Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 52-25 (2019) 467–471

Rapid Solution of Linear Equations with Distributed Algorithms over Networks ⋆

Onur Cihan

Department of Electrical and Electronics Engineering, Marmara University, 34722, Göztepe, Istanbul, Turkey, (e-mail: onur.cihan@marmara.edu.tr)

Abstract: In this study, we investigate the problem of accelerating distributed algorithms for solving linear equations over multi-agent networks. While almost all distributed algorithms in the literature assume that the equations are not shared with the neighboring agents, it is shown that the assumption is not restrictive, and an algorithm has been proposed which can be used to determine the equations of the neighbors. We also present a numerical example to illustrate that the convergence rate of a distributed algorithm can be significantly improved by using the proposed algorithm.

Keywords: distributed algorithms, distributed optimization, linear equations, multi-agent networks

1. INTRODUCTION

The problem of solving linear equations has a long history due to its applicability in real world problems such as estimation, prediction and solutions to linear approximations of nonlinear systems (Mehmood and Crowcroft, 2005). While the problem may not seem interesting when all equations are known by a central unit, it may not be possible if the equations arise in different physical locations. Furthermore, the time required to solve a linear equation with a large number of unknowns can be significantly reduced by distributing the workload to multi-processors that have lower processing power. For this purpose, researchers have published studies in the area of parallel computing where a large linear equation system is divided into smaller systems and solved via parallel processors to obtain more precise and faster solutions. On the other hand, recent research in the fields of distributed optimization and distributed consensus have led to a renewed interest in the solution of linear equations. There is a large volume of published studies concerning the design of distributed algorithms that solve linear equations over networks.

Mou and Morse (2013) examined a distributed algorithm for solving linear equations over undirected networks where the update law is defined in discrete-time. It is assumed that each agent in the network knows a single equation of $Ax = b$ and updates its solution estimate by taking the neighboring agents' solutions into account. Liu et al. (2013) showed that transmission delay and asynchronous data transmission do not affect the stability of the given algorithm. Liu et al. (2014) proposed a novel formulation which requires each agent to share a single element of its solution vector at each iteration. In this scenario, the amount of data to be transmitted in the network is significantly reduced. The proposed formulation is shown to solve a given linear equation system in a distributed manner provided that there exists at least one solution and the network graph is repeatedly jointly strongly connected. For the same assumptions, Mou et al. (2015a) utilized matrix norms to show that a distributed algorithm can be used to solve a linear equation exponentially fast. Furthermore, with a slight modification to the algorithm, the estimated solutions converge to the least square solution. Mou et al. (2015b) studied a discrete-time algorithm for solving linear equations where the control input of the agents are left-multiplied by a matrix. It is shown that the linear equation can be solved if the columns of the matrix form a basis for the nullspace of A. The advantage of this algorithm is that the number of components in the state vectors is less than the number of unknowns of the equation.

In all the previously mentioned methods, agents generate initial solutions that satisfy their local equations and update their solutions by adding vectors that are in the nullspace of the matrix A. By this mechanism, estimated solutions of the agents are guaranteed to satisfy the local equations at each time step and the update procedure will continue until estimated solutions of all agents are equal. Wang et al. (2016) proposed an algorithm in which agents start from arbitrary initial estimated solutions that not necessarily satisfy the local equations of the agents. The theoretical analysis of the algorithm is given in (Mou et al., 2016). You et al. (2016) investigated a distributed algorithm where the control law consists of two parts, consensus mechanism and a projector, and show that linear convergence is ensured for networks whose underlying graphs are fixed or uniformly jointly strongly connected. Furthermore, the algorithm is modified so as to solve the Google Pagerank problem in a distributed manner. An algorithm which uses M-Fejer mapping to solve linear equations exponentially fast is given in (Wang et al., 2017a). The algorithm is shown to work for arbitrary

⋆ This work was sponsored by Scientific and Technical Research Council of Turkey (TUBITAK) under grant 117E204.

