac{\lambda_2}{2}\|\mathbf{w}\|^2}_{f^*(\mathbf{w})}, \tag{2}$$

such as support vector machines (SVMs) or logistic regression. Here, the i-th column $A_i \in \mathbb{R}^d$ of A represents the i-th data sample and $\lambda_2 > 0$ is a regularization parameter.

The condition on g^* in Assumption 1 is not very restrictive for many problems of interest. In Section 5 of [4], the authors introduce a *Lipschitzing trick*, which amounts to the idea of modifying a given function $g(\alpha)$ to

$$\tilde{g}(\alpha) := \begin{cases} g(\alpha), & \text{if } \alpha \in B, \\ +\infty, & \text{otherwise,} \end{cases}$$

where B is some compact convex set. If one considers, e.g., problem (1), then one can easily express a bound on the norm of the optimal solution $\boldsymbol{\alpha}^*$ as

$$\lambda_1 \|\boldsymbol{\alpha}^*\|_1 \leq \min_{\boldsymbol{\alpha} \in \mathbb{R}^n} \tfrac{1}{2} \|A\boldsymbol{\alpha} - \mathbf{b}\|_2^2 + \lambda_1 \|\boldsymbol{\alpha}\|_1.$$

This means that, in order to satisfy our assumptions, one only needs to modify $g(\boldsymbol{\alpha})$ such that both the optimal solution and the initial point lies in the set B. It is reasonable to expect that this can be done with appropriately large B.

For our analysis later in the paper, we state the following lemma.

Lemma 1.1 (Corollary 13.3.3 in [14])**.** *Given proper convex g^*, it holds that g^* is L-Lipschitz (w.r.t. the ℓ_2-norm) if and only if g has L-bounded support (w.r.t. the ℓ_2-norm).*

This lemma provides the following bound, of which we make ample use:

$$\forall i : \max_{a,b \in \mathrm{dom}(g_i)} |a - b| \leq 2L. \tag{3}$$

1.2. Motivation and Literature Review

The increased size of optimization problems of interest in machine learning, as well as the availability of parallel and distributed architectures, has led to various directions of research on the design of parallel algorithms for either solving (1) directly (e.g., see [2, 3, 13]) or indirectly by solving its dual (2) (e.g., see [8, 19, 20]). For example, some previous work has focused on shared memory systems in which the amount of parallelism possible is dependent upon the number of processing units (CPUs) on a single node, which is usually in the range of 16–64 cores per node in a contemporary high performance cluster. To avoid locking and, hence, improve speed, some *asynchronous* algorithms have been proposed and analyzed [8, 13]. However, despite the fact that such methods are efficient with shared memory, naïve extensions of these ideas to distributed environments can be terribly inefficient. This is due to the fact that such extensions require continual communication between nodes, causing large overhead.

This type of observation has led many to conclude that, in a distributed computing environment, one must focus not only on the number of data accesses, but also on the number of communication steps, which is usually tied to the number of iterations of the overall optimization algorithm [9]. For this reason, researchers have proposed the use of batch (i.e., full gradient) methods, potentially using (partial or approximate) second-order information as in Newton's method [17, 22] or a quasi-Newton method such as L-BFGS [23]. An important benefit of such methods is that they typically require fewer iterations to achieve high accuracy of the solution. On the other hand,

a Newton-type method requires that a linear system be solved (approximately) in every iteration. A popular approach for this procedure is the linear conjugate gradient (CG) algorithm, such as proposed in DISCO [22]. Unfortunately, however, in many applications one might need a substantial number of CG steps in each Newton iteration. Since each CG step requires one pass over the data and one round of communication, this can also lead to substantial communication overhead.

To overcome these issues, the algorithms COCOA [6], DisDCA [21] and COCOA+ [10, 18] have been proposed to efficiently balance computation and communication in distributed optimization environments. The main idea of these methods, with the data partitioned across any number of nodes in a cluster, is to define meaningful auxiliary subproblems to be solved in each node using only locally stored data. These subproblems involve some inherited (partial) second-order information, which aids in yielding overall fast convergence for solving the original problems (A) and (B).

As is the case for many other optimization methods for solving problems arising in machine learning, the existing COCOA+ framework only yields a sublinear convergence rate of $\mathcal{O}(1/t)$ for the general convex case. That said, we are motivated by the fact that several single machine solvers can be improved by incorporating Nesterov acceleration [1, 12], leading to an improved rate of $\mathcal{O}(1/t^2)$; e.g., this was successfully done in [15]. In this paper, we are able to provably accelerate the distributed COCOA+ framework, achieving a $\mathcal{O}(1/t^2)$ rate for reducing suboptimality. (Compared to [15], our work represents a unique contribution since we consider a distributed framework. We discuss our contributions further in the next subsection.)

It is worthwhile to mention that acceleration could be achieved using the Universal Catalyst proposed in [7]. However, for our purposes, this approach is less appealing since it does not allow the local solver to produce randomized solutions which might only have sufficient quality in expectation. Perhaps the accelerated algorithms most relevant to the present work are those in [5, 11]. The subproblems in these papers are assumed to be strongly convex. In our case, this is not general enough; instead, we manage to exploit the structure of the objective $\mathcal{O}_A$, which is still quite general.

1.3. Contributions

Overall, the contributions of our work can be summarized as follows. Here, for any $\boldsymbol{\alpha} \in \mathbb{R}^n$ in the context of (A), we define a dual solution $\mathbf{w} \in \mathbb{R}^d$ for problem (B) as

$$\mathbf{w} = \mathbf{w}(\boldsymbol{\alpha}) := \nabla f(A\boldsymbol{\alpha})\,. \tag{4}$$

- We propose and analyze an accelerated communication-efficient block descent algorithm for solving problems (A) and (B). Our analysis shows that our framework possesses a convergence rate of $\mathcal{O}(1/t^2)$ for suboptimality, i.e., with $\boldsymbol{\alpha}_t$ representing the t-th algorithm iterate, we prove the accelerated reduction of $\mathcal{O}_A(\boldsymbol{\alpha}_t) - \mathcal{O}_A(\boldsymbol{\alpha}_\star)$ over time t, where $\boldsymbol{\alpha}_\star$ represents an optimal solution of (A). Given the recent work in [4], our results for reducing suboptimality can then also be cast in terms of an accelerated rate of the duality gap $G(\boldsymbol{\alpha}) := \mathcal{O}_A(\boldsymbol{\alpha}) + \mathcal{O}_B(\mathbf{w}(\boldsymbol{\alpha}))$, as a practically important accuracy certificate. Overall, we refer to our method as a *primal-dual* approach since the primal iterate updates occur in parallel, the dual iterates are communicated to the nodes, and since our theoretical results can be cast in terms of accelerated convergence of the duality gap to zero.

- The convergence analyses of CoCoA and CoCoA+ yielded bounds involving quantities which depend on the Lipschitz constants for the local subproblems. In this sense, the results might be no better than similar bounds for inexact block proximal gradient descent. By contrast, in this paper, we exploit the structure of the dual objective $\mathcal{O}_B$ to obtain complexity bounds that do not depend on Lipschitz constants of the auxiliary subproblems. Instead, our bounds merely depend on quantities related to local curvature corresponding to the subproblems.
- We extend our accelerated framework also to cover general non-strongly convex regularizers. This is important, e.g., in the context of ℓ_1-norm regularized Lasso, sparse logistic regression, and elastic net regularized problems.
- We have performed numerous numerical experiments to demonstrate that acceleration can lead to significant performance improvements. In particular, the gains are especially large for small values of the regularization parameter λ_2, which is important in very-large-scale settings where one often desires the regularization to be inversely proportional to n. The C++ code for our implementation of our framework is available on github: `https://github.com/schemmy/CoCoA-Experiments`.

We remark at the outset that one should not expect convergence guarantees for a distributed optimization algorithm to be as good as—let alone better than—corresponding guarantees for a centralized algorithm. After all, with data distributed across multiple machines and with practical reasons for limiting communication, it is natural for a theoretical convergence guarantee to degrade as more machines are involved. That said, experience has shown that good distributed optimization approaches can maintain the same convergence *rate* as a centralized approach, albeit with a larger *constant* that depends on the number of machines. This characterizes the accelerated convergence guarantees that we prove in this paper. Theoretically, we show that the convergence rate for our approach is no worse than for an accelerated centralized approach, though the constant depends on the number of machines, and that empirically our distributed algorithm achieves much better performance in terms of iterations and CPU time.

1.4. Organization

The remainder of the paper is organized as follows. We start by introducing our new accelerated CoCoA+ (AccCoCoA+) algorithm, including the design of its subproblems and strategies for solving them. We then describe our main complexity result for the algorithm, showing its improvement over that for CoCoA+. Finally, we comment on the results of our numerical experiments.

2. Accelerated CoCoA+

In this section, we introduce our proposed accelerated CoCoA+ (AccCoCoA+) algorithm. We begin by defining notation related to the manner in which data is distributed across various machines, then define quantities related to the local subproblems to be solved in each iteration of the algorithm. Of central importance for these subproblems are a carefully defined regularization scheme and a loose assumption on the accuracy to which each subproblem must be solved in each step of AccCoCoA+.

2.1. Data Partitioning

Suppose that the n columns of the dataset are split across K machines (nodes). Let the index set of columns stored on node $k \in [K] := \{1, \ldots, k\}$ be denoted as $\mathcal{P}_k$. We assume that the dataset is partitioned in a disjoint manner such that $\mathcal{P}_i \cap \mathcal{P}_j = \emptyset$ for any $i \neq j$ while $\cup_{k=1}^K \mathcal{P}_k = [n]$. For notational convenience, we split the vector $\boldsymbol{\alpha} \in \mathbb{R}^n$ into the set of K vectors $\{\boldsymbol{\alpha}^{[k]}\}_{k=1}^K$ by employing, for each $k \in [K]$, the masking operator

$$(\boldsymbol{\alpha}^{[k]})_i := \begin{cases} \alpha_i & \text{if } i \in \mathcal{P}_k, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

As a consequence of separability of g in (A), we write

$$\psi_k(\boldsymbol{\alpha}^{[k]}) := \sum_{i \in \mathcal{P}_k} g_i(\alpha_i), \tag{6}$$

so that $g(\boldsymbol{\alpha}) = \sum_{k=1}^K \psi_k(\boldsymbol{\alpha}^{[k]})$.

