Lecture: Faster Convergence for Distributed Algorithms: LTI Case

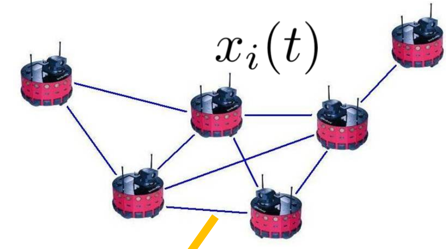
Shaoshuai Mou

PURDUE UNIVERSITY

Distributed Algorithm for Consensus

$$x_i(t+1) = \sum_{j \in \mathcal{N}_i} w_{ij} x_j(t)$$

$$w_{ij} = \begin{cases} > 0, \; j \in \mathcal{N}_i \\ 0, \; \text{otherwise.} \end{cases} \qquad \sum_{j=1}^{m} w_{ij} = 1$$



$$\boxed{x(t+1) = Ax(t)}$$

Graph of A is strongly connected and aperiodic

A is row stochastic

If A is also Primitive

Gershgorin Circle Theorem

Perron - Frobenius Theorem

$A\mathbf{1} = \mathbf{1}$

1 is the largest eigenvalue in magnitude.

1 is a simple eigenvalue

all the other eigenvalues are with magnitude strictly less than 1

$$x(t) \to \mathbf{1}w'x(0) \text{ as fast as } |\lambda_2(A)|^t \to 0$$

➢ **Distributed Algorithm for Consensus:** $x(t+1) = Ax(t)$

Given the network to be connected, one has

$$x(t) \to \mathbf{1}w'x(0) \text{ as fast as } |\lambda_2(A)|^t \to 0$$

Smaller $|\lambda_2(A)|$ is, faster the convergence is.

$$x_i(t+1) = \sum_{j \in \mathcal{N}_i} w_{ij} x_j(t)$$

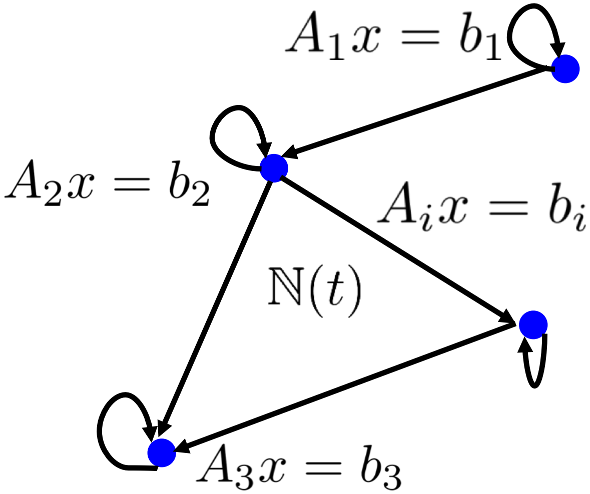
- For a given rule of choosing weights, one could

$$\min |\lambda_2(A)| \quad \text{for all connected networks}$$

- For a given network, one could

$$\min |\lambda_2(A)| \quad \text{for all possible weights.}$$

When a network is given, metropolis weights are determined. Thus distributed algorithms for averaging is fixed. And thus the matrix A is fixed. How should we improve the convergence rate?



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

$$x(t+1) = Ax(t), \quad A = (I - P + S)$$

- Analysis: **Error Dynamics** $\quad e_i(t) = x_i(t) - x^*$

$$e_i(t+1) = P_i \frac{1}{d_i(t)} \sum_{j \in \mathcal{N}_i(t)} P_j e_j(t)$$

$$e(t+1) = P(S_t \otimes I_n) Pe(t)$$

$$P = \mathtt{diag}\{P_1, P_2, ..., P_m\}$$

$$S_t = D_{\mathbb{N}(t)}^{-1} A_{\mathbb{N}(t)}$$

Case I: Fixed Undirected Graph

$$e(t+1) = P\bar{S}e(t)$$

To prove $e(t) \to 0$, it is sufficient to show $\rho(PS) < 1$

- *Are all eigenvalues real??*
- *Are all eigenvalues in the interval (-1,1]?*
- *Prove 1 is not an eigenvalue of PS by contradiction*

