A Block-wise, Asynchronous and Distributed ADMM Algorithm for General Form Consensus Optimization

Rui Zhu\(^1\), Di Niu\(^1\), Zongpeng Li\(^2\)
\(^1\) Department of Electrical and Computer Engineering, University of Alberta
\(^2\) School of Computer Science, Wuhan University
\{rzhu3, dniu\}@ualberta.ca, zongpeng@whu.edu.cn

Abstract

Many machine learning models, including those with non-smooth regularizers, can be formulated as consensus optimization problems, which can be solved by the alternating direction method of multipliers (ADMM). Many recent efforts have been made to develop asynchronous distributed ADMM to handle large amounts of training data. However, all existing asynchronous distributed ADMM methods are based on full model updates and require locking all global model parameters to handle concurrency, which essentially serializes the updates from different workers. In this paper, we present a novel block-wise, asynchronous and distributed ADMM algorithm, which allows different blocks of model parameters to be updated in parallel. The lock-free block-wise algorithm may greatly speedup sparse optimization problems, a common scenario in reality, in which most model updates only modify a subset of all decision variables. We theoretically prove the convergence of our proposed algorithm to stationary points for non-convex general form consensus problems with possibly non-smooth regularizers. We implement the proposed ADMM algorithm on the Parameter Server framework and demonstrate its convergence and near-linear speedup performance as the number of workers increases.

1 Introduction

The need to scale up machine learning in the presence of sheer volume of data has spurred recent interest in developing efficient distributed optimization algorithms. Distributed machine learning jobs often involve solving a non-convex, decomposable, and regularized optimization problem of the following form:

\[
\min_{\mathbf{x}} \sum_{i=1}^{N} f_i(x_1, \ldots, x_M) + \sum_{j=1}^{M} h_j(x_j),
\]

\[\text{s.t. } x_j \in \mathcal{X}_j, j = 1, \ldots, M\]

where each \(f_i : \mathcal{X} \rightarrow \mathbb{R}\) is a smooth but possibly non-convex function, fitting the model \(\mathbf{x} := (x_1, \ldots, x_M)\) to local training data available on node \(i\); each \(\mathcal{X}_j\) is a closed, convex, and compact set; and the regularizer \(h(x) := \sum_{j=1}^{M} h_j(x_j)\) is a separable, convex but possibly non-smooth regularization term to prevent overfitting. Example problems of this type can be found in deep learning with regularization [Dean et al., 2012; Chen et al., 2015], robust matrix completion [Niu et al., 2011; LASSO [Tibshirani et al., 2005], sparse logistic regression [Liu et al., 2009], and sparse support vector machine (SVM) [Friedman et al., 2001].

To date, a number of efficient asynchronous and distributed stochastic gradient descent (SGD) algorithms, e.g., [Niu et al., 2011] [Lian et al., 2015] [Li et al., 2014a], have been proposed, in which each worker node asynchronously updates its local model or gradients based on its local dataset, and sends them to the server(s) for model updates or aggregation. Yet, SGD is not particularly suitable for solving optimization problems with non-smooth objectives or with constraints, which are prevalent in practical machine learning adopting regularization, e.g., [Liu et al., 2009]. Distributed (synchronous) ADMM [Boyd et al., 2011; Zhang and Kwok, 2014] Chang et al., 2016a; Chang et al., 2016b] Hong, 2017 [Wei and Ozdaglar, 2013; Mota et al., 2013; Taylor et al., 2016] has been widely studied as an alternative method, which avoids the common pitfalls of SGD for highly non-convex problems, such as saturation effects, poor conditioning, and saddle points [Taylor et al., 2016]. The original idea on distributed ADMM can be found in [Boyd et al., 2011], which is essentially a synchronous algorithm. In this work, we focus on studying the asynchronous distributed alternating direction method of multipliers (ADMM) for non-convex non-smooth optimization.

Asynchronous distributed ADMM has been actively discussed in recent literature. Zhang and Kwok [2014] consider an asynchronous ADMM assuming bounded delay, which enables each worker node to update a local copy of the model parameters asynchronously without waiting for other workers to complete their work, while a single server is responsible for driving the local copies of model parameters to approach the global consensus variables. They provide proof of convergence for convex objective functions only. Wei and Ozdaglar [2013] assume that communication links between nodes can fail randomly, and propose an ADMM scheme that converges almost surely to a saddle point. Chang et al. [2016a; 2016b] propose an asynchronous ADMM algorithm with
exactly work requires each worker to solve a subproblem exactly, which is often costly in practice. Hong [2017] proposes another asynchronous ADMM algorithm, where each worker only computes the gradients based on local data, while all model parameter updates happen at a single server, a possible bottleneck in large clusters.

To our knowledge, all existing work on asynchronous distributed ADMM requires locking global consensus variables at the (single) server for each model update; although asynchrony is allowed among workers, i.e., workers are allowed to be at different iterations of model updating. Such atomic or memory-locking operations essentially serialize model updates contributed by different workers, which may seriously limit the algorithm scalability. In many practical examples, not all workers need to access all model parameters. For example, in recommender systems, a local dataset of user-item interactions is only associated with a specific set of users (and items), and therefore does not need to access the latent variables of other users (or items). In text categorization, each document usually consists of a subset of words or terms in corpus, and each worker only needs to deal with the words in its own local corpus.

A distributed machine learning algorithm is illustrated in Fig. 1. There are multiple server nodes, each known as a “PS” and stores a subset (block) of model parameters (consensus variables) \( z \). There are also multiple worker nodes; each worker owns a local dataset, and has a loss function \( f_i \) depending on one or several blocks of model parameters, but not necessarily all of them. If there is only one server node, the architecture in Fig. 1 degenerates to a “star” topology with a single master, which has been adopted by Spark [Zaharia et al., 2010]. With multiple servers, the system is also called a Parameter Server architecture [Dean et al., 2012; Li et al., 2014a] and has been adopted by many large scale machine learning systems, including TensorFlow [Abadi et al., 2016] and MXNet [Chen et al., 2015].

It is worth noting that enabling block-wise updates in ADMM is critical for training large models, such as sparse logistic regression, robust matrix completion, etc., since not all worker nodes will need to work on all model parameters — each worker only needs to work on the blocks of parameters pertaining to its local dataset. For these reasons, block-wise updates have been extensively studied for a number of gradient type of distributed optimization algorithms, including SGD [Lian et al., 2015], proximal gradient descent [Li et al., 2014b], block or stochastic coordinate descent (BCD or SCD) [Liu and Wright, 2015], as well as for a recently proposed block successive upper bound minimization method (BSUM) [Hong et al., 2016b].

In this work, we propose the first block-wise asynchronous distributed ADMM algorithm that can increase efficiency over existing single-server ADMM algorithms, by better exploiting the parallelization opportunity in model parameter updates. Specifically, we introduce the general form consensus optimization problem [Boyd et al., 2011], and solve it in a block-wise asynchronous fashion, thus making ADMM amenable for implementation on Parameter Server, with multiple servers hosting model parameters. In our algorithm, each worker only needs to work on one or multiple blocks of parameters that are relevant to its local data, while different blocks of model parameters can be updated in parallel asynchronously subject to a bounded delay. Since this scheme does not require locking all the decision variables together, it belongs to the set of lock-free optimization algorithms (e.g., HOGWILD! [Niu et al., 2011] as a lock-free version of SGD) in the literature. Our scheme is also useful on shared memory systems, such as on a single machine with multi-cores or multiple GPUs, where enforcing atomicity on all the consensus variables is inefficient.

Theoretically, we prove that, for general non-convex objective functions, our scheme can converge to stationary points. Experimental results on a cluster of 36 CPU cores have demonstrated the convergence and near-linear speedup of the proposed ADMM algorithm, for training sparse logistic regression models based on a large real-world dataset.

## 2 Preliminaries

### 2.1 Consensus Optimization and ADMM

The minimization in (1) can be reformulated into a global variable consensus optimization problem [Boyd et al., 2011]:

\[
\min_{z, (x_i) \in X} \sum_{i=1}^{N} f_i(x_i) + h(z), \quad (2a)
\]

\[
s.t. \quad x_i = z, \quad \forall i = 1, \ldots, N. \quad (2b)
\]

where \( z \) is often called the global consensus variable, traditionally stored on a master node, and \( x_i \) is its local copy updated and stored on one of \( N \) worker nodes. The function \( h \) is decomposable. It has been shown [Boyd et al., 2011] that such a problem can be efficiently solved using distributed (synchronous) ADMM. In particular, let \( y_i \) denote the Lagrange dual variable associated with each constraint in (2b) and define the Augmented Lagrangian as

\[
L(X, Y, z) = \sum_{i=1}^{N} f_i(x_i) + h(z) + \sum_{i=1}^{N} \langle y_i, x_i - z \rangle + \sum_{i=1}^{N} \frac{\rho_i}{2} \| x_i - z \|^2, \quad (3)
\]

where \( X := (x_1, \ldots, x_N) \) represents a juxtaposed matrix of all \( x_i \), and \( Y \) represents the juxtaposed matrix of all \( y_i \). We...
have, for (synchronized) rounds \( t = 0, 1, \ldots \), the following variable updating equations:

\[
\begin{align*}
x_{i}^{t+1} & = \arg \min_{x_{i} \in X} \left\{ f_{i}(x_{i}) + \langle y_{i}, x_{i} - z_{i}^{t} \rangle + \frac{\rho_{i}}{2} \| x_{i} - z_{i}^{t} \|^{2} \right\}, \\
y_{i}^{t+1} & = y_{i}^{t} + \rho_{i}(x_{i}^{t+1} - z_{i}^{t}), \\
z_{i}^{t+1} & = \arg \min_{z_{i} \in X} \left\{ h(z_{i}) + \sum_{i=1}^{N} \langle y_{i}, x_{i}^{t+1} - z_{i} \rangle + \sum_{i=1}^{N} \frac{\rho_{i}}{2} \| x_{i}^{t+1} - z_{i} \|^{2} \right\}.
\end{align*}
\]

### 2.2 General Form Consensus Optimization

Many machine learning problems involve highly sparse models, in the sense that each local dataset on a worker is only associated with a few model parameters, i.e., each \( f_{i} \) only depends on a subset of the elements in \( x \). The global consensus optimization problem in (4), however, ignores such sparsity, since in each round each worker \( i \) must push the entire vectors \( x_{i}, y_{i} \), and to the master node to update \( z \). In fact, this is the setting of all recent work on asynchronous distributed ADMM, e.g., [Zhang and Kwok, 2014]. In this case, when multiple workers attempt to update the global consensus variable \( z \), the vectors \( x_{i} \) and \( y_{i} \) must also be updated sequentially, which leads to diminishing efficiency as the number of workers \( N \) increases.