2.2. Subproblem

Iteration t of AccCoCoA+ involves the auxiliary vectors $(\mathbf{y}_t, \mathbf{z}_t) \in \mathbb{R}^n \times \mathbb{R}^n$, which one may split into $\{\mathbf{y}_t^{[k]}\}_{k=1}^K$ and $\{\mathbf{z}_t^{[k]}\}_{k=1}^K$, respectively, in the same manner as $\boldsymbol{\alpha} \in \mathbb{R}^n$ in (5). The goal in iteration t on each node $k \in [K]$ is to solve (approximately)

$$\min_{\mathbf{z}_{t+1}^{[k]} \in \mathbb{R}^n} \mathcal{G}_k(\mathbf{z}_{t+1}^{[k]}; \mathbf{y}_t, \mathbf{z}_t), \tag{7}$$

where, for scalars $\sigma' \geq 0$ and $\theta_t \geq 0$ (see below), the local objective function is

$$\begin{aligned} \mathcal{G}_k(\mathbf{z}_{t+1}^{[k]}; \mathbf{y}_t, \mathbf{z}_t) := \psi_k(\mathbf{z}_{t+1}^{[k]}) + \frac{1}{K} f(A\mathbf{y}_t) \\ + \nabla f(A\mathbf{y}_t)^\top A(\mathbf{z}_{t+1}^{[k]} - \mathbf{y}_t^{[k]}) + \frac{\lambda \theta_t \sigma'}{2} \left\| A(\mathbf{z}_{t+1}^{[k]} - \mathbf{z}_t^{[k]}) \right\|^2. \end{aligned} \tag{8}$$

At first glance, it is not obvious that this subproblem can be solved only using local data on node k due to the presence of the term $\nabla f(A\mathbf{y}_t)^\top A(\mathbf{z}_{t+1}^{[k]} - \mathbf{y}_t^{[k]})$, which is dependent on the entire dataset. However, by simply making the single shared vector $\mathbf{w}_t := \nabla f(A\mathbf{y}_t)$ available on each node, the local subproblem (7) only requires knowledge of the pair $(\mathbf{y}_t^{[k]}, \mathbf{z}_t^{[k]})$ and the local part of A, and not the full vectors $(\mathbf{y}_t, \mathbf{z}_t)$. Therefore, the storage of the variable vectors $\mathbf{y}$ and $\mathbf{z}$ can also be distributed.

The last term in (8) represents a regularization term in which the parameter $\sigma' \geq 0$ plays a critical role. It can be interpreted as a measure for the cross-dependency of the partitioning of the data. For our analysis, this term must be chosen to satisfy

$$\sigma' \geq \sigma'_{min} := \gamma \max_{\boldsymbol{\alpha} \in \mathbb{R}^n} \frac{\|A\boldsymbol{\alpha}\|^2}{\sum_{k=1}^K \|A\boldsymbol{\alpha}^{[k]}\|^2} \quad \text{for some} \quad \gamma \in [\tfrac{1}{K}, 1]. \tag{9}$$

It is easy to show that $\sigma'_{min}/\gamma = \max_{\boldsymbol{\alpha}\in\mathbb{R}^n} \frac{\|A\boldsymbol{\alpha}\|^2}{\sum_{k=1}^K \|A\boldsymbol{\alpha}^{[k]}\|^2} \in [1, K]$. If σ'_{min}/γ is equal to 1, then any pair of samples from different elements of the partition must be orthogonal to each other. In such cases, the function $\mathcal{O}_A$ is block-separable. On the other hand, if σ'_{min}/γ is close to K, then the data across the partition are strongly correlated. Note that the choice $\sigma' = \gamma K$ is "safe" in the sense that (9) holds; see [10, Lemma 4].

It is worthwhile to emphasize the dependence of σ'_{min}, and hence of σ', on the parameter γ, even though, for simplicity, we do not make this dependence explicit in our notation. In turn, the parameter γ leads to different theoretical and empirical trade-offs over its range $[\frac{1}{K}, 1]$. We discuss these trade-offs further in §3.1 and §4.

2.3. Approximate Subproblem Solutions

A strength of our framework is that each subproblem (7) need not be solved exactly. This is critical since, in the extreme, solving the subproblems exactly can be as difficult as solving the original problem. In AccCoCoA+, we make the assumption that, in iteration t, the solver employed to solve the subproblem on node k yields an approximate solution with some additive error $\epsilon_t \geq 0$. To be precise, we make the following assumption.

Assumption 2 (ϵ_t-approximate solutions)**.** *There exists a sequence $\{\epsilon_t\}_{t=0}^{\infty} \geq 0$ such that, for each $t \in \{0, 1, 2, \dots\}$ and $k \in [K]$, the local solver employed on node k in iteration t produces a (possibly random) $\mathbf{z}_{t+1}^{[k]}$ satisfying*

$$\mathbf{E}_{t+1}[\mathcal{G}_k(\mathbf{z}_{t+1}^{[k]}; \mathbf{y}_t, \mathbf{z}_t)|t] \leq \mathcal{G}_k({\mathbf{z}^\star_{t+1}}^{[k]}; \mathbf{y}_t, \mathbf{z}_t) + \epsilon_t, \tag{10}$$

where ${\mathbf{z}^\star_{t+1}}^{[k]} := \arg\min_{\mathbf{z}_{t+1}^{[k]}\in\mathbb{R}^n} \mathcal{G}_k(\mathbf{z}_{t+1}^{[k]}; \mathbf{y}_t, \mathbf{z}_t)$ and $\mathbf{E}_{t+1}[\cdot|t]$ indicates conditional expectation given the algorithm history up to time t.

2.4. Algorithm

AccCoCoA+ is stated as Algorithm 1. Given an initial iterate vector $\boldsymbol{\alpha}_0$ in the effective domain of g and the scalar $\theta_0 = 1$, each iteration involves a series of steps, only one of which involves communication between nodes. First, the auxiliary vectors $\{\mathbf{y}_t^{[k]}\}_{k=1}^K$ are set on each node, each representing a convex combination of the variables $\boldsymbol{\alpha}_t^{[k]}$ and $\mathbf{z}_t^{[k]}$. Then, for setting up the local objective (8) for each subproblem (7), the combined vector $\mathbf{y}_t$ is used to compute $\mathbf{w}_t$, which must be communicated to all nodes. After this point in iteration t, all remaining steps involve local computation on each node: each subproblem is solved approximately to compute $\{\mathbf{z}_{t+1}^{[k]}\}_{k=1}^K$, after which the elements of $\{\boldsymbol{\alpha}_{t+1}^{[k]}\}_{k=1}^K$ are set. Acceleration of the algorithm is due to the careful update for the sequence $\{\theta_t\}$, which, since it only involves a prescribed formula for a scalar quantity, can be performed identically on each node. Observe that the update sequence ensures that $\theta_t \sim \mathcal{O}(1/t)$.

3. Convergence Analysis

In this section, we study the convergence properties of the proposed AccCoCoA+ algorithm. First, we prove general complexity results, then, respectively in Sections 3.1

Algorithm 1 Accelerated CoCoA+ (AccCoCoA+)

1: choose $\boldsymbol{\alpha}_0 \in \mathrm{dom}(g) \subseteq \mathbf{R}^n$; set $\mathbf{z}_0 := \boldsymbol{\alpha}_0$, $\gamma \in [\frac{1}{K}, 1]$ and $\theta_0 := 1$
2: **for** $t \in \{0, 1, 2, \dots\}$ **do**
3: **for** $k \in [K]$ in parallel, set

$$\mathbf{y}_t^{[k]} := (1 - \gamma\theta_t)\boldsymbol{\alpha}_t^{[k]} + \gamma\theta_t \mathbf{z}_t^{[k]} \tag{11}$$

4: set $\mathbf{w}_t := \nabla f(A\mathbf{y}_t)$ and communicate to all nodes
5: **for** $k \in [K]$ in parallel, compute an ϵ_t-approximate solution $\mathbf{z}_{t+1}^{[k]}$ of subproblem (7)
6: **for** $k \in [K]$ in parallel, set

$$\boldsymbol{\alpha}_{t+1}^{[k]} := \mathbf{y}_t^{[k]} + \gamma\theta_t(\mathbf{z}_{t+1}^{[k]} - \mathbf{z}_t^{[k]}) \tag{12}$$

7: set $\theta_{t+1} := \frac{\sqrt{\gamma^2\theta_t^4 + 4\theta_t^2} - \gamma\theta_t^2}{2}$ (on each node)
8: **end for**

and 3.2, we provide interpretations of this main theorem for cases when the subproblems are solved exactly or inexactly.

First, we prove the following lemma related to the sequence $\{\theta_t\}_{t\geq 0}$. The result is similar to that given as Lemma 1 in [5].

Lemma 3.1. *The sequence $\{\theta_t\}_{t\geq 0}$ is positive, monotonically decreasing, and has*

$$\frac{1 - \gamma\theta_{t+1}}{\theta_{t+1}^2} = \frac{1}{\theta_t^2} \tag{13}$$

and

$$\theta_t \leq \frac{2}{t\gamma + 2} \leq 1 \tag{14}$$

for all $t \geq 0$.

Proof. For each $t \geq 0$, the value θ_{t+1} can be seen from Step 7 of Algorithm 1 to be the positive root of the quadratic equation

$$\theta^2 + (\gamma\theta_t^2)\theta - \theta_t^2 = 0. \tag{15}$$

Since $\theta_0 = 1$, it follows from (15), the fact that a strongly convex quadratic univariate function with a negative vertical intercept has a positive real root, and a simple inductive argument that $\theta_t > 0$ for all $t \geq 0$, as desired. Next, plugging in θ_{t+1} for θ in (15) and rearranging, we obtain (13). This can again be rearranged to yield

$$\frac{1}{\theta_{t+1}^2} = \frac{1}{\theta_t^2} + \frac{\gamma}{\theta_{t+1}}, \tag{16}$$

from which it follows that

$$\frac{1}{\theta_t^2} = 1 + \sum_{i=1}^{t} \frac{\gamma}{\theta_i} \quad \text{for all } t \geq 0.$$

This shows that $\{\theta_t\}_{t\geq 0}$ is monotonically decreasing, as desired. We now use mathematical induction to show (14). First, (14) clearly holds for $t = 0$, for which we have $\theta_0 = \frac{2}{0+2} = 1$. Assuming it holds up to t, we have from (16) that

$$\frac{1}{\theta_{t+1}^2} - \frac{\gamma}{\theta_{t+1}} = \frac{1}{\theta_t^2} \geq \frac{(\gamma t + 2)^2}{4}. \tag{17}$$

Now observe that the quadratic equation in the variable $1/\theta$ given by

$$\frac{1}{\theta^2} - \frac{\gamma}{\theta} - \frac{(\gamma t + 2)^2}{4} = 0 \;\text{ has roots }\; \frac{1}{\theta} = \frac{\gamma \pm \sqrt{\gamma^2 + (\gamma t + 2)^2}}{2}.$$

This shows that, by (17) and since $\theta_{t+1} > 0$, we have

$$\frac{1}{\theta_{t+1}} \geq \frac{\gamma + \sqrt{\gamma^2 + (\gamma t + 2)^2}}{2} \geq \frac{\gamma + \sqrt{(\gamma t + 2)^2}}{2} = \frac{\gamma(t+1) + 2}{2}.$$

Therefore, we conclude $\theta_{t+1} \leq \frac{2}{(t+1)\gamma + 2} \leq 1$, which concludes the proof. □

Next, we prove the following lemma, which is a modification of Lemma 2 in [5].