Key Idea to **Accelerate** LTI

$$x(t+1) = Ax(t)$$

- Introduce one additional memory and utilize x(t-1)

$$x(t+1) = \gamma A x(t) + (1-\gamma) x(t-1)$$

γ is a adjustable parameter to achieve faster convergence

- *Successive over-relaxation (SOR) Method:* $\gamma \in (0,2)$
- This idea can also be extended for acceleration of nonlinear, iterative update

$$x(t+1) = f(x(t)) \Longrightarrow x(t+1) = \gamma f(x(t)) + (1-\gamma) x(t-1)$$

- If more memories are available, one could achieve **finite-time** convergence

- **Distributed Average Consensus Algorithm (DACA):**

$$x_i(t+1) = \sum_{j \in \mathcal{N}_i} w_{ij} x_j(t)$$

metropolis weights

$$w_{ij} = \begin{cases} \min\{\frac{1}{d_i}, \frac{1}{d_j}\} & j \in \mathcal{N}_i,\ j \neq i; \\ 1 - \sum_{j \in \mathcal{N}_i,\ j \neq i} w_{ij} & j = i \\ 0, \text{ otherwise.} \end{cases}$$

$$x(t+1) = Ax(t)$$

- **Accelerated, Distributed Average Consensus Algorithm Fast (A-DACA)**

  - Introduce one additional memory and utilize x(t-1)

$$x_i(t+1) = \gamma \sum_{j \in \mathcal{N}_i} w_{ij} x_j(t) + (1-\gamma) x_i(t-1)$$

$$x(t+1) = \gamma Ax(t) + (1-\gamma)x(t-1) \qquad x(-1) = x(0)$$

If $\gamma \geq 2$ or $\gamma \leq 0$ not converge.

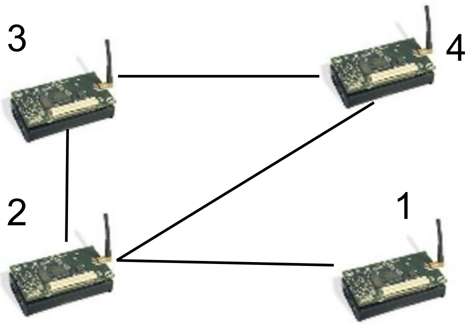
One can choose a value from the interval (0,2). For example,

$$\gamma = \frac{2}{1 + \sqrt{1 - |\lambda_2(A)|^2}}$$

which has been proved to lead to faster convergence in the following.

* S. Muthukrishnan, B. Ghosh, M. H. Schultz. First- and Second-Order Diffusive Methods for Rapid, Coarse, Distributed Load Balancing. Theory of Computing Systems. 1998

Example: One randomly initialize x(0), and compare convergence between DACA and A-DACA.



DACA: $x_i(t+1) = \sum_{j \in \mathcal{N}_i} w_{ij} x_j(t)$ $\qquad x(t+1) = Ax(t)$

A-DACA: $x_i(t+1) = \gamma \sum_{j \in \mathcal{N}_i} w_{ij} x_j(t) + (1-\gamma) x_i(t-1)$

$$x(t+1) = \gamma A x(t) + (1-\gamma) x(t-1)$$

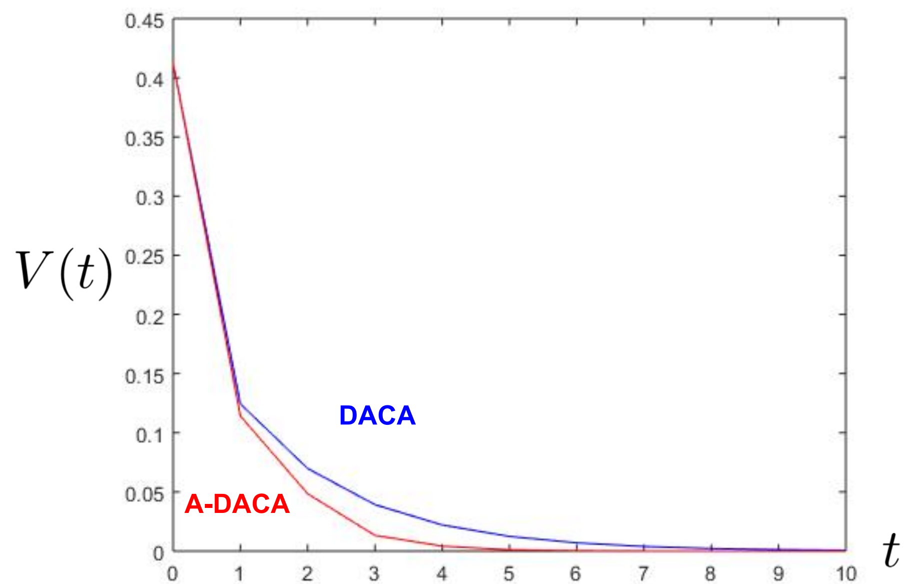
$$A = \begin{bmatrix} 3/4 & 1/4 & 0 & 0 \\ 1/4 & 1/4 & 1/4 & 1/4 \\ 0 & 1/4 & 5/12 & 1/3 \\ 0 & 1/4 & 1/3 & 5/12 \end{bmatrix} \qquad |\lambda_2(A)| = 0.75 \qquad \gamma = 0.7515$$

