Latency Aware Distributed ADMM over Networks

Shuai Zeng*, Shuangsheng Wang and Yiming He
School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing, China
* zengshuai@cqupt.edu.cn

Abstract. Methods based on the alternating direction method of multipliers (ADMM) has attracted academic attention because of its excellent convergence performance and potential application scenarios in many machine learning or optimization fields. However, classical distributed ADMM algorithm assumed ideal network communication, which do not consider the impact of network delay on computing performance. In this paper, based on the strategy of selecting bridges with lowest network latency and appropriate iterative process, we propose a latency aware distributed ADMM algorithm to alleviate the impact of network delay. The classical algorithm and proposed algorithm are tested and compared in real network scenarios. Experiments show that the proposed algorithm reduces the running time and improves the computing performance. Especially in networks with large delay, the effect is more obvious.

Keywords. ADMM, machine learning, distributed optimization, network latency

1. Introduction
Methods based on the alternating direction method of multipliers (ADMM) is a simple method to solve the problem of decomposable convex optimization, since Boyd proves that it is suitable for solving large-scale distributed optimization problem [1], this method has attracted academic research, and is widely used in various fields in recent years, including distributed machine learning, image processing, communications network optimization, Power system scheduling, etc. [2-5]

The basic form of ADMM algorithm is as follows:

\[
\min_{x_i} \left( \sum_{i=1}^{N} f_i(x_i) \right) + g(z)
\]

\[\text{s.t. } A_i x_i + B_i z = c_i, \forall i = 1, \ldots, N\] (1)

In the above equation, \(x_i \in R^m, z \in R^n, A_i \in R^{p_i \times m}, B_i \in R^{p_i \times m}, c_i \in R^{p_i}\), the objective function can be separated into several convex functions \(f_i(x_i)\) and \(g(z)\). There are two common architectures for parallel computing. One is the classical C/S architecture, it is a star-like network composed of a master node and several worker nodes. The other architecture is peer-to-peer. In this architecture, there is no master node, and each node has a piece of data, forming a decentralized network. Each node exchanges calculation results with its neighbors, and then achieves global convergence through continuous iteration. The ADMM algorithms implemented by the above two methods are Star-ADMM [6] and decentralized ADMM [7-9] respectively. Centralized networks are the earliest and most widely used networks, but also has very obvious flaws, the bottleneck lies in the center node traffic jams because all nodes need to communicate. Decentralized networks have similar total computational complexity, but the busiest nodes have much lower traffic than centralized ones. Therefore, the algorithms with decentralized
networks is faster than centralized blocks in the case of low bandwidth or high latency [10]. In order to accelerate distributed ADMM, some scholars have proposed many different schemes [11-12], and we are very interested in a semi-centralized algorithm proposed in [7][9]. This method is to select a group of nodes in the network as the master node, and their neighbor nodes as the worker node, this group of master nodes is called bridge nodes, the worker node can only communicate and exchange updates with the specified bridge node. We call it B-ADMM. B-ADMM greatly improves the convergence of the algorithm in the presence of noise, and greatly reduces the communication traffic of the master node in the centralized algorithm. It has excellent performance in the distributed ADMM solution. Of course, the acceleration of decentralized network largely depends on the choice of bridge nodes, but the existing research does not consider the network latency when selecting the bridge node.

Our research is based on B-ADMM and takes network latency into consideration in the selection strategy of bridge nodes and we propose a latency aware distributed ADMM algorithm (Greedy-Bridge-ADMM). Our proposed algorithm can select appropriate bridge nodes according to the latency of a specific network, and we allow the bridge nodes to communicate only with the work nodes. We use a virtual network simulation platform [13] to implement this algorithm and carry out a lot of research. The results show that the proposed algorithm has a faster convergence rate than the traditional algorithms when the network connectivity is poor or the network connection latency and variance cannot be ignored.

In the follow parts of this paper, we analyse the formulation of ADMM problem and propose our algorithm in section 2. In section 3, we compared our algorithm with traditional algorithm under different network conditions in real situation through virtual simulation platform. In section 4, we conclude the paper.

2. Latency Aware Distributed ADMM

2.1. Problem formulation

We target at solving some optimization or machine learning problem over a computer network based on distributed ADMM algorithm. For example, consider solving the following problem over a network.

$$\min f(x) = \sum_{i=1}^{N} f_i(x)$$

(2)

Let a graph $G = (V, E)$ denoted the network, where $V = \{1, ..., n\}$ is the set of nodes and $E = \{1, ..., m\}$ is the set of edges. The Star-ADMM algorithm (the classical distributed ADMM algorithm) for solving (2) is presented in Algorithm 1 as follow.