Lemma 3.2. *Let $\{\boldsymbol{\alpha}_t, \mathbf{z}_t\}_{t\geq 0}$ be generated by Algorithm 1. Then, for all $t \geq 0$,*

$$\boldsymbol{\alpha}_t = \sum_{l=0}^{t} \rho_t^l \mathbf{z}_l, \tag{18}$$

where the coefficients $\{\rho_t^0, \rho_t^1, ..., \rho_t^t\}$ are nonnegative and sum to 1; i.e., $\boldsymbol{\alpha}_t$ is a convex combination of the vectors $\{\mathbf{z}_0, \mathbf{z}_1, ..., \mathbf{z}_t\}$. More precisely, the coefficients are defined recursively in t as $\rho_0^0 = 1, \rho_1^0 = 1 - \gamma\theta_0, \rho_1^1 = \gamma\theta_0$ and, for all $t \geq 1$,

$$\rho_{t+1}^l = \begin{cases} (1 - \gamma\theta_t)\rho_t^l & \text{for } \; l \in \{0, ..., t\}, \\ \gamma\theta_t & \text{for } \; l = t + 1. \end{cases} \tag{19}$$

Proof. We proceed by induction. First, notice that $\boldsymbol{\alpha}_0 = \mathbf{z}_0 = \rho_0^0 \mathbf{z}_0$ where $\rho_0^0 := 1$. By (11), this implies that $\mathbf{y}_0 = \mathbf{z}_0$, which together with $\theta_0 = 1$ gives (see (12))

$$\boldsymbol{\alpha}_1 = \mathbf{y}_0 + \gamma\theta_0(\mathbf{z}_1 - \mathbf{z}_0) = \mathbf{z}_0 + \gamma\theta_0(\mathbf{z}_1 - \mathbf{z}_0) = (1 - \gamma\theta_0)\mathbf{z}_0 + \gamma\theta_0\mathbf{z}_1,$$

which proves (18) for $t = 1$. Assuming now that (18) holds for some $t \geq 1$, we obtain

$$\begin{aligned} \boldsymbol{\alpha}_{t+1} &\overset{(12)}{=} \mathbf{y}_t + \gamma\theta_t(\mathbf{z}_{t+1} - \mathbf{z}_t) \overset{(11)}{=} (1 - \gamma\theta_t)\boldsymbol{\alpha}_t + \gamma\theta_t\mathbf{z}_t + \gamma\theta_t(\mathbf{z}_{t+1} - \mathbf{z}_t) \\ &\overset{(18)}{=} (1 - \gamma\theta_t)\sum_{l=0}^{t} \rho_t^l \mathbf{z}_l + \gamma\theta_t\mathbf{z}_t + \gamma\theta_t(\mathbf{z}_{t+1} - \mathbf{z}_t) \\ &\overset{(18)}{=} \sum_{l=0}^{t} \underbrace{(1 - \gamma\theta_t)\rho_t^l}_{\rho_{t+1}^l} \mathbf{z}_l + \underbrace{\gamma\theta_t}_{\rho_{t+1}^{t+1}} \mathbf{z}_{t+1}. \end{aligned} \tag{20}$$

From (14) and since $\gamma \in [\frac{1}{K}, 1]$, it follows that $\rho_{t+1}^l \geq 0$ for all $l \in \{0, \ldots, t+1\}$. It remains to show that the constants sum to 1. This is true since, for all $t \geq 1$, the weights in (20) (for $\boldsymbol{\alpha}_{t+1}$) are obtained by taking the corresponding weights for $\boldsymbol{\alpha}_t$, multiplying them by $\gamma\theta_t \in (0, 1]$, then including the weight for $\mathbf{z}_{t+1}$ as $\gamma\theta_t$. □

Our next result relates to the optimal solution of an instance of subproblem (7).

Lemma 3.3. *Let* $\mathbf{u}^{\star[k]} := \arg\min_{\mathbf{u}^{[k]}} \mathcal{G}_k(\mathbf{u}^{[k]}; \mathbf{y}, \mathbf{z})$. *Then, for all* $\mathbf{u}^{[k]}$,

$$\mathcal{G}_k(\mathbf{u}^{[k]}; \mathbf{y}, \mathbf{z}) \geq \mathcal{G}_k(\mathbf{u}^{\star[k]}; \mathbf{y}, \mathbf{z}) + \frac{\lambda\theta_t\sigma'}{2}\|A(\mathbf{u}^{[k]} - \mathbf{u}^{\star[k]})\|^2. \tag{21}$$

Proof. Using convexity of f and g_i for all $i \in \mathcal{I}$ (and, therefore, convexity of ψ_k defined in (6)), one finds that for $\mathbf{v}^{[k]} \in \partial\psi_k(\mathbf{u}^{\star[k]})$ one has

$$\begin{aligned}
&\mathcal{G}_k(\mathbf{u}^{[k]}; \mathbf{y}, \mathbf{z}) \\
&\overset{(8)}{=} \psi_k(\mathbf{u}^{[k]}) + \frac{1}{K}f(A\mathbf{y}) + \nabla f(A\mathbf{y})^\top A(\mathbf{u}^{[k]} - \mathbf{y}^{[k]}) + \frac{\lambda\theta_t\sigma'}{2}\left\|A(\mathbf{u}^{[k]} - \mathbf{z}^{[k]})\right\|^2 \\
&\geq \psi_k(\mathbf{u}^{\star[k]}) + \langle \mathbf{v}^{[k]}, \mathbf{u}^{[k]} - \mathbf{u}^{\star[k]} \rangle \\
&\qquad + \frac{1}{K}f(A\mathbf{y}) + \nabla f(A\mathbf{y})^\top A(\mathbf{u}^{[k]} - \mathbf{u}^{\star[k]} + \mathbf{u}^{\star[k]} - \mathbf{y}^{[k]}) \\
&\qquad + \frac{\lambda\theta_t\sigma'}{2}\left\|A(\mathbf{u}^{[k]} - \mathbf{u}^{\star[k]} + \mathbf{u}^{\star[k]} - \mathbf{z}^{[k]})\right\|^2 \\
&= \psi_k(\mathbf{u}^{\star[k]}) + \langle \mathbf{v}^{[k]}, \mathbf{u}^{[k]} - \mathbf{u}^{\star[k]} \rangle \\
&\qquad + \frac{1}{K}f(A\mathbf{y}) + \nabla f(A\mathbf{y})^\top A(\mathbf{u}^{[k]} - \mathbf{u}^{\star[k]}) + \nabla f(A\mathbf{y})^\top A(\mathbf{u}^{\star[k]} - \mathbf{y}^{[k]}) \\
&\qquad + \frac{\lambda\theta_t\sigma'}{2}\left\|A(\mathbf{u}^{[k]} - \mathbf{u}^{\star[k]})\right\|^2 + \frac{\lambda\theta_t\sigma'}{2}\left\|A(\mathbf{u}^{\star[k]} - \mathbf{z}^{[k]})\right\|^2 \\
&\qquad + \lambda\theta_t\sigma'(\mathbf{u}^{[k]} - \mathbf{u}^{\star[k]})^\top A^\top A(\mathbf{u}^{\star[k]} - \mathbf{z}^{[k]}) \\
&= \underbrace{\psi_k(\mathbf{u}^{\star[k]}) + \frac{1}{K}f(A\mathbf{y}) + \nabla f(A\mathbf{y})^\top A(\mathbf{u}^{\star[k]} - \mathbf{y}^{[k]}) + \frac{\lambda\theta_t\sigma'}{2}\left\|A(\mathbf{u}^{\star[k]} - \mathbf{z}^{[k]})\right\|^2}_{=\mathcal{G}_k(\mathbf{u}^{\star[k]}; \mathbf{y}, \mathbf{z})} \\
&\qquad + \langle \mathbf{v}^{[k]} + A^\top\nabla f(A\mathbf{y}) + \lambda\theta_t\sigma' A^\top A(\mathbf{u}^{\star[k]} - \mathbf{z}^{[k]}), \mathbf{u}^{[k]} - \mathbf{u}^{\star[k]} \rangle \\
&\qquad + \frac{\lambda\theta_t\sigma'}{2}\left\|A(\mathbf{u}^{[k]} - \mathbf{u}^{\star[k]})\right\|^2 \\
&\geq \mathcal{G}(\mathbf{u}^{\star[k]}; \mathbf{y}, \mathbf{z}) + \frac{\lambda\theta_t\sigma'}{2}\left\|A(\mathbf{u}^{[k]} - \mathbf{u}^{\star[k]})\right\|^2,
\end{aligned}$$

where the fact that

$$\mathbf{v}^{[k]} + A^\top\nabla f(A\mathbf{y}) + \lambda\theta_t\sigma' A^\top A(\mathbf{u}^{\star[k]} - \mathbf{z}^{[k]}) \in \partial\mathcal{G}_k(\mathbf{u}^{\star[k]}; \mathbf{y}, \mathbf{z})$$

yields the final inequality. □

We now present our main convergence theorem. Our approach for proving the result is based on the use of randomized estimated sequences, as in [5]. However, we have included an important improvement to this approach that allows us to consider subproblems that are not strongly convex. This is only possible due to the special structure of $\mathcal{O}_A$.

Theorem 3.4. *For any optimal solution $\alpha_\star$ of problem* (A) *and all $t \geq 1$,*

$$\begin{aligned}&\mathbf{E}[\mathcal{O}_A(\boldsymbol{\alpha}_t) - \mathcal{O}_A(\boldsymbol{\alpha}_\star)] \\ &\leq \frac{4}{(t\gamma - \gamma + 2)^2}\left((1-\gamma)(\mathcal{O}_A(\boldsymbol{\alpha}_0) - \mathcal{O}_A(\boldsymbol{\alpha}_\star)) + \frac{\gamma\lambda\sigma'}{2}C + K\epsilon_0\gamma + \sum_{j=1}^{t-1} E_j\right) \qquad (22)\end{aligned}$$

holds where

$$C := \sum_{k=1}^{K} \|A(\boldsymbol{\alpha}_\star^{[k]} - \boldsymbol{\alpha}_0^{[k]})\|^2, \qquad (23a)$$

$$E_j := \frac{K\epsilon_j\gamma}{\theta_j} + \frac{\gamma}{\theta_{j-1}}\epsilon_{j-1} + \gamma\mathcal{R}\sqrt{\frac{2\lambda\sigma'}{\theta_{j-1}}\epsilon_{j-1}} \;\; \textit{for all} \;\; j \in \{1,\dots,t-1\}, \qquad (23b)$$

$$\textit{and} \;\; \mathcal{R} := \max_{k\in\{1,\dots,K\}, a,b\in\{\alpha\in\mathbb{R}^n \;|\; \forall i:\; \alpha_i\in\mathrm{dom}(g_i)\}} \|A(a^{[k]} - b^{[k]})\| \leq \max_{k\in\{1,\dots,K\}} 2Ln_k. \qquad (23c)$$

Proof. By convexity of g, it follows that, for all $t \geq 0$, one has

$$g(\boldsymbol{\alpha}_t) \overset{(18)}{=} g\left(\sum_{l=0}^{t}\rho_t^l \mathbf{z}_l\right) \leq \sum_{l=0}^{t}\rho_t^l g(\mathbf{z}_l) =: \hat{g}^t. \qquad (24)$$

Combining this definition and the result from Lemma 3.2, one gets that

$$\hat{g}^{t+1} \overset{(24),(19)}{=} \sum_{l=0}^{t+1}\rho_{t+1}^l g(\mathbf{z}_l) = \gamma\theta_t g(\mathbf{z}_{t+1}) + \sum_{l=0}^{t}\rho_{t+1}^l g(\mathbf{z}_l). \qquad (25)$$

