Fast-Converging Decentralized ADMM for Consensus Optimization

1st Jeannie He
Div. of Info. Science and Engineering
KTH Royal Institute of Technology
Stockholm, Sweden
jeannie@kth.se

2nd Ming Xiao
Div. of Info. Science and Engineering
KTH Royal Institute of Technology
Stockholm, Sweden
mingx@kth.se

3rd Mikael Skoglund
Div. of Info. Science and Engineering
KTH Royal Institute of Technology
Stockholm, Sweden
skoglund@kth.se

***Abstract*—For its well-established convergence properties and applicability to various optimization problems, the alternating direction method of multipliers (ADMM) has been at the center of several research fields. When applied to distributed problems such as consensus optimization, ADMM is typically implemented in a centralized manner. Such implementations are, however, discouraged for e.g. their dependency on the location and capacity of the central node. While there are decentralized alternatives, these implementations are either computationally and communication-wise expensive or slow. This is because existing decentralized alternatives require all worker nodes to either replicate the work of synchronizing the outputs from all nodes or execute their tasks in sequence. To address this problem, we propose a fast-converging decentralized ADMM (FCD-ADMM) algorithm. Through theoretical analysis, we prove the convergence properties of FCD-ADMM and show that FCD-ADMM can converge faster than its centralized alternative without sacrificing accuracy. As shown in our numerical experiments, FCD-ADMM can converge to the same or better solution faster than several state-of-the-art alternatives.**

***Index Terms*—Alternating Direction Method of Multipliers (ADMM), decentralized optimization, distributed optimization, consensus optimization, convergence rate.**

I. Introduction

The alternating direction method of multipliers (ADMM) is a well-known algorithm with established convergence properties. Amongst the areas applicable to ADMM, one area is distributed systems, where ADMM is typically used as the standard tool for solving the following optimization problem:

$$\begin{aligned} \min F(x) &= \min \sum_{i=1}^{N} f_i(x_i), \\ \text{s.t. } x_i - z &= 0, \quad \forall i = 1, 2, \ldots, N, \end{aligned} \tag{1}$$

where $f_i(x_i) : \mathbb{R}^n \to \mathbb{R}$ is the local loss function at node $i = 1, \ldots, N$, and $x_i \in \mathbb{R}^n$ is the local primal variable minimizing $f_i(x_i)$ at node i; $x = [x_1, \ldots, x_N]$ is the concatenation of the primal variables at all nodes; and $z \in \mathbb{R}^n$ is the global variable shared by the nodes [1]. This is a typical consensus optimization problem definition applicable to e.g. classification problems using data from smart devices and cost minimization challenges in smart grids.

By introducing a dual variable $y_i \in \mathbb{R}^n$ and a step size $\rho > 0$, one can iteratively solve (1) through the following

$$\begin{cases} x_i^{k+1} = \operatorname{argmin}_x f_i(x) + \left\langle y_i^k, x - z^k \right\rangle + \frac{\rho}{2} \left\| x - z^k \right\|^2, \\ y_i^{k+1} = y_i^k + \rho \left(x_i^{k+1} - z^k \right), \\ z^{k+1} = \frac{1}{N} \sum_i \left(x_i^{k+1} + \frac{y_i^{k+1}}{\rho} \right). \end{cases} \tag{2}$$

As in [2]–[10], the algorithm can be implemented by letting nodes i compute and send the local variables (x_i^{k+1}, y_i^{k+1}) to a central node, which will then compute and send the global variable z^{k+1} to the nodes [1]. The corresponding algorithm is henceforth referred to as the classical centralized ADMM (CC-ADMM).

Given the disadvantages of centralized ADMM implementations (e.g. high bandwidth demand [11], high dependency on the capacity and location of the central node [12], and high risk of single point failures [13]), the authors in [14]–[18] proposed algorithms to achieve decentralized ADMM by letting the worker nodes take over the responsibility for synchronizing outputs. Since the responsibility is replicated rather than shared, these algorithms suffer from replicated work as well as a high dependency on the capacity and locations of the worker nodes.