10.1016/j.ifacol.2019.12.584

initialization and for the cases where the solution is unique or infinitely many solutions exist. Wang et al. (2017b) proposed an initialization procedure which can be used to achieve a solution with minimum Euclidean distance to an arbitrary point. Moreover, it is shown that it is possible to eliminate the initialization procedure in Mou et al. (2015a) with a slight modification to the algorithm. Almost sure convergence in random graphs is studied in Alaviani and Elia (2018).

While the agents share their solutions with the neighboring agents, they are not allowed to share their local equations in all of the studies in the literature. In this paper, we show that this is not restrictive and it is possible to determine the equations of the agents in a finite number of steps by the proposed algorithm. This can be used to increase the convergence speed of the distributed algorithms which is further verified by simulations.

The remainder of the paper is organized as follows. In Section 2, a well-known algorithm in the literature is investigated and a method for determining the equations known by the neighboring agents is presented. This method is formulated as an algorithm and its convergence properties are studied in Section 3. The performance of the proposed algorithm is verified via a numerical example in Section 4. Finally, some concluding remarks are given in Section 5.

2. MATHEMATICAL MODEL

In the last decade, a considerable literature has grown up around the theme of solving linear equations by distributed algorithms over multi-agent networks (Mou and Morse, 2013; Liu et al., 2013, 2014; Mou et al., 2015a,b; Wang et al., 2016; Mou et al., 2016; You et al., 2016; Wang et al., 2017a,b; Alaviani and Elia, 2018; Liu et al., 2018). It is assumed that each agent in the network knows a subset of the linear equation set, i.e., a subset of rows of A and components of b. The subset of equations known by agent i can be represented by the equation $A_i x = b_i$. Most of the proposed algorithms in the literature assumes that the agents generate an initial solution to their local equation ($A_i x_i(0) = b_i$), and adds a vector (that is in the nullspace of A_i) to this solution at each iteration in order to reach the solution of the equation $Ax = b$ ($A \in R^{n\times m}, x \in R^m, b \in R^n$).

In this section, we investigate the following distributed algorithm proposed by Mou et al. (2015a) for solving linear equations over networks

$$x_i(k+1) = x_i(k) - \frac{1}{m_i(k)} P_i \left(m_i(k) x_i(k) - \sum_{j\in\mathcal{N}_i(k)} x_j(k) \right) \tag{1}$$

where $x_i(k)$ is the estimate of agent i in k-th time step and P_i is the orthogonal projection matrix onto $null(A_i)$ which is defined as

$$P_i = I - A_i^T (A_i A_i^T)^{-1} A_i.$$

Furthermore $\mathcal{N}_i(k)$ and $m_i(k) = |\mathcal{N}_i(k)|$ are the set of neighbors and the number of neighbors of agent i at time step k, respectively. This update law ensures that $x_i(k)$ solves the equation $A_i x_i(k) = b_i$ for all $k \geq 1$.