On the other hand, under Assumption 1, one finds that

$$\begin{aligned}&f(A\boldsymbol{\alpha}_{t+1}) \\ &\overset{(12)}{=} f(A\mathbf{y}_t + \gamma\theta_t A(\mathbf{z}_{t+1} - \mathbf{z}_t)) \\ &\leq f(A\mathbf{y}_t) + \langle \nabla f(A\mathbf{y}_t), \gamma\theta_t A(\mathbf{z}_{t+1} - \mathbf{z}_t)\rangle + \frac{\lambda\gamma^2\theta_t^2}{2}\|A(\mathbf{z}_{t+1} - \mathbf{z}_t)\|^2 \\ &\overset{(9)}{\leq} f(A\mathbf{y}_t) + \gamma\theta_t\left\langle A^\top\nabla f(A\mathbf{y}_t), \sum_{k=1}^{K}(\mathbf{z}_{t+1}^{[k]} - \mathbf{z}_t^{[k]})\right\rangle + \frac{\lambda\gamma\sigma'\theta_t^2}{2}\sum_{k=1}^{K}\left\|A(\mathbf{z}_{t+1}^{[k]} - \mathbf{z}_t^{[k]})\right\|^2 \\ &= \sum_{k=1}^{K}\Bigg\{\frac{1}{K}(1-\gamma\theta_t)f(A\mathbf{y}_t) + \gamma\theta_t\left\langle A^\top\nabla f(A\mathbf{y}_t), \mathbf{y}_t^{[k]} - \mathbf{z}_t^{[k]}\right\rangle \\ &\qquad + \gamma\theta_t\Big(\frac{1}{K}f(A\mathbf{y}_t) + \left\langle A^\top\nabla f(A\mathbf{y}_t), \mathbf{z}_{t+1}^{[k]} - \mathbf{y}_t^{[k]}\right\rangle + \frac{\lambda\sigma'\theta_t}{2}\left\|A(\mathbf{z}_{t+1}^{[k]} - \mathbf{z}_t^{[k]})\right\|^2\Big)\Bigg\}.\end{aligned}$$

Next, note that from the definition of $\mathbf{y}_t$ in the algorithm one finds

$$\begin{aligned}\mathbf{y}_t^{[k]} &\overset{(11)}{=} (1-\gamma\theta_t)\boldsymbol{\alpha}_t^{[k]} + \gamma\theta_t\mathbf{z}_t^{[k]}, \\ \gamma\theta_t\mathbf{y}_t^{[k]} - \gamma\theta_t\mathbf{z}_t^{[k]} &= (1-\gamma\theta_t)\boldsymbol{\alpha}_t^{[k]} - (1-\gamma\theta_t)\mathbf{y}_t^{[k]},\end{aligned}$$

$$\gamma\theta_t(\mathbf{y}_t^{[k]} - \mathbf{z}_t^{[k]}) = (1-\gamma\theta_t)(\boldsymbol{\alpha}_t^{[k]} - \mathbf{y}_t^{[k]}). \tag{26}$$

Defining, for all $t \geq 0$, an upper-bound on $\mathcal{O}_A(\boldsymbol{\alpha}_t)$ as

$$\hat{\mathcal{O}_A}^t := \hat{g}^t + f(A\boldsymbol{\alpha}_t) \overset{(24)}{\geq} \mathcal{O}_A(\boldsymbol{\alpha}_t), \tag{27}$$

it follows from above and convexity of f that one has

$$\begin{aligned}
\hat{\mathcal{O}_A}^{t+1} &\overset{(27)}{=} \hat{g}^{t+1} + f(A\boldsymbol{\alpha}_{t+1}) \\
&\overset{(25)}{=} \gamma\theta_t g\left(\mathbf{z}_{t+1}\right) + \sum_{l=0}^{t} \rho_{t+1}^l g\left(\mathbf{z}_l\right) + f(A\boldsymbol{\alpha}_{t+1}) \\
&\leq \gamma\theta_t g\left(\mathbf{z}_{t+1}\right) + \sum_{l=0}^{t} \rho_{t+1}^l g\left(\mathbf{z}_l\right) \\
&\quad + \sum_{k=1}^{K} \Bigg\{ \frac{1}{K}(1-\gamma\theta_t) f(A\mathbf{y}_t) \\
&\quad + \gamma\theta_t \left\langle A^\top \nabla f(A\mathbf{y}_t), \mathbf{y}_t^{[k]} - \mathbf{z}_t^{[k]} \right\rangle \\
&\quad + \gamma\theta_t \Bigg(\frac{1}{K} f(A\mathbf{y}_t) + \left\langle A^\top \nabla f(A\mathbf{y}_t), \mathbf{z}_{t+1}^{[k]} - \mathbf{y}_t^{[k]} \right\rangle \\
&\qquad + \frac{\lambda\sigma'\theta_t}{2} \left\| A(\mathbf{z}_{t+1}^{[k]} - \mathbf{z}_t^{[k]}) \right\|^2 \Bigg) \Bigg\} \\
&\overset{(19),(26)}{=} \gamma\theta_t g\left(\mathbf{z}_{t+1}\right) + (1-\gamma\theta_t) \underbrace{\sum_{l=0}^{t} \rho_t^l g\left(\mathbf{z}_l\right)}_{\hat{g}^t} \\
&\quad + \sum_{k=1}^{K} \Bigg\{ \frac{1}{K}(1-\gamma\theta_t) f(A\mathbf{y}_t) \\
&\quad + (1-\gamma\theta_t) \left\langle A^\top \nabla f(A\mathbf{y}_t), \boldsymbol{\alpha}_t^{[k]} - \mathbf{y}_t^{[k]} \right\rangle \\
&\quad + \gamma\theta_t \Bigg(\frac{1}{K} f(A\mathbf{y}_t) + \left\langle A^\top \nabla f(A\mathbf{y}_t), \mathbf{z}_{t+1}^{[k]} - \mathbf{y}_t^{[k]} \right\rangle \\
&\qquad + \frac{\lambda\sigma'\theta_t}{2} \left\| A(\mathbf{z}_{t+1}^{[k]} - \mathbf{z}_t^{[k]}) \right\|^2 \Bigg) \Bigg\} \\
&\overset{(6)}{=} (1-\gamma\theta_t)\hat{g}^t + (1-\gamma\theta_t) \left(f(A\mathbf{y}_t) + \left\langle A^\top \nabla f(A\mathbf{y}_t), \boldsymbol{\alpha}_t - \mathbf{y}_t \right\rangle \right) \\
&\quad + \gamma\theta_t \sum_{k=1}^{K} \Bigg(\psi_k(\mathbf{z}_{t+1}^{[k]}) + \frac{1}{K} f(A\mathbf{y}_t) + \left\langle A^\top \nabla f(A\mathbf{y}_t), \mathbf{z}_{t+1}^{[k]} - \mathbf{y}_t^{[k]} \right\rangle \\
&\qquad + \frac{\lambda\sigma'\theta_t}{2} \|A(\mathbf{z}_{t+1}^{[k]} - \mathbf{z}_t^{[k]})\|^2 \Bigg)
\end{aligned}$$

$$
\begin{aligned}
&\overset{(8)}{\leq} (1-\gamma\theta_t)\left(\hat{g}^t + f(A\boldsymbol{\alpha}_t)\right) + \gamma\theta_t \sum_{k=1}^{K} \mathcal{G}_k(\mathbf{z}_{t+1}^{[k]}; \mathbf{y}_t, \mathbf{z}_t) \\
&\overset{(27)}{=} (1-\gamma\theta_t)\hat{\mathcal{O}_A}^t + \gamma\theta_t \sum_{k=1}^{K} \mathcal{G}_k(\mathbf{z}_{t+1}^{[k]}; \mathbf{y}_t, \mathbf{z}_t).
\end{aligned}
$$

Conditioning on the history up to time t, it follows from above that

$$
\begin{aligned}
\mathbf{E}_{t+1}[\hat{\mathcal{O}_A}^{t+1}|t] &\leq (1-\gamma\theta_t)\hat{\mathcal{O}_A}^t + \gamma\theta_t \sum_{k=1}^{K} \mathbf{E}_{t+1}[\mathcal{G}_k(\mathbf{z}_{t+1}^{[k]}; \mathbf{y}_t, \mathbf{z}_t)|t] \\
&\overset{(10)}{\leq} (1-\gamma\theta_t)\hat{\mathcal{O}_A}^t + \gamma\theta_t \sum_{k=1}^{K} \left(\mathcal{G}_k(\mathbf{z}_{t+1}^{\star\,[k]}; \mathbf{y}_t, \mathbf{z}_t) + \epsilon_t\right),
\end{aligned}
$$

meaning that, for any $\mathbf{u}$, one has

$$
\begin{aligned}
&\mathbf{E}_{t+1}[\hat{\mathcal{O}_A}^{t+1}|t] \\
&\overset{(21)}{\leq} (1-\gamma\theta_t)\hat{\mathcal{O}_A}^t + \gamma\theta_t \sum_{k=1}^{K} \left(\mathcal{G}_k(\mathbf{u}^{[k]}; \mathbf{y}_t, \mathbf{z}_t) - \frac{\lambda\theta_t\sigma'}{2}\left\|A(\mathbf{u}^{[k]} - \mathbf{z}_{t+1}^{\star\,[k]})\right\|^2 + \epsilon_t\right).
\end{aligned}
$$

In particular, choosing $\mathbf{u} = \boldsymbol{\alpha}_\star$, where $\boldsymbol{\alpha}_\star$ is any optimal solution of problem (A), and taking the total expectation, one finds

$$
\begin{aligned}
&\mathbf{E}[\hat{\mathcal{O}_A}^{t+1}] \\
&\leq (1-\gamma\theta_t)\,\mathbf{E}[\hat{\mathcal{O}_A}^t] + \gamma\theta_t \sum_{k=1}^{K} \mathbf{E}\left[\mathcal{G}_k(\boldsymbol{\alpha}_\star^{[k]}; \mathbf{y}_t, \mathbf{z}_t) - \frac{\lambda\theta_t\sigma'}{2}\left\|A(\boldsymbol{\alpha}_\star^{[k]} - \mathbf{z}_{t+1}^{\star\,[k]})\right\|^2 + \epsilon_t\right] \\
&= (1-\gamma\theta_t)\,\mathbf{E}[\hat{\mathcal{O}_A}^t] \\
&\qquad + \gamma\theta_t \sum_{k=1}^{K} \mathbf{E}\left[\frac{\lambda\theta_t\sigma'}{2}\|A(\boldsymbol{\alpha}_\star^{[k]} - \mathbf{z}_t^{[k]})\|^2 - \frac{\lambda\theta_t\sigma'}{2}\left\|A(\boldsymbol{\alpha}_\star^{[k]} - \mathbf{z}_{t+1}^{\star\,[k]})\right\|^2 + \epsilon_t\right] \\
&\qquad + \gamma\theta_t \sum_{k=1}^{K} \mathbf{E}\left[\psi_k(\boldsymbol{\alpha}_\star^{[k]}) + \frac{1}{K} f(Ay_t) + \langle A^\top \nabla f(A\mathbf{y}_t), \boldsymbol{\alpha}_\star^{[k]} - \mathbf{y}_t^{[k]}\rangle\right] \\
&\leq (1-\gamma\theta_t)\,\mathbf{E}[\hat{\mathcal{O}_A}^t] + \gamma\theta_t \mathcal{O}_A(\boldsymbol{\alpha}_\star) \\
&\qquad + \gamma\theta_t \sum_{k=1}^{K} \mathbf{E}\left[\frac{\lambda\theta_t\sigma'}{2}\|A(\boldsymbol{\alpha}_\star^{[k]} - \mathbf{z}_t^{[k]})\|^2 - \frac{\lambda\theta_t\sigma'}{2}\left\|A(\boldsymbol{\alpha}_\star^{[k]} - \mathbf{z}_{t+1}^{\star\,[k]})\right\|^2 + \epsilon_t\right],
\end{aligned}
$$