To better exploit model sparsity in practice for further parallelization opportunities between workers, we propose the general form consensus optimization problem [Boyd et al., 2011]. Specifically, with \( N \) worker nodes and \( M \) server nodes, the vectors \( x_{i}, y_{i} \) and \( z \) can all be decomposed into \( M \) blocks. Let \( z_{j} \) denote the \( j \)-th block of the global consensus variable \( z \), located on server \( j \), for \( j = 1, \ldots, M \). Similarly, let \( x_{i,j} \) denote the corresponding \( j \)-th block of the local variable \( x_{i} \) on worker \( i \). Let \( E \) be all the \((i,j)\) pairs such that \( f_{i} \) depends on the block \( x_{i,j} \) (and correspondingly depends on \( z_{j} \)). Furthermore, let \( N(j) = \{i|(i,j) \in E\} \) denote the set of all the neighboring workers of server \( j \). Similarly, let \( N(i) = \{j|(i,j) \in E\} \).

Then, the general form consensus problem [Boyd et al., 2011] is described as follows:

\[
\begin{align*}
\min_{z_{j},(x_{i,j})} & \sum_{i=1}^{N} f_{i}(\{x_{i,j}\}_{j=1}^{M}) + h(z), \\
\text{s.t.} & \quad x_{i,j} = z_{j}, \quad \forall(i,j) \in E, \\
& \quad x_{i,j}, z_{j} \in X_{j}.
\end{align*}
\]

In fact, in \( f_{i}(\{x_{i,j}\}_{j=1}^{M}) \), a block \( x_{i,j} \) will only be relevant if \((i,j) \in E\), and will be a dummy variable otherwise, whose value does not matter. Yet, since the sparse dependencies of \( f_{i} \) on the blocks \( j \) can be captured through the specific form of \( f_{i} \), here we have included all \( M \) blocks in each \( f_{i} \)’s arguments just to simplify the notation.

The structure of problem (4) can effectively capture the sparsity inherent to many practical machine learning problems. Since each \( f_{i} \) only depends on a few blocks, the formulation in (4) essentially reduces the number of decision variables—it does not matter what value \( x_{i,j} \) will take for any \((i,j) \notin E\). For example, when training a topic model for documents, the feature of each document is represented as a bag of words, and hence only a subset of all words in the vocabulary will be active in each document’s feature. In this case, the constraint \( x_{i,j} = z_{j} \) only accounts for those words \( j \) that appear in the document \( i \), and therefore only those words \( j \) that appeared in document \( i \) should be optimized. Like (3), we also define the Augmented Lagrangian \( L(X,Y,Z) \) as follows:

\[
L(X,Y,Z) = \sum_{i=1}^{N} f_{i}(\{x_{i,j}\}_{j=1}^{M}) + h(z) + \sum_{(i,j) \in E} \langle y_{i,j}, x_{i,j} - z_{j} \rangle + \sum_{(i,j) \in E} \frac{\rho_{i}}{2} \| x_{i,j} - z_{j} \|^{2}.
\]

The formulation in (4) perfectly aligns with the latest Parameter Server architecture as shown in Fig. 1. Here we can let each server node maintain one model block \( z_{j} \), such that worker \( i \) updates \( z_{j} \) if and only if \((i,j) \in E\). Since all three vectors \( x_{i}, y_{i} \) and \( z \) in (4) are decomposable into blocks, to achieve a higher efficiency, we will investigate block-wise algorithms which not only enable different workers to send their updates asynchronously to the server (like prior work on asynchronous ADMM does), but also enable different model blocks \( z_{j} \) to be updated in parallel and asynchronously on different servers, removing the locking or atomicity assumption required for updating the entire \( z \).

### 3 A Block-wise, Asynchronous, and Distributed ADMM Algorithm

In this section, we present our proposed block-wise, asynchronous and distributed ADMM algorithm (a.k.a, AsyBADMM) for the general consensus problem. For ease of presentation, we first describe a synchronous version motivated by the basic distributed ADMM for non-convex optimization problems as a starting point.

#### 3.1 Block-wise Synchronous ADMM

The update rules presented in Sec. 2.2 represent the basic synchronous distributed ADMM approach [Boyd et al., 2011]. To solve the general form consensus problem, our block-wise version extends such a synchronous algorithm mainly by 1) approximating the update rule of \( x_{i} \) with a simpler expression under non-convex objective functions, and 2) converting the all-vector updates of variables into block-wise updates only for \((i,j) \in E\).

Generally speaking, in each synchronized epoch \( t \), each worker node \( i \) updates all blocks of its local primal variables \( x_{i,j} \) and dual variables \( y_{i,j} \) for \( j \in N(i) \), and pushes these updates to the corresponding servers. Each server \( j \), when it has received \( x_{i,j} \) and \( y_{i,j} \) from all \( i \in N(j) \), will update \( z_{j} \) accordingly, by aggregating these received blocks.

Specifically, at epoch \( t \), the basic synchronous distributed ADMM will do the following update for \( x_{i} \):

\[
x_{i}^{t+1} = \arg \min_{x_{i} \in X_{i}} \left\{ f_{i}(x_{i}) + \sum_{j \in N(i)} \langle y_{i,j}, x_{i,j} - z_{j} \rangle + \sum_{j \in N(i)} \frac{\rho_{i}}{2} \| x_{i,j} - z_{j} \|^{2} \right\}.
\]

1To simplify notations, we still use \( X \) and \( Y \) as previously defined, but entries \((i,j) \notin E \) are not taken into account.
However, this subproblem is hard, especially when $f_i$ is non-convex. To handle non-convex objectives, we adopt an alternative solution \cite{Hong_2017, Hong_2016a} to this subproblem through the following first-order approximation of $f_i$ at $z^t$:

$$x_i^{t+1} = \arg\min_{x_i} f_i(z^t) + \langle \nabla f_i(z^t), x_i - z^t \rangle + \sum_{j \in \mathcal{N}(i)} \left( \langle y_{i,j}^t, x_{i,j} - z_j^t \rangle + \frac{\rho_i}{2} \|x_{i,j} - z_j^t\|^2 \right)$$

$$= z^t - \nabla f_i(z^t) + y_i^t, \quad \gamma = \frac{1}{\rho_i}, \tag{5}$$

where (5) can be readily obtained by setting the partial derivative w.r.t. $x_i$ to zero.

The above full-vector update on $x_i$ is equivalent to the following block-wise updates on each block $x_{i,j}$ by worker $i$:

$$x_{i,j}^{t+1} = z_j^t - \frac{\nabla_j f_i(z^t) + y_{i,j}^t}{\rho_i}, \quad \gamma = \frac{1}{\rho_i}$$

where $\nabla_j f_i(z^t)$ is the partial derivative of $f_i$ w.r.t. $z_j$. Furthermore, the dual variable blocks $y_{i,j}$ can also be updated in a block-wise fashion as follows:

$$y_{i,j}^{t+1} = y_{i,j}^t + \rho_i(x_{i,j}^{t+1} - z_j^t). \quad (7)$$

Note that in fact, each $f_i$ only depends on a part of $z^t$ and thus each worker $i$ only needs to pull the relevant blocks $z_j$ for $j \in \mathcal{N}(i)$. Again, we put the full vector $z$ in $f_i$ just to simplify notation.

On the server side, worker $j$ will update $z_j$ based on the newly updated $x_{i,j}^{t+1}$, $y_{i,j}^{t+1}$ received from all workers $i$ such that $i \in \mathcal{N}(j)$. Again, the $z$ update in the basic synchronous distributed ADMM can be rewritten into the following block-wise format (with a regularization term introduced):

$$z_j^{t+1} = \arg\min_{z_j \in X_j} h_j(z_j) + \frac{\gamma}{2} \|z_j - z_j^t\|^2$$

$$+ \sum_{i \in \mathcal{N}(j)} \left( \langle y_{i,j}^{t+1}, x_{i,j}^{t+1} - z_j \rangle + \frac{\rho_i}{2} \|x_{i,j}^{t+1} - z_j\|^2 \right)$$

$$= \text{prox}_{\mu} \left( \frac{\gamma z_j^t + \sum_{i \in \mathcal{N}(j)} w_{i,j}^t}{\gamma + \sum_{i \in \mathcal{N}(j)} \rho_i} \right), \quad \gamma = \frac{1}{\rho_i}, \tag{8}$$

where $w_{i,j}^{t+1}$ is defined as

$$w_{i,j}^{t+1} := \rho_i x_{i,j}^{t+1} + y_{i,j}^{t+1}, \quad \gamma = \frac{1}{\rho_i}$$

and the proximal operator is defined as

$$\text{prox}_{\mu}(x) := \arg\min_{u \in X_j} h(u) + \frac{\mu}{2} \|x - u\|^2. \quad \gamma = \frac{1}{\rho_i} \tag{10}$$

Furthermore, the regularization term $\gamma \|x_j - z_j^t\|^2$ is introduced to stabilize the results, which will be helpful in the asynchronous case.