As solutions, Random Walk ADMM (RW-ADMM) [19], Incremental ADMM (I-ADMM) [20], and Parallel Random Walk ADMM (PW-ADMM) [21] achieve decentralized ADMM by letting the worker nodes operate in sequence, except that PW-ADMM has multiple updating threads working in parallel, each with its own version of z^k. Due to their reliance on sequential operations, these algorithms have the disadvantage of slow convergence.

A. Contribution

From the above, we can see that existing decentralized ADMM algorithms have the disadvantage of requiring all nodes to either aggregate the outputs from all nodes on their own or execute all of their tasks in sequence. Given the importance of computational and communication efficiency as well as fast decision-making, we have proposed a fast-converging decentralized ADMM algorithm (FCD-ADMM) to address this problem. In this algorithm, we introduce a shared variable z_i^k. Using this shared variable z_i^k, we make it possible for the nodes to split rather than replicate the work of synchronizing the outputs and we enable fast convergence by letting the worker nodes compute (x_i^k, y_i^k) in parallel as well as by removing the need to wait with the computation of z^k until a master node has received and confirmed the receipt of (x_i^k, y_i^k) from all nodes in the supporting network. This is what differentiates our FCD-ADMM from the aforementioned algorithms. Compared to existing algorithms, our FCD-ADMM has the following advantages:

1) Fast convergence to optimal solution - As will be shown in our numerical experiment results, our FCD-ADMM has converged faster than CC-ADMM by yielding the same outputs in a shorter time. As mentioned earlier, this is because, unlike CC-ADMM, our FCD-ADMM does not require the computation of z^k to start after a master node has received and verified the receipt of (x_i^k, y_i^k) from all worker nodes. Aside from converging faster than CC-ADMM, our FCD-ADMM also yielded the same or better optimal object value than I-ADMM, RW-ADMM, and PW-ADMM. Again, this is expected because, unlike these algorithms, FCD-ADMM lets the nodes collectively compute z^k based on the local variables that they have computed in parallel.
2) Scalability - As will be shown in our theoretical analysis, the computational and communication cost with respect to the number of nodes N in the supporting network is, for each node in each iteration of FCD-ADMM, given by $\mathcal{O}(N)$. Thus, our algorithm is scalable in that it allows the supporting network to grow without increasing the computing power or bandwidth demand.

II. Proposed Algorithm

Consider a network with nodes $i = 1, \dots, N$, where the expected time for computing x_i^k and y_i^k is the same for all nodes i and node i is connected to its neighbor $(i \bmod N)+1$. Inspired by [20], we introduce a variable $z_{i_j^k}^k$ defined as

$$z_{i_j^k}^k = z_{i_{-1}^k}^k + \frac{1}{N}\left[\left(x_{i_j^k}^k + \frac{y_{i_j^k}^k}{\rho}\right) - \left(x_{i_j^k}^{k-1} + \frac{y_{i_j^k}^{k-1}}{\rho}\right)\right], \quad (3)$$

for $j = 1, 2, \dots, N$, where $i_{-1}^k = i_{j-1}^k$,

$$z_{i_0^k}^k = \frac{1}{N}\sum_{j=1}^{N}\left(x_{i_j^k}^{k-1} + \frac{y_{i_j^k}^{k-1}}{\rho}\right), \quad (4)$$

and i_j^k is the index of the j^{th} node to contribute to the computation of $z_{i_N^k}^k$ by computing and sending $z_{i_j^k}^k$ to node i_{j+1}^k. While this can be achieved by both $i_j^k = j$ and

$$i_j^k = ((j-k) \bmod N) + 1 \quad (5)$$

as they both give $i_{j+1}^k = (i_j^k \bmod N) + 1$, we note that only (5) also gives

$$i_1^{k+1} = i_N^k. \quad (6)$$

Algorithm 1 FCD-ADMM at node i

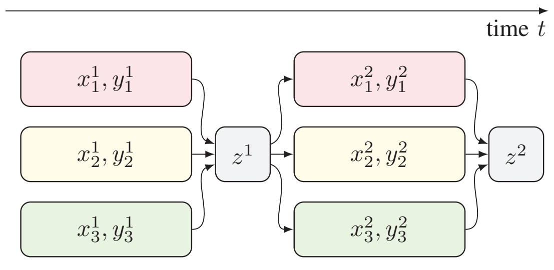
```
1: Input: $\rho$, $x_i^0$, $y_i^0$, $z^0$
2: for k = 1,2,... do
3:     set $x_i^k$ according to (2)
4:     set $y_i^k$ according to (2)
5:     if $i \neq i_1^k$ then
6:         wait for $z_{i_{-1}}^k$
7:     end if
8:     set $z_i^k$ according to (3)
9:     send $z_i^k$ to node $(i \bmod N)+1$
10:    if $i \neq i_N^k$ then
11:        wait for $z^k$
12:        if $i \neq i_{N-1}^k$ then
13:            forward $z^k$ to node $(i \bmod N)+1$
14:        end if
15:    end if
16: end for
```