where the last inequality follows from convexity of f. Defining the scalar quantity $r_{t+1}^2 = \sum_{k=1}^{K} \|A(\boldsymbol{\alpha}_\star^{[k]} - \mathbf{z}_{t+1}^{\star\,[k]})\|^2$, we may conclude from above that

$$
\mathbf{E}\left[\hat{\mathcal{O}_A}^{t+1} - \mathcal{O}_A(\boldsymbol{\alpha}_\star) + \frac{\gamma\lambda\theta_t^2\sigma'}{2} r_{t+1}^2\right]
$$

$$\leq (1-\gamma\theta_t)\,\mathbf{E}[\hat{\mathcal{O}_A}^{\,t} - \mathcal{O}_A(\boldsymbol{\alpha}_\star)] + K\epsilon_t\gamma\theta_t + \frac{\gamma\lambda\theta_t^2\sigma'}{2}\,\mathbf{E}\left[\sum_{k=1}^{K}\|A(\boldsymbol{\alpha}_\star^{[k]} - \mathbf{z}_t^{[k]})\|^2\right]. \quad (28)$$

To bound the last term on the right-hand side, observe that, for all $k \in \{1,\dots,K\}$,

$$\begin{aligned}
\mathbf{E}\left[\|A(\boldsymbol{\alpha}_\star^{[k]} - \mathbf{z}_t^{[k]})\|^2\right] &= \mathbf{E}\left[\|A(\boldsymbol{\alpha}_\star^{[k]} - \mathbf{z}_t^{\star[k]} + \mathbf{z}_t^{\star[k]} - \mathbf{z}_t^{[k]})\|^2\right] \\
&\leq \mathbf{E}[\|A(\boldsymbol{\alpha}_\star^{[k]} - \mathbf{z}_t^{\star[k]})\|^2] + \underbrace{\mathbf{E}\left[\|A(\mathbf{z}_t^{\star[k]} - \mathbf{z}_t^{[k]})\|^2\right]}_{=:C_1} \\
&\quad + \underbrace{2\,\mathbf{E}\left[\|A(\boldsymbol{\alpha}_\star^{[k]} - \mathbf{z}_t^{\star[k]})\| \cdot \|A(\mathbf{z}_t^{\star[k]} - \mathbf{z}_t^{[k]})\|\right]}_{=:C_2} \\
&\leq \mathbf{E}[r_t^2] + C_1 + C_2.
\end{aligned}$$

It remains to bound C_1 and C_2. From (21) and (10), one finds that

$$\begin{aligned}
&\mathcal{G}_k({\mathbf{z}_{t+1}^\star}^{[k]}; \mathbf{y}_t, \mathbf{z}_t) + \frac{\lambda\theta_t\sigma'}{2}\,\mathbf{E}_{t+1}[\|A({\mathbf{z}_{t+1}^\star}^{[k]} - \mathbf{z}_{t+1}^{[k]})\|^2|t] \\
&\overset{(21)}{\leq} \mathbf{E}_{t+1}[\mathcal{G}_k(\mathbf{z}_{t+1}^{[k]}; \mathbf{y}_t, \mathbf{z}_t)|t] \\
&\overset{(10)}{\leq} \mathcal{G}_k({\mathbf{z}_{t+1}^\star}^{[k]}; \mathbf{y}_t, \mathbf{z}_t) + \epsilon_t,
\end{aligned} \quad (29)$$

and hence we can conclude that

$$\frac{\lambda\theta_t\sigma'}{2}\,\mathbf{E}_{t+1}[\|A({\mathbf{z}_{t+1}^\star}^{[k]} - \mathbf{z}_{t+1}^{[k]})\|^2|t] \leq \epsilon_t. \quad (30)$$

Therefore,

$$\begin{aligned}
\frac{\gamma\lambda\theta_t^2\sigma'}{2}C_1 &= \frac{\gamma\lambda\theta_t^2\sigma'}{2}\,\mathbf{E}\left[\|A(\mathbf{z}_t^{\star[k]} - \mathbf{z}_t^{[k]})\|^2\right] \\
&= \frac{\gamma\theta_t^2}{\theta_{t-1}}\frac{\lambda\theta_{t-1}\sigma'}{2}\,\mathbf{E}_{t-1}\left[\mathbf{E}_t\left[\|A(\mathbf{z}_t^{\star[k]} - \mathbf{z}_t^{[k]})\|^2|t-1\right]\right] \overset{(30)}{\leq} \frac{\gamma\theta_t^2}{\theta_{t-1}}\epsilon_{t-1}.
\end{aligned} \quad (31)$$

Now, let us bound C_2. By defining $\mathcal{R}$ as in (23c), Jensen's inequality gives

$$\begin{aligned}
\frac{\gamma\lambda\theta_t^2\sigma'}{2}C_2 &= \frac{\gamma\lambda\theta_t^2\sigma'}{2}\left(2\,\mathbf{E}\left[\|A(\boldsymbol{\alpha}_\star^{[k]} - \mathbf{z}_t^{\star[k]})\| \cdot \|A(\mathbf{z}_t^{\star[k]} - \mathbf{z}_t^{[k]})\|\right]\right) \\
&\overset{(23c)}{\leq} \mathcal{R}\gamma\lambda\theta_t^2\sigma'\,\mathbf{E}\left[\|A(\mathbf{z}_t^{\star[k]} - \mathbf{z}_t^{[k]})\|\right] \\
&\leq \mathcal{R}\gamma\lambda\theta_t^2\sigma'\sqrt{\frac{2}{\gamma\lambda\theta_t^2\sigma'}}\sqrt{\frac{\gamma\lambda\theta_t^2\sigma'}{2}\,\mathbf{E}\left[\|A(\mathbf{z}_t^{\star[k]} - \mathbf{z}_t^{[k]})\|^2\right]} \\
&\overset{(31)}{\leq} \mathcal{R}\gamma\lambda\theta_t^2\sigma'\sqrt{\frac{2}{\gamma\lambda\theta_t^2\sigma'}}\sqrt{\frac{\gamma\theta_t^2}{\theta_{t-1}}\epsilon_{t-1}} = \gamma\theta_t^2\mathcal{R}\sqrt{\frac{2\lambda\sigma'}{\theta_{t-1}}\epsilon_{t-1}}.
\end{aligned}$$

Putting everything together leads to

$$\mathbf{E}\left[\hat{\mathcal{O}_A}^{t+1} - \mathcal{O}_A(\boldsymbol{\alpha}_\star) + \frac{\gamma\lambda\theta_t^2\sigma'}{2}r_{t+1}^2\right] \overset{(28)}{\leq} \mathbf{E}\left[(1-\gamma\theta_t)(\hat{\mathcal{O}_A}^{t} - \mathcal{O}_A(\boldsymbol{\alpha}_\star)) + \frac{\gamma\lambda\theta_t^2\sigma'}{2}r_t^2\right] + K\epsilon_t\gamma\theta_t + \frac{\gamma\theta_t^2}{\theta_{t-1}}\epsilon_{t-1} + \gamma\theta_t^2\mathcal{R}\sqrt{\frac{2\lambda\sigma'}{\theta_{t-1}}\epsilon_{t-1}}.$$

Dividing both sides by θ_t^2 and denoting $\phi_t := \mathbf{E}[\hat{\mathcal{O}_A}^{t} - \mathcal{O}_A(\boldsymbol{\alpha}_\star)]$ and $\tilde{r}_t^2 := \mathbf{E}[r_t^2]$ gives

$$\frac{1}{\theta_t^2}\phi_{t+1} + \frac{\gamma\lambda\sigma'}{2}\tilde{r}_{t+1}^2 \overset{(28)}{\leq} \frac{1-\gamma\theta_t}{\theta_t^2}\phi_t + \frac{\gamma\lambda\sigma'}{2}\tilde{r}_t^2 + \underbrace{\frac{K\epsilon_t\gamma}{\theta_t} + \frac{\gamma}{\theta_{t-1}}\epsilon_{t-1} + \gamma\mathcal{R}\sqrt{\frac{2\lambda\sigma'}{\theta_{t-1}}\epsilon_{t-1}}}_{=:E_t}. \tag{32}$$

Now, by the property of θ_t in (13), one finds

$$\frac{1-\gamma\theta_{t+1}}{\theta_{t+1}^2}\phi_{t+1} + \frac{\gamma\lambda\sigma'}{2}\tilde{r}_{t+1}^2 \leq \frac{1-\gamma\theta_t}{\theta_t^2}\phi_t + \frac{\gamma\lambda\sigma'}{2}\tilde{r}_t^2 + E_t. \tag{33}$$

Unrolling the recurrence, one obtains for $t \geq 1$ that

$$\frac{1-\gamma\theta_t}{\theta_t^2}\phi_t + \frac{\gamma\lambda\sigma'}{2}\tilde{r}_t^2 \overset{(33)}{\leq} \frac{1-\gamma\theta_0}{\theta_0^2}\phi_0 + \frac{\gamma\lambda\sigma'}{2}\tilde{r}_0^2 + \sum_{i=0}^{t-1}E_i. \tag{34}$$

Hence, along with (13), one has for $t \geq 1$ that

$$\begin{aligned}\phi_t &\leq \theta_{t-1}^2\left(\frac{1-\gamma\theta_0}{\theta_0^2}\phi_0 + \frac{\gamma\lambda\sigma'}{2}\tilde{r}_0^2 + \sum_{i=0}^{t-1}E_i.\right)\\ &\overset{(14)}{\leq} \left(\frac{2}{t\gamma-\gamma+2}\right)^2\left(\frac{1-\gamma\theta_0}{\theta_0^2}\phi_0 + \frac{\gamma\lambda\sigma'}{2}\sum_{k=1}^{K}\|A(\boldsymbol{\alpha}_\star^{[k]} - \mathbf{z}_0^{[k]})\|^2 + K\epsilon_0\gamma + \sum_{i=1}^{t-1}E_i\right)\end{aligned}$$

and (22) follows.