In the update of $z_j$, the constant $\mu$ of the proximal operator is given by $\sum_{i \in \mathcal{N}(j)} \rho_i$. Now it is clear that it is sufficient for worker $i$ to send $w_{i,j}^{t+1}$ to server $j$ in epoch $t$.

### 3.2 Block-wise Asynchronous ADMM

We now take one step further to present a block-wise asynchronous distributed ADMM algorithm, which is our main contribution in this paper. In the asynchronous algorithm, each worker $i$ will use a local epoch $t$ to keep track of how many times $x_i$ has been updated, although different workers may be in different epochs, due to random delays in computation and communication.

Let us first focus on a particular worker $i$. While worker $i$ is in epoch $t$, there is no guarantee for worker $i$ to download $z^t$—different blocks $z_j$ in $z$ may have been updated for different numbers of times, for which worker $i$ has no idea. Therefore, we use $\tilde{z}_j^t$ to denote the latest copy of $z_j$ on server $j$ while worker $i$ is in epoch $t$ and $\hat{z}_j^t = (\tilde{z}_j^1, \ldots, \tilde{z}_j^M)$. Then, the original synchronous updating equations (6) and (7) for $x_{i,j}$ and $y_{i,j}$, respectively, are simply replaced by

$$x_{i,j}^{t+1} = \tilde{z}_j^t - \frac{\nabla_j f_i(z^t) + y_{i,j}^t}{\rho_i}, \quad \gamma = \frac{1}{\rho_i}, \tag{11}$$

$$y_{i,j}^{t+1} = \tilde{z}_j^t + \rho_i(x_{i,j}^{t+1} - \tilde{z}_j^t). \quad (12)$$

Now let us focus on the server side. Since in the asynchronous case, the variables $w_{i,j}^{t+1}$ for different workers $i$ do not generally arrive at the server $j$ at the same time. In this case, we will update $z_j^{t+1}$ incrementally as soon as a $w_{i,j}^{t+1}$ is received from some worker $i$ until all $w_{i,j}^{t+1}$ are received for all $i \in \mathcal{N}(j)$, at which point the update for $z_j^{t+1}$ is fully finished.

We use $\hat{z}_j^{t+1}$ to denote the working (dirty) copy of $z_j^{t+1}$ for which the update may not be fully finished by all workers yet. Then, the update of $\hat{z}_j^{t+1}$ is given by

$$\tilde{z}_j^{t+1} = \text{prox}_{\mu} \left( \frac{\gamma \hat{z}_j^t + \sum_{i \in \mathcal{N}(j)} \tilde{w}_{i,j}}{\gamma + \sum_{i \in \mathcal{N}(j)} \rho_i} \right), \tag{13}$$

where $\tilde{w}_{i,j} = w_{i,j}^t$ if $w_{i,j}^t$ is received from worker $i$ and triggering the above update; and for all other $i$, $\tilde{w}_{i,j}$ is the latest version of $w_{i,j}$ that server $j$ holds for worker $i$. The regularization coefficient $\gamma > 0$ helps to stabilize convergence in the asynchronous execution with random delays.

Algorithm \cite{Feng_2016} describes the entire block-wise asynchronous distributed ADMM. Note that in Algorithm \cite{Feng_2016}, the block $j$ is randomly and independently selected from $\mathcal{N}(i)$ according to a uniform distribution, which is common in practice. Due to the page limit, we only consider the random block selection scheme, and we refer readers to other options including Gauss-Seidel and Gauss-Southwell block selection in the literature, e.g., \cite{Hong_2016a}.

We put a few remarks on implementation issues at the end of this section to characterize key features of our proposed block-wise asynchronous algorithm, which differs from full-vector updates in the literature \cite{Hong_2017}. Firstly, model parameters are stored in blocks, so different workers can update different blocks asynchronously in parallel, which takes advantage of the popular Parameter Server architecture. Secondly, workers can pull $z$ while others are updating some blocks, enhancing concurrency. Thirdly, in our implementation, workers will compute both gradients and local variables.
Algorithm 1 AsyBADMM: Block-wise Asynchronous ADMM

Each worker $i$ asynchronously performs:

1: pull $z_i^0$ to initialize $x_i^0 = z_i^0$
2: initialize $y_i^0$ as the zero vector.
3: for $i = 0$ to $T - 1$ do
4: select an index $j_i \in \mathcal{N}(i)$ uniformly at random
5: compute gradient $\nabla y_i^j f(x_i^t)$
6: update $y_i^{t+1}$ by (11) and (12)
7: push $w_i^{t+1}$ as defined by (9) to server $j_i$
8: pull the current models $\hat{z}^{t+1}$ from servers
9: end for

Each server $j$ asynchronously performs:

1: initialize $z_j^0$ and $\hat{w}_{i,j}$ for all $i \in \mathcal{N}(j)$
2: Upon receiving $w_i^{t+1}$ from a worker $i$:
3: let $\hat{w}_{i,j} \leftarrow w_i^{t+1}$
4: update $z_j^{t+1}$ by (13)
5: if $w_i^t$ has been received for all $i \in \mathcal{N}(j)$ then $z_j^{t+1} \leftarrow z_j^{t+1}$.

In contrast, in the full-vector ADMM [Hong, 2017], workers are only responsible for computing gradients, therefore all previously computed and transmitted $\hat{w}_{i,j}$ must be cached on servers with non-negligible memory overhead.

4 Convergence Analysis

In this section, we provide convergence analysis of our algorithm under certain standard assumptions:

Assumption 1 (Block Lipschitz Continuity). For all $(i,j) \in \mathcal{E}$, there exists a positive constant $L_{i,j} > 0$ such that

$$\|\nabla f_i(x) - \nabla f_i(z)\| \leq L_{i,j} \|x - z\|, \forall x, z \in \mathbb{R}^d.$$  

Assumption 2 (Bounded Below). Each function $f_i(x)$ is bounded below, i.e., there exists a finite number $f > -\infty$ where $f$ denotes the optimal objective value of problem (4).

Assumption 3 (Bounded Delay). The total delay of each link $(i,j) \in \mathcal{E}$ is bounded with the constant of $T_{i,j}$ for each pair of worker $i$ and server $j$. Formally, there is an integer $0 \leq \tau \leq T_{i,j}$, such that $\hat{z}^i_j = z_i^{t-\tau}$ for all $t \geq 0$. This should also hold for $\hat{w}_{i,j}$.

To characterize the convergence behavior, a commonly used metric is the squared norm of gradients. Due to potential nonsmoothness of $h(\cdot)$, Hong et al. propose a new metric [2016a] as a hybrid of gradient mapping as well as vanilla gradient as follows:

$$P(X^t, Y^t, Z^t) := \|z^t - \hat{z}^t\|^2 + \sum_{(i,j) \in \mathcal{E}} \|\nabla x_{i,j} L(X^t, Y^t, Z^t)\|^2$$

$$+ \sum_{(i,j) \in \mathcal{E}} \|x_{i,j} - z_j\|^2,$$  

(14)

where $\hat{z}^t = (\hat{z}_1^t, \ldots, \hat{z}_M^t)$ is defined as

$$\hat{z}_j^t := \text{prox}_{\rho_j}(z_j^t - \nabla z_j L(X^t, Y^t, z^t) - h(z^t)).$$  

(15)

It is clear that if $P(X^t, Y^t, z^t) \to 0$, then we obtain a stationary solution of (1).

The following Theorem [1] indicates that Algorithm 1 converges to a stationary point satisfying KKT conditions under suitable choices of hyper-parameters.

Theorem 1. Suppose that Assumptions [2][3] hold. Moreover, for all $i$ and $j$, the penalty parameter $\rho_i$ and $\gamma$ are chosen to be sufficiently large such that:

$$\infty > L(X^0, Y^0, 0) - f \geq 0$$  

(16)

$$\alpha_j := \gamma + \rho_i - \sum_{i \in \mathcal{N}(j)} \left(\frac{1}{2} + \frac{1}{\rho_i}\right) L_{i,j}^2 (T_{i,j} + 1)^2$$

$$- \sum_{i \in \mathcal{N}(j)} (4L_{i,j} + \rho_i + 1) T_{i,j}^2 > 0,$$  

(17)

$$\beta_i := \rho_i - 4 \max_{j \in \mathcal{N}(i)} L_{i,j} > 0.$$  

(18)

Then the following is true for Algorithm 7:

1. Algorithm 7 converges in the following sense:

$$\lim_{t \to \infty} \|x_{i,j}^{t+1} - x_{i,j}^t\| = 0, \quad \forall j = 1, \ldots, M,$$  

(19a)

$$\lim_{t \to \infty} \|y_{i,j}^{t+1} - y_{i,j}^t\| = 0, \quad \forall (i,j) \in \mathcal{E},$$  

(19b)

$$\lim_{t \to \infty} \|y_{i,j}^{t+1} - y_{i,j}^*\| = 0, \quad \forall (i,j) \in \mathcal{E}.$$  

(19c)

2. For each worker $i$ and server $j$, denote the limit points of $\{x_{i,j}^t\}, \{y_{i,j}^t\}$, and $\{z_j^t\}$ by $\bar{x}_{i,j}$, $\bar{y}_{i,j}$, and $\bar{z}_j$, respectively. Then these limit points satisfy KKT conditions, i.e., we have

$$\nabla f_i(x_i^*) + y_i^* = 0, \quad \forall (i,j) \in \mathcal{E},$$  

(20a)

$$\sum_{j \in \mathcal{N}(i)} y_{i,j}^* \in \partial h_j(z_j^*), \quad \forall j = 1, \ldots, M,$$  

(20b)

$$x_{i,j}^* = z_j^* \in \mathcal{X}_j, \quad \forall (i,j) \in \mathcal{E}.$$  

(20c)

When sets $\mathcal{X}_j$ are compact, the sequence of iterates generated by Algorithm 7 converges to stationary points.