This means that if $z^k = z_{i_N^k}^k$, then enforcing (5) instead of $i_j^k = j$ can reduce the time to complete iteration $k+1$ by making it possible for node with the j^{th} access to $z_{i_N^k}^k$ to be the j^{th} node to contribute to the computation of z^{k+1}. To show that $z^k = z_{i_N^k}^k$, we note that (3) - (5) give

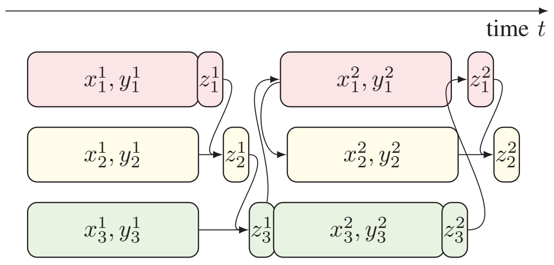
$$\begin{aligned} z_{i_N^k}^k &= z_{i_0^k}^k + \sum_{j=1}^{N}\frac{1}{N}\left[\left(x_{i_j^k}^k + \frac{y_{i_j^k}^k}{\rho}\right) - \left(x_{i_j^k}^{k-1} + \frac{y_{i_j^k}^{k-1}}{\rho}\right)\right] \\ &= \frac{1}{N}\sum_{j=1}^{N}\left(x_{i_j^k}^{k-1} + \frac{y_{i_j^k}^{k-1}}{\rho}\right) = \frac{1}{N}\sum_{i=1}^{N}\left(x_i^k + \frac{y_i^k}{\rho}\right) \\ &= z^k. \end{aligned} \quad (7)$$

Hence, we can replace the computation of z^k in (2) with the computation of $z_{i_N{}^k}^k$ in accordance with (3) - (5). The resulting algorithm is called Fast-Converging Decentralized ADMM (FCD-ADMM). The pseudo-code can be found in Algorithm 1.

For illustration, Fig. 1 shows the flow of CC-ADMM and FCD-ADMM in a network of 3 worker nodes. As shown in the figure, both FCD-ADMM and CC-ADMM start with all worker nodes i computing x_i^1 and y_i^1 in parallel. However, while CC-ADMM continues with all worker nodes i sending x_i^1 and y_i^1 to the central node to get z^1 in return; FCD-ADMM continues with, as per (5), node $i_1^1 = ((1-1) \bmod 3)+1 = 1$ computing and sending z_1^1 to node $i_2^1 = ((2-1) \bmod 3) + 1 = 2$, and so on until $z^1 = z_{i_N^1}^1 = z_{i_3^1}^1$ is computed by node $i_3^1 = ((3-1) \bmod 3) + 1 = 3$. Moreover, while z^1 is broadcasted in CC-ADMM by the central node, this variable is forwarded in FCD-ADMM, as per (6), from node $i_1^2 = i_3^1 = 3$ to node $i_2^2 = ((2-2) \bmod 3) + 1 = 1$ and so on until all nodes have access to z^1. Again, while x_i^2 and y_i^2 are computed in both CC-ADMM and FCD-ADMM as soon as node i has access to z^1, this step is followed in CC-ADMM by x_i^2 and



a) CC-ADMM



(b) FCD-ADMM

Fig. 1. Two Gantt charts demonstrating the flow of a) CC-ADMM and b) FCD-ADMM in a network of $N = 3$ worker nodes, where the activities executed by the worker nodes $i = 1, 2, 3$ and the central node are represented by red, yellow, green, and gray boxes, respectively.

y_i^2 being sent to the central node but in FCD-ADMM by node $i_1^2 = 3$ computing and sending z_3^2 to node $i_2^2 = 1$, and so on until $z^2 = z^1_{i^2_N} = z^1_{i^2_3}$ is computed by node $i_3^2 = ((3-2) \bmod 3) + 1 = 2$.