For instance, consider the problem of solving the linear equation system

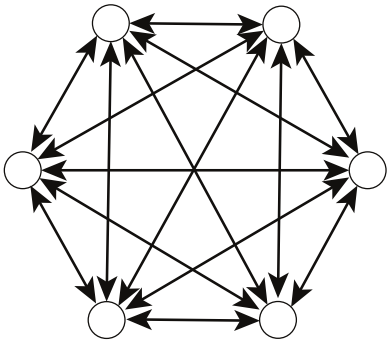

Fig. 1. The complete graph consisting of 6 agents

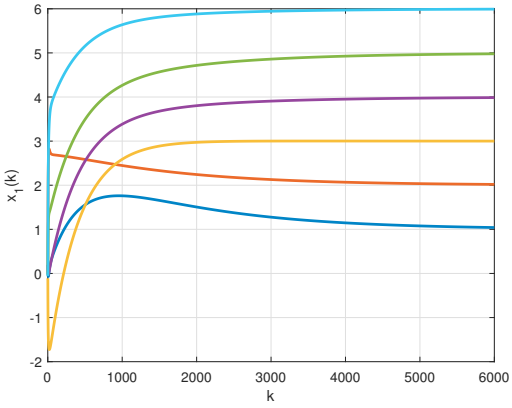

Fig. 2. The evolution of the estimated solutions of Agent 1

$$\begin{array}{rrrrrrl} x_1 & +x_2 & -4x_3 & +7x_4 & -x_5 & -2x_6 & = 2 \\ 2x_1 & -5x_2 & & -x_4 & & -x_6 & = -18 \\ & 3x_2 & -2x_3 & -7x_4 & +4x_5 & +7x_6 & = 34 \\ x_1 & -x_2 & -x_3 & -x_4 & +2x_5 & & = 2 \\ -2x_1 & -2x_2 & +3x_3 & & -6x_5 & +2x_6 & = -15 \\ & & & -3x_4 & -2x_5 & +10x_6 & = 38 \end{array} \tag{2}$$

over the 6-agent network given in Fig. 1. When each agent knows a single equation of (2) and updates its estimate with the update law (1), Fig. 2 illustrates that the estimated solution of Agent 1 converges to the unique solution $x^* = A^{-1}b = [1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6]^T$. The main weakness of the algorithms proposed in the literature is that they require a large amount of data transmission for the agents in the network to obtain the solution in a distributed manner (about 6000 steps in the illustrated example above).

2.1 Determining the equations of the neighboring agents

Almost all of the current literature on the distributed solution of linear equations assumes that the agents are not allowed to share their equations with the neighboring agents due to privacy and security reasons (Yang and Tang, 2015). However, this assumption is not restrictive as illustrated in the following example.

Example 1. Consider a two-agent network with a fully connected graph where the agents seek the solution of the equation

$$\begin{aligned} x_1 + x_2 &= 3 \\ x_1 - x_2 &= -1. \end{aligned}$$

Agent ID	Time step	Estimated solution
A	1	$[3/2,\ 3/2]^T$
A	2	$[5/4,\ 7/4]^T$
A	3	$[9/8,\ 15/8]^T$
B	1	$[-1/2,\ 1/2]^T$
B	2	$[1/4,\ 5/4]^T$
B	3	$[5/8,\ 13/8]^T$

Table 1. Estimates of the agents at $k = 1, 2, 3$

Suppose that each agent knows one of these equations and update their solution estimates by the algorithm (1). Then estimated solutions of the agents for the first 3 time steps are as given in Table 1. In a scenario where the agents do not share their equations with the neighbors, the estimated solutions of Agent A must satisfy

$$a_{11}x_1 + a_{12}x_2 = b_1$$

from the perspective of Agent B. Here, a_{11}, a_{12} and b_1 are the unknowns of the equation. Furthermore, from Table 1, Agent B knows that

$$\frac{3}{2}a_{11} + \frac{3}{2}a_{12} = b_1$$
$$\frac{5}{4}a_{11} + \frac{7}{4}a_{12} = b_1$$
$$\frac{9}{8}a_{11} + \frac{15}{8}a_{12} = b_1$$

hold. Note that these equations can be represented in matrix form as

$$C \begin{bmatrix} a_{11} \\ a_{12} \\ b_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{3}$$

where

$$C = \begin{bmatrix} 3/2 & 3/2 & -1 \\ 5/4 & 7/4 & -1 \\ 9/8 & 15/8 & -1 \end{bmatrix}. \tag{4}$$

For Agent B to determine the equation of Agent A, it is sufficient to determine the nullspace of C. Since the nullspace of C is computed as

$$null(C) = \left\{ k \begin{bmatrix} 1 \\ 1 \\ 3 \end{bmatrix} \ \mid \ k \in R \right\},$$

Agent B can compute the equation of Agent A as $x_1 + x_2 = 3$.