3. For some $\epsilon > 0$, let $T(\epsilon)$ denote the epoch that achieves the following:

$$T(\epsilon) = \min\{t | P(X^t, Y^t, z^t) \leq \epsilon, t \geq 0\}.$$  

Then there exists some constant $C > 0$ such that

$$T(\epsilon) \leq \frac{C(L(X^0, Y^0, z^0) - f)}{\epsilon},$$  

(21)

where $f$ is defined in Assumption 2.

Due to the non-convex objective function $f_i(x_i)$, no guarantee of global optimality is possible in general. The parameter $\rho_i$ acts like the learning rate hyper-parameter in gradient descent: a large $\rho_i$ slows down the convergence and a smaller one can speed it up. The term $\gamma$ is associated with the delay bound $T_{i,j}$. In the synchronous case, we can set $\gamma = 0$; otherwise, to guarantee convergence, $\gamma$ should be increased as the maximum allowable delay $T_{i,j}$ increases.
5 Experiments

We now show how our algorithm can be used to solve the challenging non-convex non-smooth problems in machine learning. We will show how AsyBADMM exhibits a near-linear speedup as the number of workers increases. We use a cluster of 18 instances of type c4.large on Amazon EC2. This type of instances has 2 CPU cores and at least 3.75 GB RAM, running 64-bit Ubuntu 16.04 LTS (HVM). Each server and worker process uses up to 2 cores. In total, our deployment uses 36 CPU cores and 67.5 GB RAM. Two machines serve as server nodes, while the other 16 machines serve as worker nodes. Note that we treat one core as a computational node (either a server or worker node).

**Setup:** In this experiment, we consider the sparse logistic regression problem:

\[
\min_x \quad \frac{1}{m} \sum_{i=1}^{m} \log(1 + \exp(-\tilde{y}_i \langle \tilde{x}_i, x \rangle)) + \lambda \|x\|_1
\]

(22)

where the constant \(C\) is used to clip out some extremely large values for robustness. The \(\ell_1\)-regularized logistic regression is one of the most popular algorithms used for large scale risk minimization. We consider a public sparse text dataset KDDa. This dataset has more than 8 million samples, 20 million features, and 305 million nonzero entries. To show the advantage of parallelism, we set up five experiments with 1, 4, 8, 16 and 32 nodes, respectively. In each experiment, the whole dataset will be evenly split into several smaller parts, and each node only has access to its local dataset.

We implement our algorithm on the ps-lite framework [Li et al., 2014a], which is a lightweight implementation of Parameter Server architecture. It supports Parameter Server for multiple devices in a single machine, and multiple machines in a cluster. This is the back end of kvstore API of the deep learning framework MXNet [Chen et al., 2015]. Each worker updates the blocks by cycling through the coordinates of \(x\) and updating each in turns, restarting at a random coordinate after each cycle.

**Results:** Empirically, Assumption 3 is observed to hold for this cluster. We set the hyper-parameter \(\gamma = 0.01\) and the clip threshold constant as \(C = 10^4\), and the penalty parameter \(\rho_{i,j} = 100\) for all \((i, j)\). Fig. 2(a) and Fig. 2(b) show the convergence behavior of our proposed algorithm in terms of objective function values. From the figures, we can clearly observe the convergence of our proposed algorithm. This observation confirms that asynchrony with tolerable delay can still lead to convergence.

To further analyze the parallelism in AsyBADMM, we measure the speedup by the relative time for \(p\) workers to perform \(k\) iterations, i.e., Speedup of \(p\) workers = \(\frac{T_{1}(p)}{T_{k}(p)}\), where \(T_{k}(p)\) is the time it takes for \(p\) workers to perform \(k\) iterations of optimization. Fig. 2(b) illustrates the running time comparison and Table 1 shows that AsyBADMM actually achieves near-linear speedup.

| Workers \(p\) | \(k = 20\) | \(k = 50\) | \(k = 100\) | Speedup |
|-------------|----------|----------|-----------|--------|
| 1           | 1404     | 3688     | 6802      | 1.0    |
| 4           | 363      | 952      | 1758      | 3.87   |
| 8           | 177      | 466      | 859       | 7.92   |
| 16          | 86       | 226      | 417       | 16.31  |
| 32          | 47       | 124      | 228       | 29.83  |

6 Concluding Remarks

In this paper, we propose a block-wise, asynchronous and distributed ADMM algorithm to solve general non-convex and non-smooth optimization problems in machine learning. Under the bounded delay assumption, we have shown that our proposed algorithm can converge to stationary points satisfying KKT conditions. The block-wise updating nature of our algorithm makes it feasible to be implemented on Parameter Server, take advantage of the ability to update different blocks of all model parameters in parallel on distributed servers. Experimental results based on a real-world dataset have demonstrated the convergence and near-linear speedup of the proposed ADMM algorithm, for training large-scale sparse logistic regression models in Amazon EC2 clusters.

---

[http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/](http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/)
References

[Abadi et al., 2016] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andrew Davis, Jeffrey Dean, et al. Tensorflow: A system for large-scale machine learning. In 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), pages 265–283. USENIX Association, 2016.

[Boyd et al., 2011] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends® in Machine Learning, 3(1):1–122, 2011.

[Chang et al., 2016a] Tsung-Hui Chang, Mingyi Hong, Wei-Cheng Liao, and Xiangfeng Wang. Asynchronous distributed admm for large-scale optimization – part i: Algorithm and convergence analysis. IEEE Transactions on Signal Processing, 64(12):3118–3130, 2016.

[Chang et al., 2016b] Tsung-Hui Chang, Wei-Cheng Liao, Mingyi Hong, and Xiangfeng Wang. Asynchronous distributed admm for large-scale optimization – part ii: Linear convergence analysis and numerical performance. IEEE Transactions on Signal Processing, 64(12):3131–3144, 2016.

[Chen et al., 2015] Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. arXiv preprint arXiv:1512.01274, 2015.

[Dean et al., 2012] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al. Large scale distributed deep networks. In Advances in neural information processing systems, pages 1223–1231, 2012.

[Friedman et al., 2001] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. The elements of statistical learning, volume 1. Springer series in statistics Springer, Berlin, 2001.

[Hong et al., 2016a] Mingyi Hong, Zhi-Quan Luo, and Meisam Razaviyayn. Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems. SIAM Journal on Optimization, 26(1):337–364, 2016.

[Hong et al., 2016b] Mingyi Hong, Meisam Razaviyayn, Zhi-Quan Luo, and Jong-Shi Pang. A unified algorithmic framework for block-structured optimization involving big data: With applications in machine learning and signal processing. IEEE Signal Processing Magazine, 33(1):57–77, 2016.

[Hong, 2017] Mingyi Hong. A distributed, asynchronous and incremental algorithm for nonconvex optimization: An admm approach. IEEE Transactions on Control of Network Systems, PP(99):1–1, 2017.

[Li et al., 2014a] Mu Li, David G Andersen, Jun Woo Park, Alexander J Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J Shekita, and Bor-Yiing Su. Scaling distributed machine learning with the parameter server. In 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14), pages 583–598, 2014.

[Li et al., 2014b] Mu Li, David G Andersen, Alexander J Smola, and Kai Yu. Communication efficient distributed machine learning with the parameter server. In Advances in Neural Information Processing Systems, pages 19–27, 2014.

[Lian et al., 2015] Xiangru Lian, Yijun Huang, Yuncheng Li, and Ji Liu. Asynchronous parallel stochastic gradient for nonconvex optimization. In Advances in Neural Information Processing Systems, pages 2737–2745, 2015.

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

[Liu et al., 2009] Jun Liu, Jianhui Chen, and Jieping Ye. Large-scale sparse logistic regression. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 547–556. ACM, 2009.

[Mota et al., 2013] João FC Mota, João MF Xavier, Pedro MQ Aguiar, and Markus Puschel. D-admm: A communication-efficient distributed algorithm for separable optimization. IEEE Transactions on Signal Processing, 61(10):2718–2723, 2013.

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

[Taylor et al., 2016] Gavin Taylor, Ryan Burmeister, Zheng Xu, Bharat Singh, Ankit Patel, and Tom Goldstein. Training neural networks without gradients: A scalable admm approach. In International Conference on Machine Learning, pages 2722–2731, 2016.

[Tibshirani et al., 2005] Robert Tibshirani, Michael Saunders, Saharon Rosset, Ji Zhu, and Keith Knight. Sparcity and smoothness via the fused lasso. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 67(1):91–108, 2005.

[Wei and Ozdaglar, 2013] Ermin Wei and Asuman Ozdaglar. On the o(1/k) convergence of asynchronous distributed alternating direction method of multipliers. arXiv preprint arXiv:1307.8254, 2013.

[Zaharia et al., 2010] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark: cluster computing with working sets. HotCloud, 10:10–10, 2010.