III. Theoretical analysis

In what follows, we will analyze FCD-ADMM from various perspectives under the following definitions and assumptions:

Definition III.1. The communication delay of a message is defined as the time interval between the moment the sender transmits the message and when it is received by the recipient.

Definition III.2. The start of iteration $k+1$ is defined as the end of iteration k. The end of iteration k is the point when $\{x_i^k, y_i^k, z^k | i = 1, 2, \ldots, N\}$ is computed.

Definition III.3. The iteration time, t^k, is defined as the time interval between the start and end of iteration k.

Definition III.4. The aggregated iteration time, T^k, is defined as the time interval between the start and end of iteration 1 and k.

Definition III.5. The convergence time is defined as the time required for an algorithm to reach the state of convergence since the start of iteration 1.

Assumption III.6. The communication delays of two messages are about the same if the messages have the same sender, receiver, and size.

Assumption III.7. Compared to the time expended on the computation of (x_i^k, y_i^k, z_i^k) and on the waiting for other nodes, the time expended on other operations in FCD-ADMM is negligible. Thus, the iteration time in these algorithms can be simplified as solely dependent on the computational time of (x_i^k, y_i^k, z_i^k) and communication delays.

A. Convergence Analysis

Lemma III.8. *Given the same input $\{x_i^0, y_i^0, z^0; i = 1, \ldots, N\}$, the output $\{x_i^k, y_i^k, z^k; i = 1, \ldots, N, k = 1, 2, \ldots\}$ is, as shown in (7), the same in FCD-ADMM as in CC-ADMM.*

Corollary III.9. *Following Lemma III.8, the number of iterations required to reach the state of convergence is the same in FCD-ADMM as in CC-ADMM.*

B. Computational Cost Analysis

In this subsection, we will analyze and compare FCD-ADMM against CC-ADMM from the perspective of computational cost.

Remark III.10. Unlike CC-ADMM, where the computational cost with respect to the number of nodes N is given by $\mathcal{O}(N)$ for the central node, the computational cost with respect to the number of nodes N is given by $\mathcal{O}(1)$ for every node in FCD-ADMM.

C. Communication Cost Analysis

In this subsection, we will analyze and compare FCD-ADMM against CC-ADMM from the perspective of communication cost.

Proposition III.11. *As shown in Figure 1 and Algorithm 1, a node i in FCD-ADMM only needs to send up to 2 messages per iteration to one other node, namely node $(i \text{ mode } N) + 1$. Furthermore, the content of each message sent during iteration k is either z^{k-1} or z_i^k. The communication cost per iteration is thus, for each node, given by $\mathcal{O}(1)$.*

D. Time Analysis on FCD-ADMM

In this subsection, we will analyze FCD-ADMM by deriving its iteration time, t^k, and aggregated iteration time, T^K, for $k \in \{1, 2, \ldots\}$.

Corollary III.12. *Following Assumption III.6 and Proposition III.11, the communication delay in FCD-ADMM is the same for all messages coming from the same node.*

Following Corollary III.12, we have that the communication delay of a message sent from node i in FCD-ADMM can be simplified as a variable solely dependent on the index i. For simplicity, we will therefore denote the communication delay of a message sent from node i in FCD-ADMM as $t_{i,c} \in \mathbb{R}_+$.

Proposition III.13. *The iteration time in FCD-ADMM is, for $k = 1, 2, \ldots$, given by*

$$t^k \leq \max_{i \in \{1,2,\ldots,N\}} \left\{ t^k_{i^k_j,l} \right\} + \sum_{i=1}^{N} t^k_{i,g} + \sum_{j=1}^{N-1} t_{i^k_j,c}, \tag{8}$$

where $t^k_{i,l}$ is the time required to compute the local variables (x^k_i, y^k_i) at node i and iteration k; $t^k_{i,g}$ is the time required to compute the corresponding shared variable z^k_i; $t_{i,c}$ is the communication delay of a message from node i to its subsequent node in FCD-ADMM.