Since C has 3 columns (unknown coefficients and the constant of the equation of Agent A), two linearly independent solutions to the equation $x_1 + x_2 = 3$ are sufficient to compute the nullspace of C and consequently determine the equation of Agent A.

This result is generalized in the following Theorem.

Theorem 1. Suppose that a linear equation set has n unknowns and $p < n$ linearly independent equations. Then the coefficients and the constants of the linear equation set can be determined by using pn linearly independent solutions.

Proof. Let the linear equation set be expressed as

$$\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{p1} & \cdots & a_{pn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_p \end{bmatrix}. \tag{5}$$

Let $y = [a_{11}, \ldots, a_{1n}, \ldots, a_{n1}, \ldots, a_{nn}, b_1, \ldots, b_p]^T \in R^{p(n+1)}$ be the vector consisting of the coefficients and the constants of (5) and $\bar{x}_j \in R^n$ $(j = 1, \ldots, pn)$ be linearly independent solutions to (5). Then, we have

$$Cy = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \tag{6}$$

where

$$C = \begin{bmatrix} \bar{x}_1^T & & & -1 & & \\ & \ddots & & & \ddots & \\ & & \bar{x}_1^T & & & -1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \bar{x}_{pn}^T & & & -1 & & \\ & \ddots & & & \ddots & \\ & & \bar{x}_{pn}^T & & & -1 \end{bmatrix}_{pn \times p(n+1)}. \tag{7}$$

Note that since $\bar{x}_j \in R^n$ $(j = 1, \ldots, pn)$ are linearly independent, we have $rank(C) = pn$. From Rank-Nullity Theorem (Horn and Johnson, 2012), we conclude that $nullity(C) = p(n+1) - pn = p$ and the set of vectors that span $null(C)$ are the coefficients and the constants of p equations.

3. MAIN RESULTS

As stated in Section 2.1, it is possible to determine the equations of neighboring agents from their solution estimates. In this section, we propose an algorithm with an equation detection phase which can be used to rapidly solve linear equations over networks with fixed topology.

Algorithm 1 Algorithm for solving linear equations over networks with fixed topology

$P_i \leftarrow I - A_i^T(A_iA_i^T)^{-1}A_i$
$x_i(1) \leftarrow A_i^T(A_iA_i^T)^{-1}b_i$
for each $j \in \mathcal{N}_i$ **do**
  Initialize X_j as an empty 2D array
end for
for $k := 1$ **to** pn **do**
  for each $j \in \mathcal{N}_i$ **do**
    Append the rows of $[I_p \otimes x_j(k)^T \ \ -I_p]$ to X_j
    $\bar{w}_{ij} \leftarrow$ A random number between 0 and 1
  end for
  for each $j \in \mathcal{N}_i$ **do**
    $w_{ij} \leftarrow \bar{w}_{ij} / \sum_i w_{ij} = 1$
  end for
  $x_i(k+1) \leftarrow x_i(k) - P_i \left(x_i(k) - \sum_{j \in \mathcal{N}_i} w_{ij} x_j(k) \right)$
end for
$\bar{A} \leftarrow A_i$
$\bar{b} \leftarrow b_i$

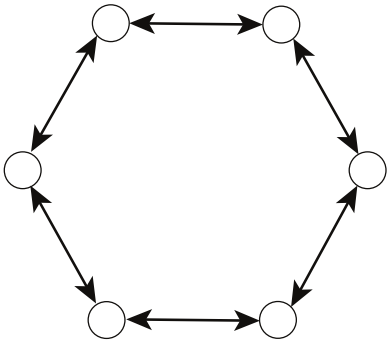

Fig. 3. An undirected graph consisting of 6 agents

Algorithm 1 Algorithm for solving linear equations over networks with fixed topology (continued)