[Zhang and Kwok, 2014] Ruiliang Zhang and James Kwok. Asynchronous distributed admm for consensus optimization. In International Conference on Machine Learning, pages 1701–1709, 2014.
A  Key Lemmas

Lemma 1. Suppose Assumptions 1-3 are satisfied. Then we have
\[ \| y_{i,j}^{t+1} - y_{i,j}^t \|^2 \leq L_{i,j}^2 (T_{i,j} + 1) \sum_{t'=0}^{T_{i,j}} \| z_{i,j}^{t+1-t'} - z_{i,j}^{t-t'} \|^2. \] (23)

Proof. For simplicity, we say \((i,j)\) is performed at epoch \(t\) when worker \(i\) is updating block \(j\) at epoch \(t\). If the updating \((i,j)\) is not performed at epoch \(t\), this inequality holds in trivial, as \(y_{i,j}^{t+1} = y_{i,j}^t\). So we only consider the case that \((i,j)\) is performed at epoch \(t\). Note that in this case, we have
\[ \nabla_j f_i(\tilde{z}^{t+1}) + y_{i,j}^t + \rho_i (x_{i,j}^{t+1} - z_{i,j}^{t+1}) = 0. \] (24)

Since \(y_{i,j}^{t+1} = y_{i,j}^t + \rho_i (x_{i,j}^{t+1} - z_{i,j}^{t+1})\), we have
\[ \nabla_j f_i(\tilde{z}^{t+1}) + y_{i,j}^{t+1} = 0. \] (25)

Therefore, we have
\[ \| y_{i,j}^{t+1} - y_{i,j}^t \| = \| \nabla_j f_i(\tilde{z}^{t+1}) - \nabla_j f_i(\tilde{z}^t) \| \leq L_{i,j} \| z_{i,j}^{t+1} - z_{i,j}^t \|. \]

Since the actual updating time for \(z_{i,j}^{t+1}\) should be in \(\{t + 1, \ldots, t + 1 - T_{i,j}\}\), and for \(z_{i,j}^t\) in \(\{t, \ldots, t - T_{i,j}\}\), then we have
\[ \| y_{i,j}^{t+1} - y_{i,j}^t \|^2 \leq \sum_{t=0}^{T_{i,j}} \| z_{i,j}^{t+1-t'} - z_{i,j}^{t-t'} \|^2, \] (26)

which proves the lemma. \(\Box\)

Lemma 2. At epoch \(t\), we have
\[ \nabla_j f_i(\tilde{z}^{t+1}) + y_{i,j}^t + \rho_i (x_{i,j}^{t+1} - z_{i,j}^{t+1}) = \rho_i (\tilde{z}_{i,j}^{t+1} - z_{i,j}^{t+1}). \] (27)

Proof. Updating \(x_{i,j}^{t+1}\) is performed as follows
\[ x_{i,j}^{t+1} = \arg \min_{x_{i,j}} f_i(\tilde{z}^{t+1}) + \langle \nabla_j f_i(\tilde{z}^{t+1}), x_{i,j} - z_{i,j}^{t+1} \rangle + \langle y_{i,j}^t, x_{i,j} - z_{i,j}^{t+1} \rangle + \frac{\rho_i}{2} \| x_{i,j} - z_{i,j}^{t+1} \|^2. \]

Thus, we have
\[ \nabla_j f_i(\tilde{z}^{t+1}) + y_{i,j}^t + \rho_i (x_{i,j}^{t+1} - z_{i,j}^{t+1}) = 0, \]

And therefore
\[ \nabla_j f_i(\tilde{z}^{t+1}) + y_{i,j}^t + \rho_i (x_{i,j}^{t+1} - z_{i,j}^{t+1}) = \rho_i (\tilde{z}_{i,j}^{t+1} - z_{i,j}^{t+1}). \] \(\Box\)

Lemma 3. Suppose Assumptions 1-3 are satisfied. Then we have
\[ L(X^t, Y^t, Z^t) - L(X^0, Y^0, Z^0) \leq - \sum_{t=0}^{T-1} \sum_{(i,j) \in E} \beta_i (\| x_{i,j}^{t+1} - z_{i,j}^{t+1} \|^2 + \| x_{i,j}^{t} - z_{i,j}^{t+1} \|^2) - \sum_{t=0}^{T-1} \sum_{(i,j) \in E} \alpha_j \| z_{i,j}^{t+1} - z_{i,j}^t \|^2, \] (28)

where
\[ \alpha_j := (\gamma + \rho_i) - \sum_{i \in N(j)} \left( \frac{1}{\rho_i} + \frac{1}{2} \right) L_{i,j}^2 (T_{i,j} + 1)^2 - \sum_{i \in N(j)} \frac{(4L_{i,j} + \rho_i + 1)T_{i,j}^2}{2}, \] (30)
\[ \beta_i := \max_{j \in N(i)} \frac{4L_{i,j}}{2|N(i)|}. \] (31)

Lemma 4. Suppose that Assumptions 1-3 hold. Then the sequence of solutions \(\{X^t, Z^t, Y^t\}\) satisfies
\[ \lim_{t \to \infty} L(X^t, Z^t, Y^t) \geq f - \text{diam}^2(\mathcal{X}) \sum_{(i,j) \in E} \frac{L_{i,j}}{2} > -\infty. \] (32)
Next we try to bound the gap between two consecutive Augmented Lagrangian values and break it down into three steps, namely, updating $X$, $Y$ and $Z$:

$$
L(X^{t+1}, Y^{t+1}, Z^{t+1}) - L(X^t, Y^t, Z^t) = L(X^t, Y^t, Z^{t+1}) - L(X^t, Y^t, Z^t) + L(X^{t+1}, Y^t, Z^{t+1}) - L(X^{t+1}, Y^t, Z^t)
$$  \( (33) \)

To prove Lemma 3, we bound the above three gaps individually. Firstly, we bound the on $X$. For each worker $i$, at epoch $t$, we use the following auxiliary function for convergence analysis:

$$
l_i(x_i, y_{i,j}, z_j) := f_i(x_i) + (y_{i,j}, x_i - z_j) + \frac{\rho_i}{2} \|x_i - z_j\|^2 \quad (34)$$

To simplify our proof in this section, we consider the case that only one block is updated. Therefore, only block $j$ in $x_i^{t+1}$ differs from $x_i^t$, and similarly for $y_j^{t+1}$ and $z_j^{t+1}$. We will use $\tilde{z}^t := z^{t(i,j)}$ as the delayed version of $Z$ in this proof.

**Lemma 5.** For node $i$, we have the following inequality to bound the gap after updating $x_{i,j}$:

$$
l_i(x_i^{t+1}, y_{i,j}, z_j^{t+1}) - l_i(x_i^t, y_{i,j}, z_j^{t+1}) \leq \frac{L_{ij,i}}{2} \|x_i^{t+1} - z_i^{t+1}\|^2 + \frac{L_{ij,i}}{2} \|\tilde{z}_j^{t+1} - z_j^{t+1}\|^2 + \frac{L_{ij,j}}{2} \|x_i^{t+1} - x_i^{t}\|^2 - \frac{\rho_i}{2} \|x_i^{t+1} - z_j^{t+1}\|^2 + \|x_i^{t+1} - z_j^{t+1}\|^2.
$$  \( (35) \)

**Proof.** From the block Lipschitz assumption and the updating rule, we have

$$
f_i(x_i^{t+1}) \leq f_i(x_i^t) + \langle \nabla_j f_i(x_i^t), x_i^{t+1} - x_i^t \rangle + \frac{L_{ij,i}}{2} \|x_i^{t+1} - x_i^t\|^2.
$$  \( (36) \)

By the definition of $l_i(\cdot)$, we have

$$
l_i(x_i^{t+1}, y_{i,j}, z_j^{t+1}) \leq l_i(x_i^t, y_{i,j}, z_j^{t+1}) + \langle \nabla_j f_i(x_i^t), x_i^{t+1} - x_i^t \rangle + \frac{L_{ij,i}}{2} \|x_i^{t+1} - x_i^t\|^2 + \langle y_j^t, x_i^{t+1} - x_i^t \rangle + \frac{\rho_i}{2} \|x_i^{t+1} - z_j^{t+1}\|^2 - \frac{\rho_i}{2} \|x_i^{t+1} - z_j^{t+1}\|^2.
$$  \( (37) \)

The right hand side is actually a quadratic function w.r.t. $x_i^{t+1}$. Therefore, it is strongly convex, and we have

$$
l_i(x_i^{t+1}, y_{i,j}, z_j^{t+1}) \leq l_i(x_i^t, y_{i,j}, z_j^{t+1}) + \frac{L_{ij,i}}{2} \|x_i^{t+1} - x_i^t\|^2 + \frac{\rho_i}{2} \|\tilde{z}_j^{t+1} - z_j^{t+1}\|^2 + \|x_i^{t+1} - z_j^{t+1}\|^2 - \frac{\rho_i}{2} \|x_i^{t+1} - z_j^{t+1}\|^2
$$  \( (38) \)