Proof: Let t^k_i be the time interval between the start of iteration k and the moment z^k_i is computed in FCD-ADMM. As shown in Section II, FCD-ADMM lets node $i = i^k_j$ start computing x^k_i, y^k_i, and z^k_i at the start of the iteration if $k = 1$ or $j = 1$; but requires the node to wait with the computation until it has received the necessary inputs otherwise. For $k = 1$ and $n \in \{2, \ldots, N\}$, this gives

$$\begin{aligned} t^k_{i^k_n} &= \max\{t^k_{i^k_{n-1}} + t_{i^k_{n-1},c}, t^k_{i^k_n,l}\} + t^k_{i^k_n,g} \\ &\leq \max\{t^k_{i^k_{n-1}}, t^k_{i^k_n,l}\} + t_{i^k_{n-1},c} + t^k_{i^k_n,g}, \end{aligned} \tag{9}$$

where $t^k_{i^k_1} = t^k_{i^k_1,l} + t^k_{i^k_1,g}$. Through deduction, this gives

$$t^1_{i^1_N} \leq \max_{j \in \{1,2,\ldots,N\}} \left\{ t^1_{i^1_j,l} \right\} + \sum_{j=1}^{N} t^1_{i^1_j,g} + \sum_{j=1}^{N-1} t_{i^1_j,c}. \tag{10}$$

For $k = 2, 3, \ldots$ and $n \in \{2, \ldots, N\}$, we get

$$\begin{aligned} t^k_{i^k_n} &= \max\{t^k_{i^k_{n-1}} + t_{i^k_{n-1},c}, \sum_{j=1}^{n-1} t^k_{i^k_j,c} + t^k_{i^k_n,l}\} + t^k_{i^k_n,g} \\ &= \max\{t^k_{i^k_{n-1}}, \sum_{j=1}^{n-2} t_{i^k_j,c} + t^k_{i^k_n,l}\} + t_{i^k_{n-1},c} + t^k_{i^k_n,g}, \end{aligned} \tag{11}$$

where $t^k_{i^k_1} = t^k_{i^k_1,l} + t^k_{i^k_1,g}$. Through deduction, this gives

$$t^k_{i^k_N} \leq \max_{j \in \{1,2,\ldots,N\}} \left\{ t^k_{i^k_j,l} \right\} + \sum_{j=1}^{N} t^k_{i^k_j,g} + \sum_{j=1}^{N-1} t_{i^k_j,c}. \tag{12}$$

From (10) and (12), we see that (8) holds. Q.E.D.

Proposition III.14. *The aggregated iteration time in FCD-ADMM is bounded by*

$$\begin{aligned} T^K \leq & \sum_{k=1}^{K} \left(\max_i \{t^k_{i,l}\} + \sum_{i=1}^{N} t^k_{i,g} \right) \\ & + \left(K - \left\lfloor \frac{K}{N} \right\rfloor \right) \sum_{i=1}^{N} t_{i,c} - \sum_{i=1}^{K \bmod N} t_{i,c}. \end{aligned} \tag{13}$$

Proof: We have from Proposition III.13 that

$$\begin{aligned} T^K &\leq \sum_{k=1}^{K} t^k = \sum_{k=1}^{K} \left(\max_i \{t^k_{i,l}\} + \sum_{j=1}^{N-1} t_{i,c} + \sum_{i=1}^{N} t^k_{i,g} \right) \\ &= \sum_{k=1}^{K} \left(\max_i \{t^k_{i,l}\} + \sum_{i=1}^{N} t^k_{i,g} \right) + \left(K - c^K_i \right) \sum_{i=1}^{N} t_{i,c}, \end{aligned} \tag{14}$$

where c^K_i is the number of times that the node i has been given the contribution index $j = N$ between the start of iteration 1 and the end of iteration K. Moreover, we have from (5) that

$$i^k_N = (N - k) \bmod N + 1 = -k \bmod N + 1. \tag{15}$$

Due to periodicity, (15) shows that, counting from the start of iteration 1 to the end of iteration, we have

$$c^K_i = \left\lfloor \frac{K}{N} \right\rfloor + \begin{cases} 1, \text{ if } i \in \{1, \ldots, K \bmod N\}, \\ 0, \text{ otherwise.} \end{cases} \tag{16}$$

Substituting (16) into (14) gives (13). Q.E.D.