Error Function: $V(t) = ||x(t) - x^* \mathbf{1}||_2^2$

measure the closeness between all agents' states away from the global average.

$$V(t) \geq 0 \qquad\qquad x^* = \frac{1}{4} \mathbf{1}' x(0)$$

with equality holds if any only if all agents states reach the desired value.



What if each agent has more memories?

Each agent *i* is able to store a number M states

$$x_i(t), x_i(t-1), x_i(t-2), ..., x_i(t-M+1)$$

How to achieve a even faster algorithm?

- A distributed algorithm using idea similar to SOR:

$$x_i(t) = \gamma x_i^P(t) + (1-\gamma) x_i^W(t)$$

$$x_i^W(t) = w_{ii} x_i(t-1) + \sum_{j \in \mathcal{N}_i} w_{ij} x_j(t-1)$$

$$x_i^P(t) = \theta_M x_i^W(t) + \sum_{j=1}^{M-1} \theta_j x_i(t-M+j)$$

- T. C. Anysal, B. N. Oreshkin and M. J. Coates. Accelerated Distributed Average Consensus via Localized Node State Prediction. IEEE Transactions on Signal Processing, 57(4), 2009.

*Intuitively, more memories, faster convergence one could achieve, till **Finite-Time** Convergence*

➢ Finite-Time Distributed Average Consensus Algorithm (FT-DACA)

$$x(t+1) = Ax(t)$$

Assumption: Each agent knows the characteristic polynomial of the update matrix A

$$\det(\lambda I - A) = \lambda^n + c_{n-1}\lambda^{n-1} + \cdots + c_1\lambda + c_0$$

✓ **Cayley-Hamilton Theorem.**

$$A^n x(0) + c_{n-1}A^{n-1}x(0) + \cdots + c_1 A x(0) + c_0 I x(0) = 0$$

$$x(k) = A^k x(0)$$

$$x(t+n) + c_{n-1}x(t+n-1) + c_{n-2}x(t+n-2) + \cdots + c_1 x(t+1) + c_0 x(t+0) = 0$$

$$x_i(t+n) + c_{n-1}x_i(t+n-1) + c_{n-2}x_i(t+n-2) + \cdots + c_1 x_i(t+1) + c_0 x_i(t) = 0$$

After n steps, each agent i has $x_i(0), x_i(1), x_i(2), ..., x_i(n-1)$

then each agent could only employ Cayley-Hamilton Theorem to compute

$$x_i(n+t) \text{ for } t = 1, 2, 3, ...$$

no further communications with its neighbors after t=n.

Each agent *i* store its own states $x_i(0), x_i(1), x_i(2), ..., x_i(n)$,

and can compute $x_i(t), t = n, n+1, ..., \infty$ *iteratively* by.

$$x_i(t+n) + c_{n-1}x_i(t+n-1) + c_{n-2}x_i(t+n-2) + \cdots + c_1 x_i(t+1) + c_0 x_i(t) = 0$$

How to achieve $x_i(\infty)$ **in one step**?

Z-Transform $X(z) = \sum_{t=0}^{\infty} x(t) z^{-t}$

Try by yourself.

$x(t)$	$X(z)$
$x(t+N)$	$z^N X(z) - \sum_{k=0}^{N-1} x(k) z^{N-k}$

$$\left(z^n X_i(z) - \sum_{k=0}^{n-1} x_i(k) z^{n-k}\right) + c_{n-1}\left(z^{n-1} X_i(z) - \sum_{k=0}^{n-2} x_i(k) z^{n-1-k}\right)$$

$$+ \cdots + c_1\left(zX_i(z) - zx_i(0)\right) + c_0 X_i(z) = 0$$

$$\left(z^n + c_{n-1}z^{n-1} + \cdots + c_1 z + c_0\right) X_i(z) =$$

$$\sum_{k=0}^{n-1} x_i(k) z^{n-k} + c_{n-1}\sum_{k=0}^{n-2} x_i(k) z^{n-1-k} + \cdots + c_1 z x_i(0)$$

$$X_i(z) = \frac{\sum_{k=0}^{n-1} x_i(k) z^{n-k} + c_{n-1}\sum_{k=0}^{n-2} x_i(k) z^{n-1-k} + \cdots + c_1 z x_i(0)}{z^n + c_{n-1}z^{n-1} + \cdots + c_1 z + c_0}$$