By Lemma 2, we have

$$\nabla_j f_i(x_i^{t+1}) + y_j^t + \rho_i(x_i^{t+1} - z_j^{t+1}) = \nabla_j f_i(\tilde{z}_j^{t+1}) + y_j^t + \rho_i(x_i^{t+1} - z_j^{t+1}) + (\nabla_j f_i(x_i^{t+1}) - \nabla_j f_i(\tilde{z}_j^{t+1})) = \rho_i(z_j^{t+1} - z_j^{t+1}) + (\nabla_j f_i(x_i^{t+1}) - \nabla_j f_i(\tilde{z}_j^{t+1})).$$
Therefore, we have
\[
\begin{align*}
& l_i(x_{i,t}^{t+1}, y_{i,j}, z_j^{t+1}) - l_i(x_{i,t}^{t+1}, y_{i,j}, z_j^{t+1}) + \langle \nabla_j f_i(x_{i,t}^{t+1}) - \nabla_j f_i(z_j^{t+1}), z_j^{t+1} - z_j^{t+1} \rangle \\
& + \frac{L_{i,j}}{2} \|x_{i,j}^{t+1} - x_{i,j}^t\|^2 - \frac{\rho_i}{2} \|x_{i,j}^{t+1} - z_j^{t+1}\|^2 - \frac{\rho_i}{2} \|x_{i,j}^{t+1} - z_j^{t+1}\|^2 + \frac{\rho_i}{2} \|z_j^{t+1} - z_j^{t+1}\|^2 \\
& \leq l_i(x_{i,t}^{t+1}, y_{i,j}, z_j^{t+1}) + \frac{L_{i,j}}{2} \|x_{i,j}^{t+1} - z_j^{t+1}\|^2 + \frac{L_{i,j} + \rho_i}{2} \|z_j^{t+1} - z_j^{t+1}\|^2 + \frac{L_{i,j}}{2} \|x_{i,j}^{t+1} - x_{i,j}^t\|^2 \\
& - \frac{\rho_i}{2} \left( \|x_{i,j}^{t+1} - z_j^{t+1}\|^2 + \|x_{i,j}^{t+1} - z_j^{t+1}\|^2 \right) \\
& \leq l_i(x_{i,t}^{t+1}, y_{i,j}, z_j^{t+1}) + \frac{4L_{i,j}}{2} \|x_{i,j}^{t+1} - z_j^{t+1}\|^2 + \frac{4L_{i,j} + \rho_i}{2} \|z_j^{t+1} - z_j^{t+1}\|^2 + \frac{3L_{i,j}}{2} \|x_{i,j}^{t+1} - x_{i,j}^{t+1}\|^2 \\
& - \frac{\rho_i}{2} \left( \|x_{i,j}^{t+1} - z_j^{t+1}\|^2 + \|x_{i,j}^{t+1} - z_j^{t+1}\|^2 \right) \\
& \leq l_i(x_{i,t}^{t+1}, y_{i,j}, z_j^{t+1}) + \left( \frac{\rho_i}{2} - \frac{4L_{i,j}}{2} \right) \|x_{i,j}^{t+1} - z_j^{t+1}\|^2 - \left( \frac{\rho_i}{2} - \frac{3L_{i,j}}{2} \right) \|x_{i,j}^{t+1} - z_j^{t+1}\|^2 \\
& + \frac{4L_{i,j} + \rho_i}{2} \|z_j^{t+1} - z_j^{t+1}\|^2.
\end{align*}
\]

Corollary 2. If block \( j \) is randomly drawn from uniform distribution, we have
\[
\mathbb{E}_j[L_i(x_{i,t}^{t+1}, y_{i,j}, z_j^{t+1})] \leq \mathbb{E}_j[L_i(x_{i,t}^{t+1}, y_{i,j}, z_j^{t+1})] - \frac{1}{|N(i)|} \sum_{j \in N(i)} \left( \frac{\rho_i}{2} - \frac{4L_{i,j}}{2} \right) \|x_{i,j}^{t+1} - z_j^{t+1}\|^2 \\
- \frac{1}{|N(i)|} \sum_{j \in N(i)} \left( \frac{\rho_i}{2} - \frac{3L_{i,j}}{2} \right) \|x_{i,j}^{t+1} - z_j^{t+1}\|^2 + \frac{1}{|N(i)|} \sum_{j \in N(i)} \frac{4L_{i,j} + \rho_i}{2} \|z_j^{t+1} - z_j^{t+1}\|^2.
\]

Lemma 6. For node \( i \), we have the following inequality to bound the gap after updating \( y_j \):
\[
l_i(x_{i,t}^{t+1}, y_{i,j}, z_j^{t+1}) - l_i(x_{i,t}^{t+1}, y_{i,j}, z_j^{t+1}) \leq \left( \frac{1}{\rho_i} + \frac{1}{2} \right) \|y_{i,j}^{t+1} - y_{i,j}^t\|^2 + \frac{1}{2} \|z_j^{t+1} - z_j^{t+1}\|^2.
\]

Proof. From (42) we have
\[
l_i(x_{i,t}^{t+1}, y_{i,j}, z_j^{t+1}) - l_i(x_{i,t}^{t+1}, y_{i,j}, z_j^{t+1}) = \langle y_{i,j}^{t+1} - y_{i,j}^t, z_j^{t+1} - z_j^{t+1} \rangle \\
= \langle y_{i,j}^{t+1} - y_{i,j}^t, z_j^{t+1} - z_j^{t+1} \rangle + \langle y_{i,j}^{t+1} - y_{i,j}^t, z_j^{t+1} - z_j^{t+1} \rangle \\
\leq \frac{1}{\rho_i} \|y_{i,j}^{t+1} - y_{i,j}^t\|^2 + \frac{1}{2} \|y_{i,j}^{t+1} - y_{i,j}^t\|^2 + \frac{1}{2} \|z_j^{t+1} - z_j^{t+1}\|^2 \\
= \left( \frac{1}{\rho_i} + \frac{1}{2} \right) \|y_{i,j}^{t+1} - y_{i,j}^t\|^2 + \frac{1}{2} \|z_j^{t+1} - z_j^{t+1}\|^2.
\]

Corollary 3. If block \( j \) is randomly drawn from uniform distribution, we have
\[
\mathbb{E}_j[L_i(x_{i,t}^{t+1}, y_{i,j}, z_j^{t+1})] - \mathbb{E}_j[L_i(x_{i,t}^{t+1}, y_{i,j}, z_j^{t+1})] \\
\leq \frac{1}{|N(i)|} \left( \frac{1}{\rho_i} + \frac{1}{2} \right) \sum_{j \in N(i)} \|y_{i,j}^{t+1} - y_{i,j}^t\|^2 + \frac{1}{|N(i)|} \sum_{j \in N(i)} \|z_j^{t+1} - z_j^{t+1}\|^2
\]

Lemma 7. After updating \( z_j \) to \( z_j^{t+1} \), we have
\[
\mathbb{E}_j[l(X_i, y_{i,j}, z_j^{t+1})] - \mathbb{E}_j[l(X_i, y_{i,j}, z_j^{t+1})] \leq \sum_{i \in N(j)} \gamma + \frac{\rho_i}{|N(i)|} \cdot \|z_j^{t+1} - z_j^{t+1}\|^2.
\]

(43)
Proof. We begin our proof by analyzing the block \(j\). Let

\[
l(X, Y, z_j) := h_j(z_j) + \sum_{i \in \mathcal{N}(j)} \langle y_{i,j}, x_{i,j} - z_j \rangle + \sum_{i \in \mathcal{N}(j)} \frac{\rho_i}{2} \|x_{i,j} - z_j\|^2.
\]

Firstly, it is clear that \(\langle y_{i,j}, x_{i,j} - z_j \rangle + \rho_i \|x_{i,j} - z_j\|^2\) is a quadratic function and thus strongly convex. Then, we have:

\[
\sum_{i \in \mathcal{N}(j)} \langle y_{i,j}, x_{i,j} - z_j^{t+1} \rangle + \frac{\rho_i}{2} \|x_{i,j} - z_j^{t+1}\|^2 - \sum_{i \in \mathcal{N}(j)} \langle y_{i,j}, x_{i,j}^t - z_j^t \rangle - \frac{\rho_i}{2} \|x_{i,j} - z_j^t\|^2 \\
\leq \langle -y_{i,j} - \rho_i(x_{i,j}^t - z_j^{t+1}), z_j^{t+1} - z_j^t \rangle - \sum_{i \in \mathcal{N}(j)} \frac{\rho_i}{2} \|z_j^{t+1} - z_j^t\|^2.
\]

By the optimality in (15), we have

\[
\langle p_j^{t+1} - \sum_{i \in \mathcal{N}(j)} y_{i,j}^t + \rho_i(z_j^{t+1} - x_{i,j}^t) + \gamma(z_j^{t+1} - z_j^t), z_j^{t+1} - z_j^t \rangle \leq 0,
\]

where \(p_j^{t+1} \in \partial h_j(z_j^{t+1})\) is a subgradient. By convexity of \(h_j\), we have

\[
h_j(z_j^{t+1}) - h_j(z_j^t) \leq \langle p_j^{t+1}, z_j^{t+1} - z_j^t \rangle \\
\leq \langle \sum_{i \in \mathcal{N}(j)} y_{i,j}^t - \rho_i(z_j^{t+1} - x_{i,j}^t) - \gamma(z_j^{t+1} - z_j^t), z_j^{t+1} - z_j^t \rangle
\]

Therefore, by taking expectation on \(j\), we have

\[
\mathbb{E}_j \left[ l(X^t, Y^t, z_j^{t+1}) + \frac{\gamma}{2} \|z_j^{t+1} - z_j^t\|^2 \right] - \mathbb{E}_j[l(X^t, Y^t, z_j^t)] \\
\leq \langle - \sum_{i \in \mathcal{N}(j)} \frac{1}{|\mathcal{N}(i)|} (y_{i,j}^t - \rho_i(x_{i,j}^t - z_j^{t+1})), z_j^{t+1} - z_j^t \rangle - \sum_{i \in \mathcal{N}(j)} \frac{\rho_i}{2|\mathcal{N}(i)|} \|z_j^{t+1} - z_j^t\|^2 \\
+ \langle \sum_{i \in \mathcal{N}(j)} \frac{1}{|\mathcal{N}(i)|} (y_{i,j}^t - \rho_i(z_j^{t+1} - x_{i,j}^t) - \gamma(z_j^{t+1} - z_j^t)), z_j^{t+1} - z_j^t \rangle \\
= - \sum_{i \in \mathcal{N}(j)} \frac{\gamma + 2\rho_i}{2|\mathcal{N}(i)|} \|z_j^{t+1} - z_j^t\|^2.
\]

which proves the lemma. \(\square\)

We now proceed to prove Lemma 3. From Corollary 2.5 and Lemma 7, we have three upper bounds when updating \(x_{i,j}^t\), \(y_{i,j}^t\) and \(z_j^t\), respectively, and we observe that the sign of \(\|x_{i,j} - z_j\|\) can be negative by assuming \(\rho_i \geq 3L_{i,j}\), and similarly for \(\|x_{i,j}^{t+1} - z_j^t\|\) by assuming \(\rho_i \geq 4L_{i,j} \geq 0\). Therefore, let

\[
\rho_i - 4 \max_{j \in \mathcal{N}(i)} L_{i,j} \geq 0,
\]

and then we can guarantee that the efficient for all \((i, j) \in \mathcal{E}\), the coefficients for \(\|x_{i,j} - z_j\|\) and \(\|x_{i,j}^{t+1} - z_j^t\|\) are always negative.