IV. Numerical Results and Discussions

In this section, we will present, compare, and analyze the performance of FCD-ADMM against some of the ADMM algorithms mentioned in Section I in networks of $N = 30$ worker nodes.

The algorithms are CC-ADMM, RW-ADMM, I-ADMM, and PW-ADMM, where PW-ADMM is implemented with $M = N$ updating threads and the others with $M = 1$ updating thread as per design. The algorithms are chosen for their established convergence properties and applicability to the problem defined in (1).

To reduce the impact of time and statistical variability, the performance is measured by, for each algorithm, repeating $R = 100$ rounds of simulations with 100 iterations per round.

The simulations are run on a Linux server with 12 CPU cores and 64 GB RAM.

Since the novelty of our algorithm from computing the outputs as CC-ADMM in a decentralized manner without causing replicated work or extended convergence time, we did not do any hyperparameter tuning. Instead, we implemented the algorithms with $\rho = 20$ as in [20] and used $x^0_i = y^0_i = z^0 = 0$ for $i = 1, 2, \ldots, N$. For the same reason, we used the default search method from the SciPy Library, namely Broyden–Fletcher–Goldfarb–Shanno algorithm (BFGS), as the method for solving $x^k_i := \operatorname{argmin}_x f_i(x) + \langle y^k_i, x - z^k \rangle + \frac{\rho}{2}|x - z^k|^2$, where the local loss function is defined as

$$f_i(x) = \|A_i x - b_i\|^2, \tag{17}$$

where $A_i \in \mathcal{R}^{n \times m}$ is the feature matrix in the data distributed to node i, $b_i \in \mathcal{R}^m$ is the corresponding label vector. The data comes from [22] and has $n = 4177$ samples with $m = 8$ features per sample. Again, we note that we only considered one local loss function definition and one dataset in our simulations because the novelty of our algorithm is not related to how the local primal variable x^k_i is computed or how the data is handled.

Since the objective is to find the optimal solution z minimizing $F(z) = \sum_i f_i(z)$, we have used various measures to evaluate each algorithm's effectiveness in finding the optimal solution. To begin with, we have, for each iteration k, calculated the average total loss defined as

$$\bar{F}^k = \bar{F}(\bar{T}^k) = \frac{1}{RM} \sum_{r=1}^{R} \sum_{m=1}^{M} \sum_{i=1}^{N} f_i(z^{r,m,k}), \tag{18}$$

where $z^{r,m,k}$ is z^k from thread m and repetition r, and $\bar{T}^k$ is the average time taken to compute z^k since the start of iteration $k = 1$. Following (18), we define the optimal value achieved by an algorithm as

$$\bar{F}^* = \min_k \{\bar{F}^k\}. \quad (19)$$

Based on (18) and (19), we say that an algorithm has converged at iteration k if

$$|\bar{F}^k - \bar{F}^*| < 0.1|\bar{F}^*|. \quad (20)$$

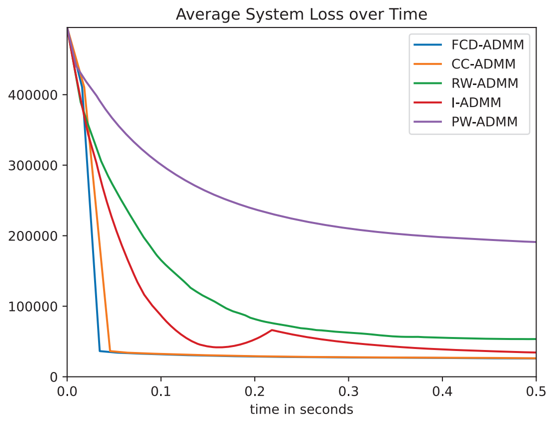


Fig. 2. A figure showing the development of the average system loss value $\bar{F}^k$ for FCD-ADMM, CC-ADMM, RW-ADMM, I-ADMM, and PW-ADMM over time. The y-axis represents the value of $\bar{F}^k$ in each algorithm. The x-axis represents the value of $\bar{T}^k$ as the average taken to complete iteration k in each algorithm, measured from the start of the 1st iteration.