for each $j \in \mathcal{N}_i$ **do**
  Compute $\bar{A}_j$ whose columns span $null(X_j)$
  Append the first n columns of $\bar{A}_j$ as new rows of $\bar{A}$
  Append the last column of $\bar{A}_j$ to $\bar{b}$
end for
$A_i \leftarrow \bar{A}$
$b_i \leftarrow \bar{b}$
$P_i \leftarrow I - A_i^T(A_iA_i^T)^{-1}A_i$
$x_i(1) \leftarrow A_i^T(A_iA_i^T)^{-1}b_i$
$k \leftarrow 1$
Receive new solutions of the neighbors
repeat
  $x_i(k+1) \leftarrow x_i(k) - P_i\left(x_i(k) - \sum_{j\in\mathcal{N}_i} w_{ij}x_j(k)\right)$
  $k \leftarrow k+1$
until $||x(k+1) - x(k)|| < \epsilon$

Theorem 2. Given an undirected and connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, Algorithm 1 solves a linear equation set of the form $Ax = b$ in a distributed manner if the union of the known equations by agents is equal to the equation set.

Proof. In Algorithm 1, each agent collects the solutions of its neighbors for pn steps and determines the equations of the neighbors. Note that the solutions of the neighbors are almost surely linearly independent since the weighting coefficients of the update algorithm are randomly chosen at each time step. The equations known by the agents, together with the equations of the neighbors, are used to generate new initial solutions and new projection matrices P_i. Since the remaining steps of the Algorithm 1 solve a linear equation set with the given assumptions by Theorem 1 of (Mou et al., 2015a), we conclude that Algorithm 1 solves a linear equation set over an undirected network.

4. SIMULATION RESULTS

Reconsider the problem of distributed solution of the linear equation system

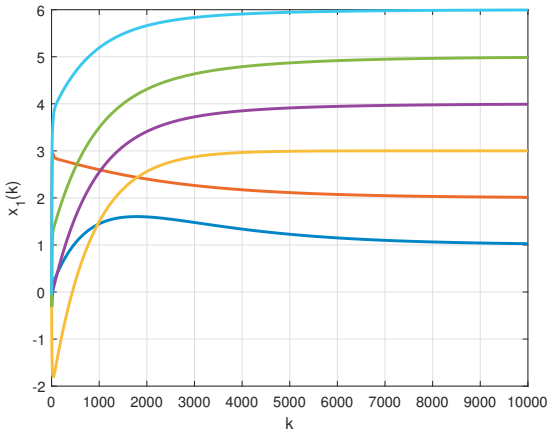

Fig. 4. The evolution of the estimated solutions of Agent 1 when (1) is used

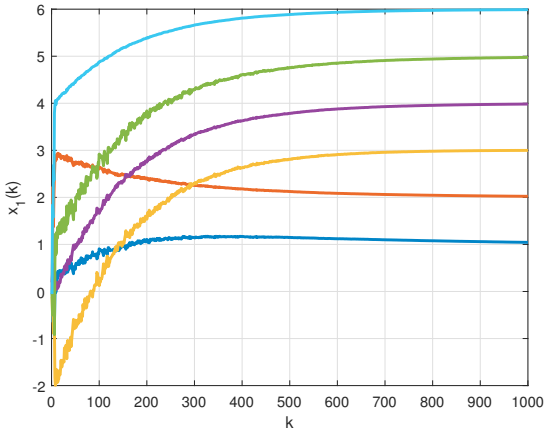

Fig. 5. The evolution of the estimated solutions of Agent 1 when Algorithm 1 is used

$$\begin{array}{rrrrrrl} x_1 & +x_2 & -4x_3 & +7x_4 & -x_5 & -2x_6 & = 2 \\ 2x_1 & -5x_2 & & -x_4 & & -x_6 & = -18 \\ & 3x_2 & -2x_3 & -7x_4 & +4x_5 & +7x_6 & = 34 \\ x_1 & -x_2 & -x_3 & -x_4 & +2x_5 & & = 2 \\ -2x_1 & -2x_2 & +3x_3 & & -6x_5 & +2x_6 & = -15 \\ & & & -3x_4 & -2x_5 & +10x_6 & = 38 \end{array}$$

over the network shown in Fig. 3. Suppose that each agent in the network knows a single equation of the system, for instance, i-th agent knows i-th equation. Fig. 4 illustrates the evolution of the estimated solutions of Agent 1 when the update equation (1) is used. On the other hand, when the agents update their estimates by Algorithm 1, the evolution of the estimated solutions of Agent 1 are as depicted in Fig. 5.