Then, the major challenge is to make the coefficient of \(\|z_j^{t+1} - z_j^t\|\) be negative, and we attempt to make it as follows:

\[
\frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \left( \frac{1}{\rho_i} + \frac{1}{2} \right) \|y_{i,j}^{t+1} - y_{i,j}^t\|^2 \leq \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \left( \frac{1}{\rho_i} + \frac{1}{2} \right) L_{i,j}^2 (T_{i,j} + 1) \sum_{t' = 0}^{T_{i,j}} \|z_j^{t-t'} - z_j^{t-1}\|^2,
\]

\[
\frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \frac{4L_{i,j} + \rho_i + 1}{2} \|z_j^{t+1} - z_j^t\|^2 \leq \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \frac{4L_{i,j} + \rho_i + 1}{2} T_{i,j} \sum_{t' = 0}^{T_{i,j} - 1} \|z_j^{t+1-t'} - z_j^{t-t'}\|^2.
\]
We now combine equations (40), (41) and (42), and sum over all workers $i$:

\[
\begin{align*}
\mathbb{E}_j[L(X^{t+1}, Y^{t+1}, z^{t+1})] - \mathbb{E}_j[L(X^t, Y^t, z^t)] &
\leq - \sum_{(i,j) \in \mathcal{E}} \frac{1}{|N(i)|} \left( \frac{\rho_i}{2} - \frac{4L_{i,j}}{2} \right) \|x_{i,j}^{t+1} - z_{i,j}^{t+1}\|^2 - \sum_{(i,j) \in \mathcal{E}} \frac{1}{|N(i)|} \left( \frac{\rho_i}{2} - \frac{3L_{i,j}}{2} \right) \|x_{i,j}^t - z_{i,j}^{t+1}\|^2 \\
&\quad + \sum_{(i,j) \in \mathcal{E}} \frac{4L_{i,j} + \rho_i + 1}{2|N(i)|} \cdot \|z_{i,j}^{t+1} - z_{i,j}^t\|^2 + \frac{1}{|N(i)|} \left( \frac{\rho_i}{2} + \frac{1}{2} \right) \|y_{i,j}^{t+1} - y_{i,j}^t\|^2 - \sum_{(i,j) \in \mathcal{E}} \frac{\gamma + \rho_i}{|N(i)|} \cdot \|z_{i,j}^{t+1} - z_{i,j}^t\|^2 \\
&\leq - \sum_{(i,j) \in \mathcal{E}} \frac{\rho_i - 4L_{i,j}}{2|N(i)|} \left( \|x_{i,j}^{t+1} - z_{i,j}^t\|^2 + \|x_{i,j}^t - z_{i,j}^{t+1}\|^2 \right) - \sum_{(i,j) \in \mathcal{E}} \frac{\gamma + \rho_i}{|N(i)|} \cdot \|z_{i,j}^{t+1} - z_{i,j}^t\|^2 \\
&\quad + \sum_{(i,j) \in \mathcal{E}} \frac{1}{|N(i)|} \left( \frac{1}{\rho_i} + \frac{1}{2} \right) L_{i,j}^2 (T_{i,j} + 1) \sum_{t' = 0}^{T_{i,j} - 1} \|z_{i,j}^{t+1-t'} - z_{i,j}^{t-t'}\|^2 \\
&\quad + \sum_{(i,j) \in \mathcal{E}} \frac{4L_{i,j} + \rho_i + 1}{2|N(i)|} \cdot T_{i,j} \sum_{t' = 0}^{T_{i,j} - 1} \|z_{i,j}^{t+1-t'} - z_{i,j}^{t-t'}\|^2.
\end{align*}
\]

By taking the telescope sum for $t = 0, \ldots, T - 1$, we have

\[
\begin{align*}
\mathbb{E}_j[L(X^T, Y^T, z^T)] - \mathbb{E}_j[L(X^0, Y^0, z^0)] &
\leq - \sum_{t=0}^{T-1} \sum_{(i,j) \in \mathcal{E}} \frac{\rho_i - 4L_{i,j}}{2|N(i)|} \left( \|x_{i,j}^{t+1} - z_{i,j}^t\|^2 + \|x_{i,j}^t - z_{i,j}^{t+1}\|^2 \right) - \sum_{t=0}^{T-1} \sum_{(i,j) \in \mathcal{E}} \frac{\gamma + \rho_i}{|N(i)|} \cdot \|z_{i,j}^{t+1} - z_{i,j}^t\|^2 \\
&\quad + \sum_{t=0}^{T-1} \sum_{(i,j) \in \mathcal{E}} \left( \frac{L_{i,j}^2 (T_{i,j} + 1)^2}{|N(i)|} \left( \frac{1}{\rho_i} + \frac{1}{2} \right) + \frac{(4L_{i,j} + \rho_i + 1)T_{i,j}^2}{2|N(i)|} \right) \|z_{i,j}^{t+1} - z_{i,j}^t\|^2 \\
&\leq - \sum_{t=0}^{T-1} \sum_{(i,j) \in \mathcal{E}} \beta_t (\|x_{i,j}^{t+1} - z_{i,j}^t\|^2 + \|x_{i,j}^t - z_{i,j}^{t+1}\|^2) - \sum_{t=0}^{T-1} \sum_{(i,j) \in \mathcal{E}} \alpha_t \|z_{i,j}^{t+1} - z_{i,j}^t\|^2,
\end{align*}
\]

where

\[
\alpha_j := (\gamma + \rho_i) - \sum_{i \in N(j)} \left( \frac{1}{\rho_i} + \frac{1}{2} \right) L_{i,j}^2 (T_{i,j} + 1)^2 - \sum_{i \in N(j)} \frac{(4L_{i,j} + \rho_i + 1)T_{i,j}^2}{2|N(i)|},
\]

\[
\beta_i := \rho_i - \max_{j \in N(i)} 4L_{i,j}.
\]

By making $\alpha_j > 0$ and $\beta_i > 0$ for all $(i,j) \in \mathcal{E}$, we prove the lemma.

**C Proof of Lemma 4**

**Proof.** From Lipschitz continuity assumption, we have:

\[
\begin{align*}
f_i(x^{t+1}) &\leq f_i(x_i^{t+1}) + \sum_{j \in N(i)} \langle \nabla_j f_i(x_j^{t+1}), z_j^{t+1} - x_{i,j}^{t+1} \rangle + \sum_{j \in N(i)} \frac{L_{i,j}}{2} \|x_{i,j}^{t+1} - z_{i,j}^{t+1}\|^2 \\
&= f_i(x_i^{t+1}) + \sum_{j \in N(i)} \langle \nabla_j f_i(x_j^{t+1}) - \nabla_j f_i(z_j^{t+1}), z_j^{t+1} - x_{i,j}^{t+1} \rangle \\
&\quad + \sum_{j \in N(i)} \langle \nabla_j f_i(z_j^{t+1}), z_j^{t+1} - x_{i,j}^{t+1} \rangle + \sum_{j \in N(i)} \frac{L_{i,j}}{2} \|x_{i,j}^{t+1} - z_{i,j}^{t+1}\|^2 \\
&\leq f_i(x_i^{t+1}) + \sum_{j \in N(i)} \langle \nabla_j f_i(z_j^{t+1}), z_j^{t+1} - x_{i,j}^{t+1} \rangle + \sum_{j \in N(i)} \frac{3L_{i,j}}{2} \|x_{i,j}^{t+1} - z_{i,j}^{t+1}\|^2.
\end{align*}
\]
Now we have

\[
L(X^{t+1}, z^{t+1}, Y^{t+1}) = h(z^{t+1}) + \sum_{i=1}^{N} f_i(x_i^{t+1}) + \sum_{j \in N(i)} \langle y_{i,j}^{t+1}, x_i^{t+1} - z_j^{t+1} \rangle + \frac{\rho_i}{2} \|x_i^{t+1} - z_j^{t+1}\|^2
\]

\[
\geq h(z^{t+1}) + \sum_{i=1}^{N} f_i(z^{t+1}) + \sum_{(i,j) \in E} \langle \nabla f_i(z^{t+1}) - \nabla f_i(z^{t+1}), z_{i,j}^{t+1} - x_{i,j}^{t+1} \rangle
\]

\[
+ \sum_{(i,j) \in E} \frac{\rho_i - 3L_i,j}{2} \|x_{i,j}^{t+1} - z_j^{t+1}\|^2
\]

\[
\geq h(z^{t+1}) + \sum_{i=1}^{N} f_i(z^{t+1}) + \sum_{(i,j) \in E} \left( \frac{\rho_i - 3L_i,j}{2} \|x_{i,j}^{t+1} - z_j^{t+1}\|^2 - \frac{L_i,j}{2} \|z_{i,j}^{t+1} - z_i^{t+1}\|^2 \right)
\]

\[
\geq h(z^{t+1}) + \sum_{i=1}^{N} f_i(z^{t+1}) - \sum_{(i,j) \in E} \frac{L_i,j}{2} \|z_{i,j}^{t+1} - z_i^{t+1}\|^2
\]

\[
\geq f - \text{diam}^2(\mathcal{A}) \sum_{(i,j) \in E} \frac{L_i,j}{2} > -\infty.
\]