The stated upper bound for the quantity $\mathcal{R}$, which measures the maximum possible distance between any two feasible solutions, can be derived as follows:

$$\begin{aligned}\mathcal{R} &\overset{(23c)}{:=} \max_{k\in\{1,2,\ldots,K\},a,b\in\{\alpha\in\mathbb{R}^n \mid \forall i:\ \alpha_i\in\mathrm{dom}(g_i)\}} \|A(a^{[k]} - b^{[k]})\|\\ &\leq \max_{k\in\{1,2,\ldots,K\},a,b\in\{\alpha\in\mathbb{R}^n \mid \forall i:\ \alpha_i\in\mathrm{dom}(g_i)\}} \sigma_k\|a^{[k]} - b^{[k]}\|\\ &\overset{(3)}{\leq} \max_{k\in\{1,2,\ldots,K\}} \sqrt{n_k}\sqrt{n_k 4L^2} = \max_{k\in\{1,2,\ldots,K\}} 2Ln_k,\end{aligned}$$

where $\sigma_k^2 = \max_{\boldsymbol{\alpha}^{[k]}} \frac{\|A\boldsymbol{\alpha}^{[k]}\|^2}{\|\boldsymbol{\alpha}^{[k]}\|^2} \leq n_k$. □

Table 1. Important quantities for our comparisons with related convergence results.

symbol	expression	worst-case upper bound	worst-case $\lim_{K\to n}$
$\tilde{\sigma}^2$	$\max_{\boldsymbol{\alpha}} \frac{\|A\boldsymbol{\alpha}\|^2}{\|\boldsymbol{\alpha}\|^2}$	n	n
σ_k^2	$\max_{\boldsymbol{\alpha}^{[k]}} \frac{\|A\boldsymbol{\alpha}^{[k]}\|^2}{\|\boldsymbol{\alpha}^{[k]}\|^2}$	$n_k = \frac{n}{K}$	1
σ^2	$\sum_{k=1}^K \|\mathcal{P}_k\|\sigma_k^2$	$\sum_k n_k^2 = K\frac{n^2}{K^2} = \frac{n^2}{K}$	n
$\mathcal{R}$	$\max_{k,a,b\in\mathrm{dom}(g)} \|A(a^{[k]} - b^{[k]})\|$	$\max_k 2Ln_k = \frac{2Ln}{K}$	$2L$

Theorem 3.4 describes the behavior of suboptimality only. We can, however, use the following theorem from [4], which relates suboptimality with the duality gap.

Theorem 3.5 ([4, Theorem 4])**.** *Suppose problem* (A) *is solved by a (possibly randomized) algorithm producing a sequence of iterates* $\{\boldsymbol{\alpha}_t\}_{t=0}^{\infty}$ *such that, for all* $t \geq 1$*,*

$$\mathbf{E}[\mathcal{O}_A(\boldsymbol{\alpha}_t) - \mathcal{O}_A(\boldsymbol{\alpha}_\star)] \leq \frac{F}{d(t)}$$

for some scalar $F \geq 0$ *and function* d*. If, for* $t \geq 1$*, it holds that*

$$d(t) \geq \max\left\{\frac{2F\lambda n}{\tilde{\sigma}^2 L^2}, \frac{2F\tilde{\sigma}^2 L^2}{\lambda n \epsilon^2}\right\}, \tag{35}$$

where $\tilde{\sigma}^2$ *is the maximum eigenvalue of* $A^\top A$*, then the expected duality gap satisfies*

$$\mathbf{E}[\mathcal{O}_A(\boldsymbol{\alpha}_t) + \mathcal{O}_B(\mathbf{w}(\boldsymbol{\alpha}_t))] \leq \epsilon.$$

Related to (35), henceforth, we assume that ϵ is such that $\frac{2F\lambda n}{\tilde{\sigma}^2 L^2} < \frac{2F\tilde{\sigma}^2 L^2}{\lambda n \epsilon^2}$.

Before stating our key corollaries of Theorem 3.4 and comparisons with results for other methods in the literature, let us define a few important quantities on which these results depend (see Table 1). The first quantity is $\tilde{\sigma}^2$, already defined in Theorem 3.5. Due to the fact that for each data column A_i we have $\|A_i\| \leq 1$, it follows that this quantity is bounded by n. The second quantity is σ_k^2, the maximum eigenvalue of a Gram matrix for the local data on node k. This value will be large if the samples stored on node k are correlated. The next important quantity is σ^2, which depends on each σ_k^2 and the size of each partition n_k.

3.1. Exact Subproblem Solutions

If the subproblems are solved exactly, i.e., if $\epsilon_t = 0$ for all $t \geq 0$, then (22) has the form

$$\begin{aligned}\mathbf{E}[\mathcal{O}_A(\boldsymbol{\alpha}_t) - \mathcal{O}_A(\boldsymbol{\alpha}_\star)] \leq & \frac{4}{(t\gamma-\gamma+2)^2}\Big((1-\gamma)(\mathcal{O}_A(\boldsymbol{\alpha}_0) - \mathcal{O}_A(\boldsymbol{\alpha}_\star)) \\ & + \frac{\gamma\lambda\sigma'}{2}\sum_{k=1}^K \|A(\boldsymbol{\alpha}_\star^{[k]} - \boldsymbol{\alpha}_0^{[k]})\|^2\Big).\end{aligned} \tag{36}$$

A nice property of this result is that the second term in the parentheses might be equal to zero even if $\mathbf{z}_0 = \boldsymbol{\alpha}_0 \neq \boldsymbol{\alpha}_\star$. This is not the case for other results for accelerated algorithms as their subproblems are strongly convex [1, 5, 7, 12] and hence have the

term $\|\boldsymbol{\alpha}_\star - \boldsymbol{\alpha}_0\|^2$ present in their complexity guarantees. Another nice property is that $\sum_{k=1}^K \|A(\boldsymbol{\alpha}_\star^{[k]} - \boldsymbol{\alpha}_0^{[k]})\|^2$ can be bounded above by $\sum_{k=1}^K \sigma_k^2 \|\boldsymbol{\alpha}_\star^{[k]} - \boldsymbol{\alpha}_0^{[k]}\|^2$, though the former can be much smaller.

Corollary 3.6. *Consider the extreme cases $\gamma = \frac{1}{K}$ and $\gamma = 1$ and assume that $n_k = \frac{n}{K}$. As discussed in Section 2, for the first case, one can choose $\sigma' = 1$ while for the second case one can choose $\sigma' = K$. To obtain $\mathbf{E}[\mathcal{O}_A(\boldsymbol{\alpha}_t) - \mathcal{O}_A(\boldsymbol{\alpha}_\star)] \leq \epsilon$, one has to run Algorithm 1 for at least $t > T$ iterations, where T is defined as follows for each case.*

- **Case** $\gamma = 1$*:*

$$\begin{aligned} T &= \sqrt{\tfrac{2\lambda\sigma'}{\epsilon}\textstyle\sum_{k=1}^K \|A(\boldsymbol{\alpha}_\star^{[k]} - \boldsymbol{\alpha}_0^{[k]})\|^2} \leq \sqrt{\tfrac{2}{\epsilon}}\sqrt{\lambda\sigma'\textstyle\sum_{k=1}^K \sigma_k^2 \|(\boldsymbol{\alpha}_\star^{[k]} - \boldsymbol{\alpha}_0^{[k]})\|^2} \\ &\leq \sqrt{\frac{2}{\epsilon}}\sqrt{\lambda\sigma'\textstyle\sum_{k=1}^K \sigma_k^2 4L^2 n_k} = \sqrt{\frac{8L^2}{\epsilon}}\sqrt{\lambda K \textstyle\sum_{k=1}^K \frac{n}{K} n_k} = \sqrt{\frac{8L^2 n^2 \lambda}{\epsilon}}. \end{aligned}$$

- **Case** $\gamma = 1/K$*:*

$$\begin{aligned} T &\leq K\sqrt{\tfrac{4}{\epsilon}\left((1-\tfrac{1}{K})(\mathcal{O}_A(\boldsymbol{\alpha}_0) - \mathcal{O}_A(\boldsymbol{\alpha}_\star)) + \tfrac{\gamma\lambda\sigma'}{2}\textstyle\sum_{k=1}^K \|A(\boldsymbol{\alpha}_\star^{[k]} - \boldsymbol{\alpha}_0^{[k]})\|^2\right)} \\ &\leq \sqrt{\tfrac{4K(K-1)}{\epsilon}(\mathcal{O}_A(\boldsymbol{\alpha}_0) - \mathcal{O}_A(\boldsymbol{\alpha}_\star)) + \tfrac{4}{\epsilon}\tfrac{K\lambda\sigma'}{2}\textstyle\sum_{k=1}^K \sigma_k^2 \|(\boldsymbol{\alpha}_\star^{[k]} - \boldsymbol{\alpha}_0^{[k]})\|^2} \\ &\leq \sqrt{\tfrac{4K(K-1)}{\epsilon}(\mathcal{O}_A(\boldsymbol{\alpha}_0) - \mathcal{O}_A(\boldsymbol{\alpha}_\star)) + \tfrac{8L^2 n^2 \lambda}{\epsilon}}. \end{aligned}$$

Comparison with CoCoA+. As found in [10], the complexity for $\gamma = 1$ is better as one requires fewer iterations in order to have an ϵ-approximate solution in expectation. Hence, let us focus only on the case of $\gamma = 1$ and compare our rate with the results derived in [10]. From the proof of [10, Theorem 8], one obtains that for CoCoA+, if $\boldsymbol{\alpha}_0 = \mathbf{0}$, it holds that $\mathbf{E}[\mathcal{O}_A(\boldsymbol{\alpha}_t) - \mathcal{O}_A(\boldsymbol{\alpha}_\star)] \leq \frac{4KL^2\lambda\sigma^2}{(1+\frac{1}{2}(t-t_0))}$, where

$$t_0 \geq \max\{0, \log(2(\mathcal{O}_A(\boldsymbol{\alpha}_0) - \mathcal{O}_A(\boldsymbol{\alpha}_\star))/(4KL^2\lambda\sigma))\}.$$

To obtain suboptimality below ϵ, CoCoA+ needs to be run for $t \geq T = t_0 + \frac{8KL^2\lambda\sigma^2}{\epsilon} \leq t_0 + \frac{8L^2n^2\lambda}{\epsilon}$ iterations. Neglecting the t_0 term, CoCoA+ needs $\frac{8L^2n^2\lambda}{\epsilon}$ iterations, whereas AccCoCoA+ needs only $\sqrt{\frac{8L^2n^2\lambda}{\epsilon}}$. This improvement is consistent with proximal gradient and accelerated gradient descent [11], which is as expected since, in the worst-case, they will produce the same iterates as CoCoA+ and AccCoCoA+.

Let us also derive a complexity bound for the duality gap. Corollary 3.6 with Theorem 3.5 implies that whenever $t \geq T$, where $T = \frac{2L}{\epsilon}\sqrt{\frac{\sigma'\tilde{\sigma}^2}{n}\sum_{k=1}^K \|A(\boldsymbol{\alpha}_\star^{[k]} - \boldsymbol{\alpha}_0^{[k]})\|^2} \sim \mathcal{O}(\frac{1}{\epsilon})$, the expected duality gap satisfies $\mathbf{E}[\mathcal{O}_A(\boldsymbol{\alpha}_t) + \mathcal{O}_B(\mathbf{w}(\boldsymbol{\alpha}_t))] \leq \epsilon$. This is valid for the number of iterations of AccCoCoA+. Note also that for the worst-case complexity for non-accelerated batch-SDCA (or full gradient method) for hinge loss ($L = 1$), the known results hold for the *average solution* and the number of iterations are $\sim \mathcal{O}(\frac{1}{\epsilon})$ (see, e.g., [19, 20] with the batch size chosen as large as possible).