Algorithm 1 Star-ADMM algorithm (the classical distributed ADMM algorithm)

Form the graph $G$ into a star network with a master node and $n-1$ worker nodes.

for $i \in V$ do
    The master initializes and sends $z_0^i$ to worker $i$.
    Worker $i$ initializes $y_i^0$.
end for

$k = 0$

while accuracy not reached do
    for $i \in V$ do
        The worker $i$ calculate the following formula.
        $$\begin{cases}
        x_i^{k+1} = \arg \min_{x_i} (f_i(x_i) + y_i^{kT} (x_i - z^k) + \frac{\rho}{2} \|x_i - z^k\|_2^2) \\
        y_i^{k+1} = y_i^k + \rho (x_i^{k+1} - z^{k+1})
        \end{cases}$$
        The worker $i$ sends $x_i^{k+1}$ and $y_i^{k+1}$ to the master.
    end for

end while
The master calculates the following formula.

\[ z^{k+1} = \frac{1}{N} \sum_{i=1}^{N} (x_i^{k+1} + (1/\rho) y_i^{k}) \]

for \( i \in V \) do

The master sends \( z^{k+1} \) to worker \( i \).

end for

\( k = k+1 \)

end while

From the iterative process of algorithm 1, we can see that each iteration of the algorithm has to wait for the feedback of all workers. Once the communication delay of nodes in the network is prone to fluctuation, deterioration or abnormality, the algorithm will be greatly limited.

2.2 Proposed algorithm

The basic idea of the proposed algorithm is to split the master node in the Star-ADMM algorithm into multiple bridges. And each bridge has some nodes as its own workers. The proposed algorithm plans these bridges and workers as little delay as possible, and makes all bridges in the computing cluster converge to the final calculation result through some precise iterative calculation. Specifically, the algorithm first selects a worker, and then checks whether it and its every neighbor have a common bridge. If it does not exist, a public bridge is selected according to the principle of minimum network delay. This goes back and forth until each worker and its every neighbor have a common bridge. Then, in the iterative process, each bridge and its workers perform iterative calculation. Because the bridges and workers in the whole computing cluster are cross connected with each other, the optimization or machine learning problem finally converges.

We call the proposed algorithm Greedy-Bridge-ADMM algorithm because the algorithm performs greedy search according to delay when selecting bridges and their workers. The pseudocode of the Greedy-Bridge-ADMM algorithm is shown in Algorithm 2. The symbols used in the algorithm are defined as follows.

\( N(i) \): Set of neighbors of node \( i \).

\( N(i, j) \): Set of common neighbors of node \( i \) and \( j \).

\( N'(i, j) \): Set of common neighbors of node \( i \) and \( j \), including node \( i \) and \( j \).

\( B(i) \): Set of bridges of node \( i \).

\( L(i, j) \): Network delay between nodes \( i \) and \( j \).

\( candidate \): New bridge selected.

\( |B(i)| \): Number of bridges of node \( i \).

**Algorithm 2** Greedy-Bridge-ADMM algorithm (the proposed distributed ADMM algorithm)

for \( i \in V \) do

for \( j \in N(i) \) do

if \( B(i) \cap B(j) = \emptyset \) do

\( N(i, j) = N(i) \cap N(j) \)

\( N'(i, j) = N(i, j) \cup \{i\} \cup \{j\} \)

\( candidate = \arg \min_{k \in N'(i, j)} \{ \max(k \in N'(i, j)), \max(L(i, k)) \} \)

\( B(i) = B(i) \cup \{candidate\} \)

\( B(j) = B(j) \cup \{candidate\} \)

end if

end for

end for
end if
end for
end for
for \( i \in V \)
do
for \( b \in B(i) \)
do
\( b \) initializes and sends \( z_{b,i}^0 \) to worker \( i \).
Worker \( i \) initializes \( y_{b,i}^0 \).
end for
end for
\( k = 0 \)
while accuracy not reached do
for \( i \in V \)
do
for \( b \in B(i) \)
do
The worker \( i \) calculates the following formula.
\[
\phi_{b,i}^k = \frac{1}{|B(i)|} \sum_{b \in B(i)} (x_{b,i}^{k+1} + (1/\rho) y_{b,i}^k)
\]
\( b \) sends \( z_{b,i}^{k+1} \) to their workers.
end for
end for
\( k = k + 1 \)
end while