C.1 Proof of Theorem 1

Proof. From Lemma 3, we must have, as \( t \to \infty \),

\[
x_{i,j}^{t+1} - z_{i,j}^{t+1} \to 0, \quad z_j^{t+1} - z_j^t \to 0, \quad x_j^t - z_j^{t+1} \to 0, \quad \forall (i,j) \in E. \tag{43}
\]

Given Lemma 1 we have \( y_{i,j}^{t+1} - y_{i,j}^t \to 0 \). Since

\[
\|x_{i,j}^{t+1} - x_{i,j}^t\| \leq \|x_{i,j}^{t+1} - z_{i,j}^{t+1}\| + \|z_{i,j}^{t+1} - x_{i,j}^t\| + \|z_{i,j}^{t+1} - z_{i,j}^{t+1}\| \to 0,
\]

which proves (20c), the first part.

For the second part, we have the following inequality from the optimality condition of (13):

\[
0 \in \partial h_j(z_j^{t+1}) - \sum_{i \in N(j)} \left( y_{i,j}^{t+1} + \rho_i (x_{i,j}^{t+1} - z_{i,j}^{t+1}) + \gamma (z_j^t - z_{i,j}^{t+1}) \right). \tag{44}
\]

From (19a) and (20c), we have

\[
0 \in \partial h_j(z_j^t) - \sum_{i \in N(j)} y_{i,j}^t,
\]

which proves (20b). Finally, from the optimality condition in (13), we have (11) which implies (20a), the second part of the theorem.

We now turn to prove the last part. Let \( L'(X, Y, z) := L(X, Y, z) - h(z) \), which excludes \( h(z) \) from \( L(X, Y, z) \). Then, we have

\[
z_j - \nabla z_j l'(X, Y, z_j) = z_j - \sum_{i \in N(j)} y_{i,j} - \sum_{i \in N(j)} \rho_i (x_{i,j} - z_j)
\]

\[
= z_j - \sum_{i \in N(j)} \rho_i (z_j - x_{i,j} - \frac{y_{i,j}}{\rho_i}).
\]
Therefore, we have
\[ \|z^t_j - \text{prox}_h(z^t_j - \nabla z_j l'(X^t, Y^t, z^t_j))\| \leq \|z^t_j - z_j^{t+1} + z_j^{t+1} - \text{prox}_h(z^t_j - \nabla z_j l'(X^t, Y^t, z^t_j))\| \]
\[ \leq \|z^t_j - z_j^{t+1}\| + \|z_j^{t+1} - \text{prox}_h(z_j - \sum_{i \in N(j)} \rho_i (z^t_j - x_i^t - \frac{y_{i,j}^t}{\rho_i}))\| \]
\[ \leq \|z^t_j - z_j^{t+1}\| + \|\text{prox}_h(z_j^{t+1} - \sum_{i \in N(j)} \rho_i (z_j^{t+1} - x_i^t - \frac{y_{i,j}^t}{\rho_i})) + \gamma(z_j^{t+1} - z_j^t)\|
- \text{prox}_h(z_j^t - \sum_{i \in N(j)} \rho_i (z_j^t - x_i^t - \frac{y_{i,j}^t}{\rho_i}))\| \]
\[ \leq \left(2 + \gamma + \sum_{i \in N(j)} \rho_i \right) \|z_j^t - z_j^{t+1}\|\]  \hspace{1cm} (46)
where (46) is from the optimality in [13] as
\[ z_j^{t+1} = \text{prox}_h(z_j^{t+1} - \sum_{i \in N(j)} \rho_i (z_j^{t+1} - x_i^t - \frac{y_{i,j}^t}{\rho_i}) + \gamma(z_j^{t+1} - z_j^t)) \]
and (47) is from the firm nonexpansiveness property of proximal operator. Then, by the update rule of \(x_{i,j}^{t+1}\), if \(x_{i,j}\) is selected to update at epoch \(t\), we have
\[ \|\nabla_{x_{i,j}} L(X^t, Y^t, z^t)\|^2 = \|\nabla_j f_j(x_j^t) + \rho_i (x_{i,j}^t - z_j^t + \frac{y_{i,j}^t}{\rho_i})\|^2 \]
\[ = \|\nabla_j f_j(x_j^t) - \nabla_j f_j(\tilde{z}^t_j) + (y_{i,j}^t - y_{i,j}^{t-1}) + \rho_i (\tilde{z}_j^t - z_j^t)\|^2 \]
\[ \leq 3\|\nabla_j f_j(x_j^t) - \nabla_j f_j(\tilde{z}^t_j)\|^2 + 3\|y_{i,j}^t - y_{i,j}^{t-1}\|^2 + 3\|\rho_i (\tilde{z}_j^t - z_j^t)\|^2 \]
\[ \leq 3L(z_j^t - z_j^{t+1})^2 + 3 \rho_i^2 \|z_{i,j}^t - z_{i,j}^{t+1}\|^2 \]
\[ \leq 3L^2(z_j^t - z_j^{t+1})^2 + 3 \rho_i^2 \|z_{i,j}^t - z_{i,j}^{t+1}\|^2 \]

which implies that there must exist two positive constant \(\sigma_1 > 0\) and \(\sigma_2 > 0\) such that
\[ \sum_{(i,j) \in E} \|\nabla_{x_{i,j}} L(X^t, Y^t, z^t)\|^2 \leq \sum_{(i,j) \in E} \sigma_1 \|x_{i,j}^t - z_{i,j}^t\|^2 + \sum_{(i,j) \in E} \sigma_2 \sum_{t'=0}^{T_{i,j} - 1} \|z_{i,j}^{t-t'} - z_{i,j}^{t-t'-1}\|^2\]  \hspace{1cm} (48)

The last step is to estimate \(\|x_{i,j}^t - z_{i,j}^t\|^2\), which can be done as follows:
\[ \|x_{i,j}^t - z_{i,j}^t\|^2 \leq \|x_{i,j}^t - \tilde{z}_{i,j}^t\|^2 + \|\tilde{z}_{i,j}^t - z_{i,j}^t\|^2 \]
\[ \sum_{(i,j) \in E} \|x_{i,j}^t - z_{i,j}^t\|^2 \leq \sum_{(i,j) \in E} \|x_{i,j}^t - \tilde{z}_{i,j}^t\|^2 + \|\tilde{z}_{i,j}^t - z_{i,j}^t\|^2 \]  \hspace{1cm} (49)

Combining (47), (48) and (50), and summing up \(t = 0, \ldots, T - 1\), we have
\[ \sum_{i,j \in E} \text{prox}_h(z_j^{t+1} - \tilde{z}_j^{t+1} - z_j^t) \]
\[ \leq \sum_{i,j \in E} \sigma_3 \|x_{i,j}^t - \tilde{z}_{i,j}^t\|^2 + \sigma_4 T_{i,j} \|z_j^{t+1} - z_j^t\|^2 \]
\[ \sum_{i,j \in E} \text{prox}_h(z_j^{t+1} - \tilde{z}_j^{t+1} - z_j^t) \]
\[ \leq \sum_{i,j \in E} \delta_1 \|x_{i,j}^{t+1} - \tilde{z}_j^{t+1} - z_j^t\|^2 + \delta_2 \|z_j^{t+1} - z_j^t\|^2 \]  \hspace{1cm} (51)

From Lemma 3 we have
\[ L(X^T, Y^T, z^T) - L(X^0, Y^0, z^0) \]
\[ \leq - \sum_{i,j \in E} \beta_i (\|x_{i,j}^{t+1} - z_j^{t+1}\|^2 + \|x_{i,j}^t - z_j^{t+1}\|^2) - \sum_{i,j \in E} \alpha_3 \|z_j^{t+1} - z_j^t\|^2 \]
\[ \leq - \sum_{i,j \in E} \delta_1 \|x_{i,j}^{t+1} - z_j^{t+1}\|^2 + \delta_2 \|z_j^{t+1} - z_j^t\|^2 \]  \hspace{1cm} (52)
where $\delta_1 := \min_i \beta_i$ and $\delta_2 := \min_j \alpha_j$. Now we can find some $C > 0$, such that the following equation hold:

$$\sum_{t=0}^{T-1} P(X^t, Y^t, z^t) \leq C(L(X^0, Y^0, z^0) - L(X^T, Y^T, z^T))$$

$$\leq C(L(X^0, Y^0, z^0) - f),$$

where the last inequality we have used the fact that $L(X^t, Y^t, z^t)$ is lowered bounded by $f$ for all $t$ from Lemma 4. Let $T = T(\epsilon)$ and we have

$$T(\epsilon) \leq \frac{C(L(X^0, Y^0, z^0) - f)}{\epsilon},$$

which proves the last part of Theorem 1. \hfill \Box