As can be seen from Figs. 4 and 5, the number of required steps (and consequently the amount of required energy for transmission) to achieve the desired accuracy in the solution estimates is reduced by 90% when Algorithm 1 is used.

5. CONCLUSION

In this paper, we studied the distributed solution of a linear equation over networks. It is shown that when agents

share their solutions with their neighbors, it is possible to reconstruct the equations of the neighbors by the proposed algorithm. Once the equations of the neighbors are computed, agents can re-initialize their estimates and the projection matrices for a faster convergence to the solution. The performance of the proposed algorithm is also verified by a numerical example.

REFERENCES

Seyyed Shaho Alaviani and Nicola Elia. A distributed algorithm for solving linear algebraic equations over random networks. In *2018 IEEE Conference on Decision and Control (CDC)*, December 2018.

Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, New York, NY, USA, 2012.

Ji Liu, Shaoshuai Mou, and A. Stephen Morse. An asynchronous distributed algorithm for solving a linear algebraic equation. In *52nd IEEE Conference on Decision and Control*, December 2013.

Ji Liu, Xiaobin Gao, and Tamer Başar. A communication-efficient distributed algorithm for solving linear algebraic equations. In *2014 7th International Conference on NETwork Games, COntrol and OPtimization (NetGCoop)*, pages 62–69, Oct 2014.

Ji Liu, Shaoshuai Mou, and A. Stephen Morse. Asynchronous distributed algorithms for solving linear algebraic equations. *IEEE Transactions on Automatic Control*, 63(2):372–385, 2018.

Rashid Mehmood and Jon Crowcroft. Parallel iterative solution method of large sparse linear equation systems. Technical report, University of Cambridge, UK, 2005.

Shaoshuai Mou and A. Stephen Morse. A fixed-neighbor, distributed algorithm for solving a linear algebraic equation. In *2013 European Control Conference (ECC)*, July 2013.

Shaoshuai Mou, Ji Liu, and A. Stephen Morse. A distributed algorithm for solving a linear algebraic equation. *IEEE Transactions on Automatic Control*, 60(11): 2863–2878, 2015a.

Shaoshuai Mou, A. Stephen Morse, Z. Lin, Lili Wang, and Daniel Fullmer. A distributed algorithm for efficiently solving linear equations. In *2015 54th IEEE Conference on Decision and Control (CDC)*, December 2015b.

Shaoshuai Mou, Zhiyun Lin, Lili Wang, Daniel Fullmer, and A. Stephen Morse. A distributed algorithm for efficiently solving linear equations and its applications (special issue JCW). *Systems & Control Letters*, 91: 21–27, 2016.

Lili Wang, Daniel Fullmer, and A. Stephen Morse. A distributed algorithm with an arbitrary initialization for solving a linear algebraic equation. In *2016 American Control Conference (ACC)*, July 2016.

Peng Wang, Wei Ren, and Zhisheng Duan. Distributed solution to linear equations from arbitrary initializations. In *2017 American Control Conference (ACC)*, May 2017a.

Xuan Wang, Shaoshuai Mou, and Dengfeng Sun. Improvement of a distributed algorithm for solving linear equations. *IEEE Transactions on Industrial Electronics*, 64(4):3113–3117, 2017b.

Mu Yang and Choon Yik Tang. A distributed algorithm for solving general linear equations over networks. In *2015 54th IEEE Conference on Decision and Control (CDC)*, December 2015.

Keyou You, Shiji Song, and Roberto Tempo. A networked parallel algorithm for solving linear algebraic equations. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, December 2016.