2.3 Algorithm analysis

As mentioned earlier, the proposed algorithm does not use the master in the Star-ADMM algorithm as a single computing center node, but selects multiple bridges and their workers in the network for calculation respectively. On the one hand, it reduces the computing pressure of the master. On the other hand, more importantly, compared with the master and workers in the Star-ADMM algorithm, the network delays between each bridge and its workers are smaller, which makes the whole computing process not limited to some links with poor delay. In the proposed algorithm, each pair of neighbors ensures that there is at least one common bridge, and the calculation results of each small calculation cluster composed of bridge and its workers are fully integrated in the step of \( y_{b,i}^{k+1} = y_{b,i}^k + \rho (x_{b,i}^{k+1} - z_{b,i}^{k+1}) \) and the step of \( z_{b,i}^{k+1} = (1/|B(i)|) \sum_{b \in B(i)} (x_{b,i}^{k+1} + (1/\rho) y_{b,i}^k) \) in the iterative process, which makes the final overall calculation results converge. At the same time, because the process of each small calculation
cluster composed of bridge and its workers is similar to the Star-ADMM algorithm, its computational complexity and convergence speed are not inferior to the classical distributed ADMM algorithm.

3. Experiments
Experiments run on a full connectivity cluster of 16 nodes on GENI[13]. Experiments setting retain part of the links and their delays to simulate various real network scenarios. The implemented LASSO problem is defined as follow.

\[
\min_{x \in \mathbb{R}^n} \frac{1}{2} \sum_{i=1}^{N} \|A_i x - b_i\|_2^2 + \theta \|x\|_1
\]

Where \(A_i \in \mathbb{R}^{m \times N}, b_i \in \mathbb{R}^m, i = 1, \ldots, N\), and \(\theta > 0\). The elements of \(A_i\) are randomly generated following the Gaussian distribution with zero mean and unit variance; each is generated by \(b_i = A_i x^0 + v_i\), where \(x^0 \in \mathbb{R}^N\) is a sparse random vector with approximately \(0.05N\) non-zero entries and \(v_i\) is a noise vector.

![Figure 1. Running Time](image1.png)

In Figure 1, we compare the real convergence time of the Star-ADMM algorithm and Greedy-Bridge-ADMM algorithm in typical random network. In the figure, the abscissa is the running time(s) and the ordinate is the accuracy. As can be seen from Figure 1, the Greedy-Bridge-ADMM algorithm converges faster and can converge to a reasonable value in about 40 seconds, while the Star-ADMM algorithm needs to converge to a reasonable value in about 80 seconds. The final accuracy of the Greedy-Bridge-ADMM algorithm is slightly worse than that of the Star-ADMM algorithm, but within the acceptable range.

![Figure 2. Performance in Different Scenarios](image2.png)
In Figure 2, we compare the performance of the Star-ADMM algorithm and the Greedy-Bridge-ADMM algorithm in different network scenarios. Specifically, in Figure 2(a), the performance of the algorithm under different network connectivity is described. The abscissa in the figure is the connection probability between any two nodes, and the ordinate is the running time. It can be seen that the running time of the Star-ADMM algorithm and Greedy-Bridge-ADMM algorithm decreases with the increase of connection probability. However, the running time of the Greedy-Bridge-ADMM algorithm is always lower than that of the Star-ADMM algorithm, so the performance is better. At the same time, in Figure 2(b), lines describe the performance of the algorithm under different scenarios of network delay variance. The abscissa in the figure is the probability of the edge with large delay in the network, and the ordinate is the running time of the algorithm. It can be seen that the running time of the Star-ADMM algorithm and Greedy-Bridge-ADMM algorithm increases with the increase of large delay edge. However, the running time of the Greedy-Bridge-ADMM algorithm is always lower than that of the Star-ADMM algorithm, so the performance is better. Moreover, with the increase of large delay edges, the advantages of the Greedy-Bridge-ADMM algorithm are more obvious.

4. Conclusions
Distributed ADMM algorithm has been studied more and more widely because of its excellent convergence performance and potential application in many fields like the emerging machine learning, data analytics, and largescale network system applications. However, the existing research is still lacking, especially ignoring the real network situation such as the latency.