TABLE I
A TABLE SHOWING THE AVERAGE NUMBER OF MILLISECONDS TAKEN TO COMPLETE AN ITERATION AND TO COMPUTE (x_i^k, y_i^k) AS $\bar{t}$ AND $\bar{t}_l$, RESPECTIVELY; THE NUMBER OF ITERATIONS AND SECONDS UNTIL REACHING THE STATE OF CONVERGENCE DEFINED IN (20); AND THE OPTIMAL VALUE $\bar{F}^* = \min_k \bar{F}^k$ ACHIEVED BY EACH ALGORITHM.

ALGORITHM	$\bar{t}$	$\bar{t}_l$	k^*	t^*	$\bar{F}^*$
FCD-ADMM	19 ± 5	5 ± 4	39	0.7	$23k$
CC-ADMM	20 ± 5	6 ± 5	39	0.8	$23k$
RW-ADMM	6 ± 4	6 ± 4	3782	21.5	$24k$
I-ADMM	5 ± 4	5 ± 4	607	4.0	$23k$
PW-ADMM	14 ± 27	9 ± 10	35	0.4	$181k$

As shown in Fig. 2 and Table I, the outcomes of CC-ADMM and FCD-ADMM are almost the same, except that FCD-ADMM has a lower average iteration time, which in turn led to a faster convergence. This confirms Lemma III.8. The lower average iteration time can be explained by the fact that, unlike FCD-ADMM, CC-ADMM requires the computation of z^k to start after the master node has received, stored, and verified that it has registered (x_i^k, y_i^k) from all nodes $i = 1, \ldots, N$.

From Table I, we can further see that FCD-ADMM has converged to the same or better $\bar{F}^*$ than I-ADMM and RW-ADMM using about 6 and 30 times less time, respectively. Moreover, we can see from Fig. 2 that given the same timestamp $t \geq 0.05$ where $t = 0$ at the start of iteration $k = 1$, our FCD-ADMM has, on average, achieved a significantly lower $\bar{F}$ than I-ADMM, RW-ADMM and PW-ADMM. As mentioned earlier, this is expected because, unlike these algorithms, FCD-ADMM lets the nodes collectively compute z^k based on the local variables that they have computed in parallel. Also, we can see from Table I that $\bar{F}^*$ is about 8 times lower and thereby better in FCW-ADMM than in PW-ADMM.

V. CONCLUSIONS

We have proposed an algorithm to achieve decentralized ADMM with fast convergence without causing replicated work. Through theoretical analysis and numerical experiments, we have shown that FCD-ADMM has the same convergence properties as CC-ADMM in terms of the number of iterations required to reach the state of convergence as well as the development of z^k over k. By allowing the computation of z^k to start before all nodes have computed their local variables (x_i^k, y_i^k), our FCD-ADMM has achieved a shorter iteration time than CC-ADMM and thereby a faster convergence in our numerical experiments. By allowing the nodes to collectively compute z^k based on the local variables that they have computed in parallel, our FCD-ADMM has also, given the same timestamp t where $t = 0$ at the start of iteration $k = 1$, yielded a significantly better object value than I-ADMM, RW-ADMM, and PW-ADMM. Altogether, these demonstrate the superiority of our FCD-ADMM.

REFERENCES

[1] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[2] X. Wang, J. Yan, B. Jin, and W. Li, “Distributed and parallel ADMM for structured nonconvex optimization problem,” *IEEE transactions on cybernetics*, vol. 51, no. 9, pp. 4540–4552, 2019.

[3] H. Wang, Y. Gao, Y. Shi, and R. Wang, “Group-based alternating direction method of multipliers for distributed linear classification,” *IEEE transactions on cybernetics*, vol. 47, no. 11, pp. 3568–3582, 2016.

[4] T.-H. Chang, M. Hong, W.-C. Liao, and X. Wang, “Asynchronous distributed ADMM for large-scale optimization—part i: Algorithm and convergence analysis,” *IEEE Transactions on Signal Processing*, vol. 64, no. 12, pp. 3118–3130, 2016.

[5] J. Zhang, S. Nabavi, A. Chakrabortty, and Y. Xin, “ADMM optimization strategies for wide-area oscillation monitoring in power systems under asynchronous communication delays,” *IEEE Transactions on Smart Grid*, vol. 7, no. 4, pp. 2123–2133, 2016.