3.2. Inexact Subproblem Solutions

To get a better understanding of the case when the subproblems are solved approximately, let us define an auxiliary nonnegative sequence $\{a_t\}_{t=0}^{\infty}$ such that $\frac{\sum_{t=0}^{\infty}\sqrt{a_t}}{t^2} \to 0$. In this section, we assume that the errors for the local solvers are set as $\epsilon_t = a_t\theta_t$.

Before analyzing this case in more detail, let us bound the total accumulated error up to iteration t, i.e., $\sum_{j=1}^{t-1} E_j$. One finds that

$$\begin{aligned}\sum_{j=1}^{t-1} E_j &= K\gamma\sum_{j=1}^{t-1} a_j + \sum_{j=0}^{t-2} a_j + \mathcal{R}\sqrt{2\lambda\sigma'}\sum_{j=1}^{t-1}\sqrt{a_{j-1}} \\ &\leq (K+1)\sum_{j=0}^{t-1} a_j + \mathcal{R}\sqrt{2\lambda\sigma'}\sum_{j=0}^{t-1}\sqrt{a_j} \\ &\leq (K+1)\sum_{j=0}^{t-1} a_j + \sqrt{\frac{8L^2}{\lambda K}}\sum_{j=0}^{t-1}\sqrt{a_j} =: S_t.\end{aligned}$$

Let us now consider two cases:

- Suppose that $a_t = r \in \mathbb{R}^+$. In this case, $\epsilon_t = r\theta_t = \mathcal{O}(1/t)$. Moreover, $S_t = \left((K+1)r + \sqrt{r\frac{8L^2}{\lambda K}}\right)t$. This implies that

$$\begin{aligned}\mathbf{E}[\mathcal{O}_A(\boldsymbol{\alpha}_t) - \mathcal{O}_A(\boldsymbol{\alpha}_\star)] \leq & 4\frac{\frac{\lambda\sigma'}{2}\sum_{k=1}^{K}\|A(\boldsymbol{\alpha}_\star^{[k]} - \boldsymbol{\alpha}_0^{[k]})\|^2 + K\epsilon_0 + S_t}{(t+1)^2} \\ & \sim \mathcal{O}\left(\frac{(K+1)r + \sqrt{r\frac{8L^2}{\lambda K}}}{t}\right).\end{aligned}$$

  Note that, in this case, $\epsilon \sim \mathcal{O}(1/t)$. This might create the impression that the local solver has to do more work as t increases; however, note that the Lipschitz constant of the gradient of the smooth part of the subproblem solved by the local solver also scales as $\theta_t \sim \mathcal{O}(1/t)$.
- A second interesting case is when $\lim_{t\to\infty} S_t =: S_\infty < \infty$. For example, suppose $a_t = \frac{r}{t^p}$ with $p > 2$. Then, indeed, $\lim_{t\to\infty} S_t$ is finite. In this case, one obtains

$$\begin{aligned}\mathbf{E}[\mathcal{O}_A(\boldsymbol{\alpha}_t) - \mathcal{O}_A(\boldsymbol{\alpha}_\star)] &\leq 4\frac{\frac{\lambda\sigma'}{2}\sum_{k=1}^{K}\|A(\boldsymbol{\alpha}_\star^{[k]} - \boldsymbol{\alpha}_0^{[k]})\|^2 + K\epsilon_0 + S_t}{(t+1)^2} \\ &\leq 4\frac{\frac{\lambda\sigma'}{2}\sum_{k=1}^{K}\|A(\boldsymbol{\alpha}_\star^{[k]} - \boldsymbol{\alpha}_0^{[k]})\|^2 + K\epsilon_0 + S_\infty}{(t+1)^2} \\ &\leq \frac{\frac{8L^2}{\lambda} + 4K\epsilon_0 + 4(K+1)\sum_{i=0}^{\infty} a_i + 4\sqrt{\frac{8L^2}{\lambda K}}\sum_{i=0}^{\infty}\sqrt{a_i}}{(t+1)^2} \sim \mathcal{O}(1/t^2).\end{aligned}$$

4. Numerical Experiments

In this section, we report the results of numerical experiments. The purpose of providing the results of these experiments is twofold. For one thing, we use them to illustrate the benefits of acceleration by providing results that compare the performance

of AccCoCoA+ versus CoCoA+ [10, 18]. In addition, we explore the communication/computation tradeoff and the scalability of AccCoCoA+. Here, it is important to recall that for AccCoCoA+ one communication is needed for each iteration. Hence, one reduces the communication costs by reducing the number of required iterations. For a recent comparison of CoCoA+ to other distributed solvers, including Quasi-Newton methods and ADMM, we refer the reader to [10, 18].

Our implementations of CoCoA+ and AccCoCoA+ are written in C++ using MPI for communication, run on m3.xlarge Amazon EC2 instances. We run all the experiments with $K = 4$ nodes using SDCA [16] as the local solver. The datasets used are summarized in Table 2.[1] For all runs, we chose σ' by the "safe rule," i.e., $\sigma' = K\gamma$.

Table 2. Datasets used in the numerical experiments.

Dataset	n	d	size(GB)
url	2,396,130	3,231,961	2.21
rcv1.test	677,399	47,236	1.21
covtype	581,012	54	0.07
epsilon	400,000	2,000	3.6

We first compare CoCoA+ versus AccCoCoA+ for solving hinge-loss SVM problems of the form (2). For both algorithms, the local solver, SDCA, is run for $H = 5 \times 10^5$ iterations (closed-form single coordinate solutions). In Figure 1, we compare the evolution of the duality gap with respect to the number of iterations and elapsed time. The results suggest a benefit of acceleration in terms of decreasing the duality gap, both when $\gamma = 1/K$ and when $\gamma = 1$. That said, the performance of AccCoCoA+ is not uniformly better than that of CoCoA+ for all iterations and at all time. Indeed, it is possible for CoCoA+ to "catch up" to AccCoCoA+, such as in the results for news20. That said, it is clear that AccCoCoA+ typically outperforms CoCoA+ (here and in the remainder of our experiments).

In Figure 2, we show how the regularization parameter λ_2 (indicated as λ) can affect the performance of both algorithms when solving the problem with the `url` dataset. In particular, the experiments suggest that as the value of λ_2 becomes smaller, there will be a more significant benefit from employing the accelerated algorithm.

[1] All data are available at `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`.

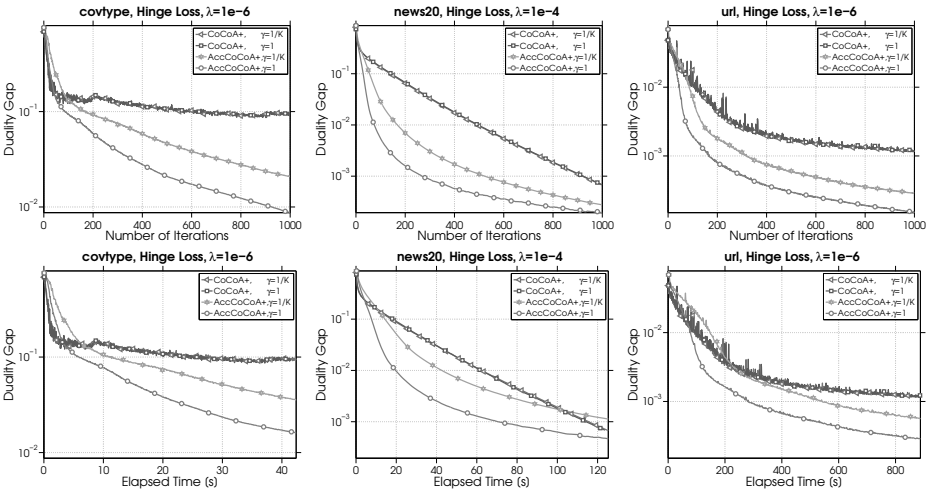

Figure 1. Duality gap as a function of iterations (top row) and elapsed time (bottom row) when solving hinge-loss SVM problems.

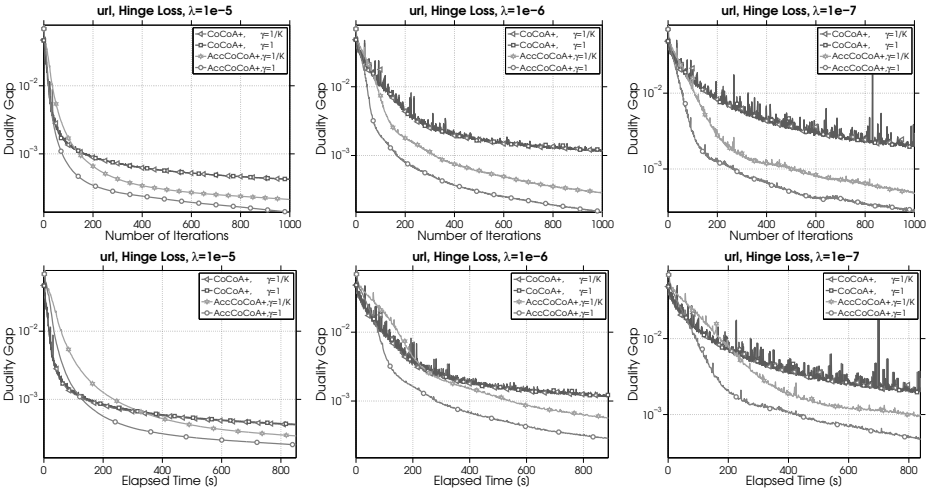

Figure 2. Duality gap as a function of iterations (top row) and elapsed time (bottom row) when solving hinge-loss SVM problems with different regularization values (λ_2) on the `url` dataset.

Figure 3 shows analogous results when the algorithms are employed to solve the Lasso problem in (1), for which increasing the regularization parameter λ_1 typically leads to more sparsity of the solution vector. The results indicate that the accelerated algorithm offers faster convergence of the sub-optimality gap to zero, especially for small values of λ_1.

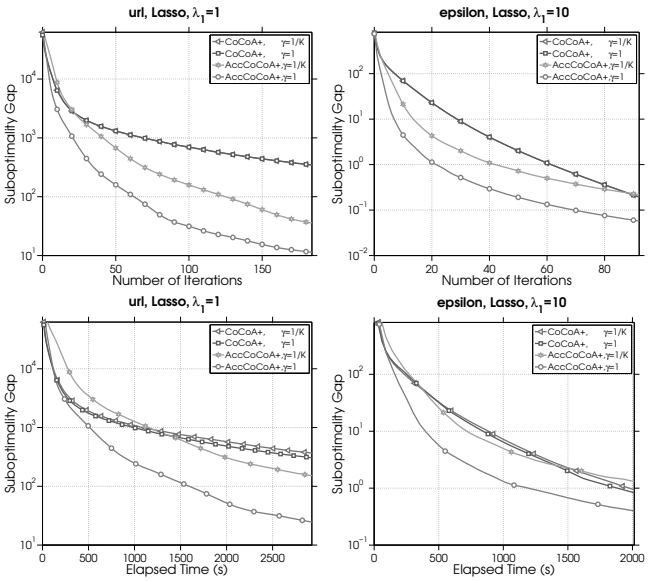

Figure 3. Sub-optimality gap as a function of iterations (top row) and elapsed time (bottom row) when solving Lasso problems. The choice of the regularizer value (λ_1) are such that for the *url* dataset the density of the optimal solution is 5.3%, while for the *epsilon* dataset the density is 13.56%.