Characteristic polynomial of A

- For distributed consensus/averaging $x(t+1) = Ax(t)$, one has A has a simple eigenvalue at 1, and all the other eigenvalues are with magnitude strictly less than 1

the dominator= $(z-1)p(z)$

$$p(z) = z^{n-1} + (1+c_{n-1})z^{n-2} + (1+c_{n-1}+c_{n-2})z^{n-3} + \cdots + (1+\sum_{k=2}^{n-1} c_k)z + (1+\sum_{k=1}^{n-1} c_k)$$

- By the **Final Value Theorem**, one has

$$\lim_{t\to\infty} x_i(t) = \lim_{z\to 1}(z-1)X_i(z)$$

$$= \frac{\begin{bmatrix} x_i(n-1) & x_i(n-2) & \cdots & x_i(1) & x_i(0)\end{bmatrix} S}{\begin{bmatrix} 1 & 1 & \cdots & 1 & 1\end{bmatrix} S}$$

Characteristic Polynomial of A

$$\det(\lambda I - A) =$$

$$\lambda^n + c_{n-1}\lambda^{n-1} + \cdots + c_1\lambda + c_0$$

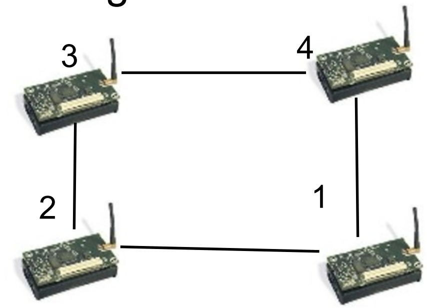
$$S = \begin{bmatrix} 1 \\ 1 + c_{n-1} \\ 1 + c_{n-1} + c_{n-2} \\ \vdots \\ 1 + \sum_{j=1}^{n-1} c_j \end{bmatrix}$$

Global Information.

Any way to achieve this in a distributed way?

- S. Sundaram, C. N. Hadjicostis. Finite-Time Distributed Consensus in Graphs with Time-Invariant Topologies. Proceedings of American Control Conference, 2007.

Other Ways to Achieve Finite-Time Convergence for Distributed Average



- **Matrix Factorization**

  - C. K. Ko, X. Gao. On matrix factorization and finite-time average-consensus. Proceedings of the 48th IEEE Conference on Decision and Control. 2009

A specific types of periodic-gossiping: (1,4), (2,3), (1,2), (3,4) achieves the average finite-time (one-period). Try by yourself.

$$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1/2&1/2\\0&0&1/2&1/2\end{bmatrix}\begin{bmatrix}1&0&0&0\\0&1/2&1/2&0\\0&1/2&1/2&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}1/2&1/2&0&0\\1/2&1/2&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}1/2&0&0&1/2\\0&1&0&0\\0&0&1&0\\1/2&0&0&1/2\end{bmatrix}=\begin{bmatrix}1/4&1/4&1/4&1/4\\1/4&1/4&1/4&1/4\\1/4&1/4&1/4&1/4\\1/4&1/4&1/4&1/4\end{bmatrix}$$

Such factorization is only applicable to specific types of networks.

- **Push-Sum Idea**

$$x^* = \frac{x_1(0)+x_2(0)+\cdots+x_m(0)}{m}$$

sum of initial states

number of agents

$$x_i(t+1)=\begin{cases} x_i(0)+\sum_{j\in\mathcal{N}_i} x_j(0), & t=0;\\ \sum_{j\in\mathcal{N}_i} x_j(t)+(1-r_i)x_i(t-1), & t\geq 1.\end{cases}$$

- Given the network to be a **tree** graph, the distributed update is able to achieve the sum in a number of $d_{\mathbb{G}}$ steps

- Achieve the number of agents in a distributed way.

Introduce one additional state at each agent with $z_i(0)=1$

$$m = z_1(0)+z_2(0)+\cdots+z_m(0)$$

- S. Mou, A. S. Morse. Finite-time Distributed Averaging. Proceedings of American Control Conference, 2014.

Please Stay Safe at Home!

Feel free to contact me if you have any questions about the course or your projects. mous@purdue.edu