[6] M. Hong, “A distributed, asynchronous, and incremental algorithm for nonconvex optimization: An ADMM approach,” *IEEE Transactions on Control of Network Systems*, vol. 5, no. 3, pp. 935–945, 2018.

[7] S. Zeng, S. Wang, and Y. Zhang, “Optimization of distributed ADMM algorithm based on minimum network latency,” in *2019 12th International Symposium on Computational Intelligence and Design (ISCID)*, vol. 2. IEEE, 2019, pp. 7–10.

[8] A. Aytekin and M. Johansson, “Exploiting serverless runtimes for large-scale optimization,” *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*, pp. 499–501, 2019.

[9] X. Zhang, M. M. Khalili, and M. Liu, “Improving the privacy and accuracy of ADMM-based distributed algorithms,” in *International Conference on Machine Learning*. PMLR, 2018, pp. 5796–5805.

[10] Z. Xu, G. Taylor, H. Li, M. A. Figueiredo, X. Yuan, and T. Goldstein, “Adaptive consensus ADMM for distributed optimization,” in *International Conference on Machine Learning*. PMLR, 2017, pp. 3841–3850.

[11] C. Hu, J. Jiang, and Z. Wang, "Decentralized federated learning: A segmented gossip approach," *arXiv preprint arXiv:1908.07782*, 2019.

[12] P. Singh, M. Masud, M. S. Hossain, A. Kaur, G. Muhammad, and A. Ghoneim, "Privacy-preserving serverless computing using federated learning for smart grids," *IEEE Transactions on Industrial Informatics*, 2021.

[13] M. Wang, Y. Su, L. Chen, Z. Li, and S. Mei, "Distributed optimal power flow of DC microgrids: A penalty based ADMM approach," *CSEE Journal of Power and Energy Systems*, vol. 7, no. 2, pp. 339–347, 2019.

[14] B. Wang, J. Fang, H. Duan, and H. Li, "Graph simplification-aided ADMM for decentralized composite optimization," *IEEE Transactions on Cybernetics*, vol. 51, no. 10, pp. 5170–5183, 2019.

[15] S. Shethia, A. Gupta, O. Thapliyal, and I. Hwang, "Distributed fast-tracking alternating direction method of multipliers (ADMM) algorithm with optimal convergence rate," in *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2021, pp. 976–981.

[16] F. Rey, Z. Pan, A. Hauswirth, and J. Lygeros, "Fully decentralized ADMM for coordination and collision avoidance," in *2018 European Control Conference (ECC)*. IEEE, 2018, pp. 825–830.

[17] J. Yan, F. Guo, C. Wen, and G. Li, "Parallel alternating direction method of multipliers," *Information Sciences*, vol. 507, pp. 185–196, 2020.

[18] W. Deng, M.-J. Lai, Z. Peng, and W. Yin, "Parallel multi-block ADMM with $O(1/k)$ convergence," *Journal of Scientific Computing*, vol. 71, no. 2, pp. 712–736, 2017.

[19] X. Mao, K. Yuan, Y. Hu, Y. Gu, A. H. Sayed, and W. Yin, "Walkman: A communication-efficient random-walk algorithm for decentralized optimization," *IEEE Transactions on Signal Processing*, vol. 68, pp. 2513–2528, 2020.

[20] Y. Ye, H. Chen, M. Xiao, M. Skoglund, and H. V. Poor, "Privacy-preserving incremental ADMM for decentralized consensus optimization," *IEEE Transactions on Signal Processing*, vol. 68, pp. 5842–5854, 2020.

[21] Y. Ye, H. Chen, Z. Ma, and M. Xiao, "Decentralized consensus optimization based on parallel random walk," *IEEE Communications Letters*, vol. 24, no. 2, pp. 391–395, 2020.

[22] W. J. Nash, T. L. Sellers, S. R. Talbot, A. J. Cawthorn, and W. B. Ford, "The population biology of abalone (haliotis species) in tasmania. i. blacklip abalone (h. rubra) from the north coast and islands of bass strait," *Sea Fisheries Division, Technical Report*, vol. 48, p. p411, 1994.