In this paper, we propose an algorithm to construct ADMM networks based on network latency. It selects the bridges according to the principle of minimum network latency on the premise that every worker node and its neighbor have the same bridge node. We call the proposed algorithm Greedy-Bridge-ADMM algorithm. We implemented and evaluated the algorithm in an emulated network. The results show that proposed algorithm achieves real convergence time acceleration with little precision sacrifice, and demonstrate better performance than classical ADMM in different network scenarios, especially in cases of large delay.

Acknowledgment
Part of the research was conducted when Shuai Zeng was a visiting scholar at Renaissance Computing Institute (RENCI) of University of North Carolina at Chapel Hill (UNC). We would like to thank Yufeng Xin and Kush Prasad for the early work. This work was sponsored by Natural Science Foundation of Chongqing (No. cstc2018jcyjA0743, cstc2016jcyjA0560), Science and Technology Research Program of Chongqing Municipal Education Commission (No. KJQN201800640, KJQN201800633), The National Science Foundation of China (No. 61701058), and Foundation of key Laboratory in National Defense Science and Technology for Equipment pre-research in the 13-th Five-year Plan (NO.61422090301).

References
[1] S. Boyd, N. Parikh, E. Chu, B. Peleato and J. Eckstein, "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, 2010.
[2] Zhang, R. & Kwok, J.. (2014). Asynchronous Distributed ADMM for Consensus Optimization. Proceedings of the 31st International Conference on Machine Learning, in PMLR 32(2):1701-1709
[3] J. M. Bioucas-Dias and M. A. T. Figueiredo, "Alternating direction algorithms for constrained sparse regression: Application to hyperspectral unmixing," 2010 2nd Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing, 2010, pp. 1-4, doi: 10.1109/WHISPERS.2010.5594963.
[4] Z. Wang, Z. Lin, T. Lv and W. Ni, "Energy-Efficient Resource Allocation in Massive MIMO-NOMA Networks with Wireless Power Transfer: A Distributed ADMM Approach," in IEEE Internet of Things Journal, doi: 10.1109/JIOT.2021.3068721.

[5] Mohamed A. Mohamed, Hossein Chabok, Emad Mahrous Awwad, Ahmed M. El-Sherbeeny, Mohammed A. Elmeligy, Ziad M. Ali, Stochastic and distributed scheduling of shipboard power systems using MθFOA-ADMM, Energy, Volume 206, 2020, 118041, ISSN 0360-5442.

[6] T.-H. Chang, et al., “Asynchronous distributed admm for large-scale optimization part 1: Algorithm and convergence analysis,” Trans. Sig. Proc, vol. ED-64, no. 12, pp. 3118–3130, June. 2016.

[7] I. D. Schizas, A. Ribeiro, and G. B. Giannakis, “Consensus in ad hoc wsns with noisy links—part 1: Distributed estimation of deterministic signals,” IEEE Transactions on Signal Processing, vol. ED-56, no. 1, pp. 350–364, Jan. 2008.

[8] A. Makhdoumi and A. Ozdaglar, “Convergence rate of distributed admm over networks,” IEEE Transactions on Automatic Control, vol. ED-62, no. 10, pp. 5082–5095, Oct. 2017.

[9] F. Iutzeler, P. Bianchi, P. Ciblat and W. Hachem, "Explicit Convergence Rate of a Distributed Alternating Direction Method of Multipliers," IEEE Transactions on Automatic Control, vol. 61, no. 4, pp. 892-904, April. 2016.

[10] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, “Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent,” In Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS’17, pp. 5336–5346, USA, 2017. Curran Associates Inc.

[11] W. Shi, Q. Ling, K. Yuan, G. Wu and W. Yin, "On the Linear Convergence of the ADMM in Decentralized Consensus Optimization," in IEEE Transactions on Signal Processing, vol. 62, no. 7, pp. 1750-1761, April, 2014, doi: 10.1109/TSP.2014.2304432.

[12] Y. Wang, L. Wu and S. Wang, "A Fully-Decentralized Consensus-Based ADMM Approach for DC-OPF With Demand Response," in IEEE Transactions on Smart Grid, vol. 8, no. 6, pp. 2637-2647, Nov. 2017, doi: 10.1109/TSG.2016.2532467.

[13] Exogeni website and wiki. Available: http://www.exogeni.net.