We also ran experiments to demonstrate how the performance of AccCoCoA+ depends on the number of iterations (H) that SDCA runs for solving each subproblem. Figure 4 shows that for larger H the subproblems will be solved more accurately, and thus fewer outer iterations can be expected to reach a desired tolerance on the duality gap. However, in terms of running time, it is not always better to choice larger H. For example, for the `covtype` dataset, choosing $H = 10^4$ results in less time to reach a tolerance of 10^{-2} than is needed when $H = 10^3$ or $H = 10^5$. However, for the `news20` dataset, the time required decreases with H for all values considered in our experiments. The reason that this occurs is that d is quite large in this dataset, which makes each round of communication quite time consuming. Therefore, by solving the subproblems more accurately (by running more iterations), one achieves a better balance between communication and computation.

Such a tradeoff between communication and computation can also be observed in Figure 5, where we compare how the values of H can affect the convergence of the duality gap for the `covtype` and `rcv_test` datasets. The left two plots illustrate that to reach the same tolerance on the duality gap, the number of iterations can always be reduced by reaching a more accurate solution for each subproblem. However, the curves in the other two plots indicate that there is always a best value of H that leads to fastest convergence with respect to running time.

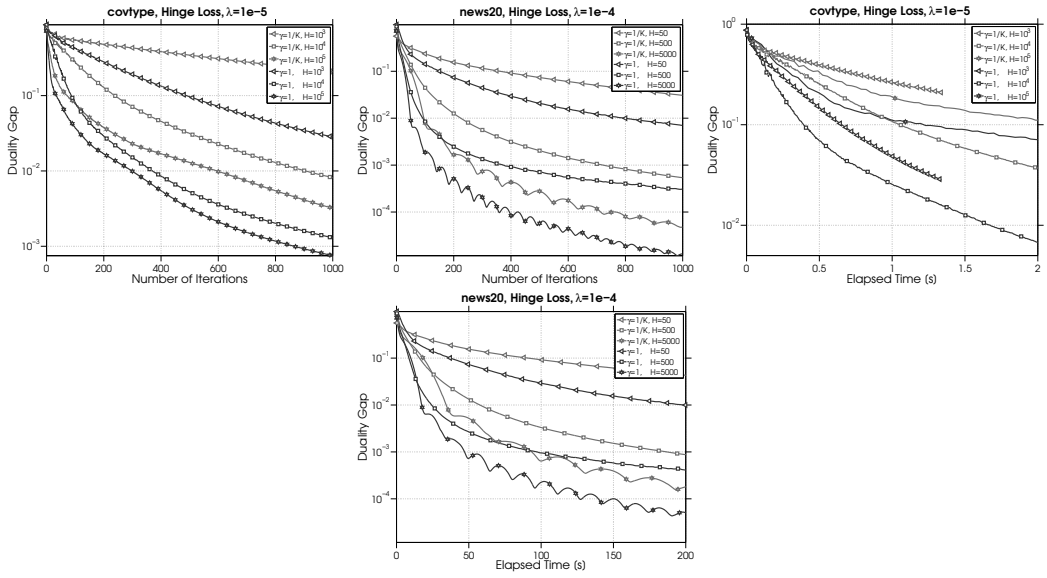

Figure 4. Comparison of performance for different inner iteration limits when $\gamma = \frac{1}{K}$ and $\gamma = 1$.

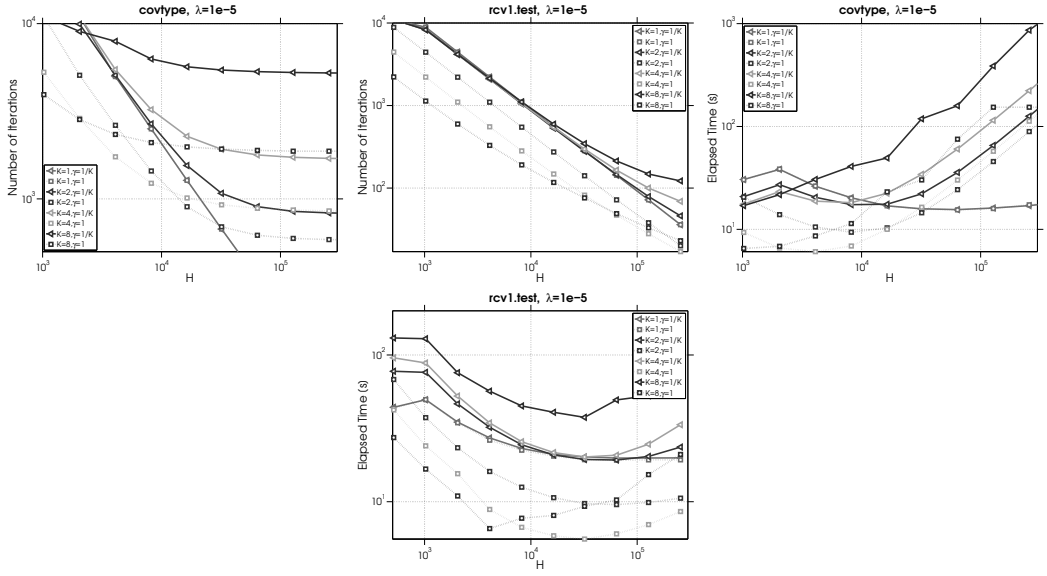

Figure 5. Number of iterations and running time required to reach a tolerance of 10^{-3} on the duality gap as the inner iteration limit (H) is varied.

Our last experiment shows how AccCoCoA+ scales with the number of machines (K). The results are shown in Figure 6. We set $H = \frac{n}{K}$ for every K to make sure that the same amount of data is utilized during each iteration of AccCoCoA+ across all K machines. By doing so, it is also expected that each subproblem has been solved to similar accuracy, according to the complexity result of SDCA in [16]. The results show that when $\gamma = 1$, AccCoCoA+ takes almost the same amount of time regardless of how many machines are used, which demonstrates the better scaling properties it has than the $\gamma = \frac{1}{K}$ case.

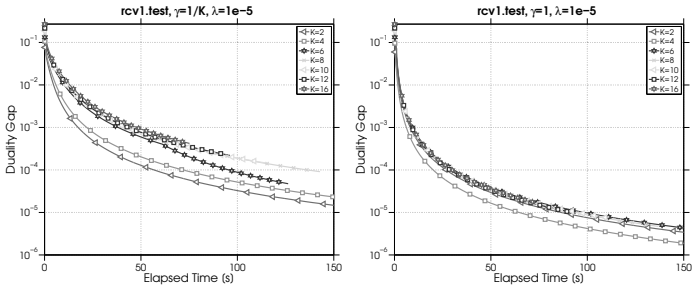

Figure 6. Running times on different numbers of machines K.

5. Conclusion

We proposed and analyzed AccCoCoA+, an accelerated variant of CoCoA+ achieving the optimal $\mathcal{O}(1/t^2)$ convergence rate. The method is robust to inaccurate subproblems both in theory and practice, and performs well in large-scale experiments. Our analysis provides constants in the convergence rate which are significantly tighter compared to those previously obtained for CoCoA+.

Funding

This material is based upon work supported by the U.S. National Science Foundation, Division of Computing and Communications Foundations, under Award Number CCF-1618717 and CCF:1740796.

References

[1] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

[2] Joseph K Bradley, Aapo Kyrola, Danny Bickson, and Carlos Guestrin. Parallel coordinate descent for l1-regularized loss minimization. *arXiv:1105.5379*, 2011.

[3] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

[4] Celestine Dünner, Simone Forte, Martin Takáč, and Martin Jaggi. Primal-dual rates and certificates. *In 33rd International Conference on Machine Learning, ICML 2016*, 2016.

[5] Olivier Fercoq and Peter Richtárik. Accelerated, parallel and proximal coordinate descent. *SIAM Journal on Optimization*, 25(4):1997–2023, 2015.

[6] Martin Jaggi, Virginia Smith, Martin Takác, Jonathan Terhorst, Sanjay Krishnan, Thomas Hofmann, and Michael I Jordan. Communication-efficient distributed dual coordinate ascent. In *Advances in Neural Information Processing Systems*, pages 3068–3076, 2014.

[7] Hongzhou Lin, Julien Mairal, and Zaid Harchaoui. A universal catalyst for first-order optimization. In *Advances in Neural Information Processing Systems*, pages 3366–3374, 2015.

[8] Ji Liu and Stephen J Wright. Asynchronous stochastic coordinate descent: Parallelism and convergence properties. *SIAM Journal on Optimization*, 25(1):351–376, 2015.

[9] Chenxin Ma, Jakub Konečnỳ, Martin Jaggi, Virginia Smith, Michael I Jordan, Peter Richtárik, and Martin Takáč. Distributed optimization with arbitrary local solvers. *Optimization Methods and Software*, 32(4):813–848, 2017.

[10] Chenxin Ma, Virginia Smith, Martin Jaggi, Michael I Jordan, Peter Richtárik, and Martin Takáč. Adding vs. averaging in distributed primal-dual optimization. *In 32th International Conference on Machine Learning, ICML 2015*, 2015.

[11] Yu Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, 2013.

[12] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

[13] Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 693–701, 2011.

[14] R Tyrrell Rockafellar. Convex analysis princeton university press. *Princeton, NJ*, 1970.

[15] Shai Shalev-Shwartz and Tong Zhang. Accelerated mini-batch stochastic dual coordinate ascent. In *Advances in Neural Information Processing Systems*, pages 378–385, 2013.

[16] Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *The Journal of Machine Learning Research*, 14(1):567–599, 2013.

[17] Ohad Shamir, Nati Srebro, and Tong Zhang. Communication-efficient distributed optimization using an approximate newton-type method. pages 1000–1008, 2014.

[18] Virginia Smith, Simone Forte, Ma Chenxin, Martin Takáč, Michael I Jordan, and Martin Jaggi. COCOA: A general framework for communication-efficient distributed optimization. *Journal of Machine Learning Research*, 18:230, 2018.

[19] Martin Takáč, Avleen Bijral, Peter Richtárik, and Nathan Srebro. Mini-batch primal and dual methods for svms. In *In 30th International Conference on Machine Learning, ICML 2013*, 2013.

[20] Martin Takáč, Peter Richtárik, and Nathan Srebro. Distributed mini-batch SDCA. *arXiv:1507.08322*, 2015.

[21] Tianbao Yang, Shenghuo Zhu, Rong Jin, and Yuanqing Lin. Analysis of distributed stochastic dual coordinate ascent. *arXiv:1312.1031*, 2013.

[22] Yuchen Zhang and Lin Xiao. Communication-efficient distributed optimization of self-concordant empirical loss. In *Large-Scale and Distributed Optimization*, pages 289–341. Springer, 2018.

[23] Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, 23(4):550–560, 1997.