論文紹介：When are Distributed Algorithms Robust ?

FL06-23-1

Azwirman Gusrialdi

Today's Talk

- Introduction
- Basic Framework (definition of controlled agents, network, failures, robustness, etc)
- Example (Average Consensus)
- How to make Algorithms Robust (byzantine general problems).

Introduction

- Distributed Algorithm : Average-Consensus, rendezvous, sensor coverage, deployment. Etc
- Involving many agents which are cooperating ⇒to get a greater accuracy
- Huge numbers of Agents ⇒components will be cheap ⇒high failures rates
- Is the task still satisfactorily executed even when some agents fail to communicate or perform correctly ?

Definition I : A controlled agent

1. $x(k) \in X$ is the state
2. $u(k) \in U$ is the control input
3. $x(0) \in X(0)$ is the initial condition
4. $f: X \times U \rightarrow X$ is a map defines dynamics

ie. $x(k+1)=f(x(k),u(k))$ or

$x(k+1)=A(k)x(k)+B(k)u(k)$

Definition II : Network of controlled agents

1. $I=\{1,\ldots,N\}$ is the set of unique identifiers
2. $A=\{A_i\}_{i\in I}$ is the set of controlled agents
3. G_{comm} is the set of allowed communication graph. Assume the undirected graphs (agent *i* is the neighbor of agent *j* if the two can communicate.)
4. Additional environmental variable:V (eg. Location of obstacles)

Definition III : Cooperative task

$$C: \prod_i \{x_i(k)\}_{k=0}^{\infty} \times \prod_i \{u_i(k)\}_{k=0}^{\infty} \times \prod_i x_i(0) \times V \mapsto \Re^+$$

The aim of any algorithm that carries out the task is to minimize the cost function

Definition IV : Cooperative Algorithm

⇒ is a choice of communication and control laws for every agent.

Basic Framework

- Average Consensus: the task is to ensure each agent has the value m (arithmetic mean)
- Dynamics : $x_i(k+1)=x_i(k)+u_i(k)$
- Cost Function :

$$C = \lim_{k\to\infty}\left(\sum_{\text{all nodes i}}\left(x_i(k) - \frac{1}{N}\sum x_i(0)\right)^2\right)$$

- Control input :

$$u_i(k) = -h\sum_{\text{j:j is a neighbor of i}}\left(x_i(k) - x_j(k)\right)$$

Basic Framework

- One way to characterize an algorithm is through the value of the task cost function C that it achieves.
- PC:Performance Cost by the algorithm ⇒**can be a function of x(0) and V.**
- Average Cost, PC_{avg} ⇒**averaging PC as x(0) and V are chosen from a given set S.**
- Worst Case, PC_{wc} ⇒**compute the supremum of PC.**
- PC also depends on the number of agents, $PC_{avg}(N)$

Basic Framework

When an agent fails, it alters the control law and the communication law that it follows. Failure modes can be defined as follows :

- Failure mode 1 : An agent may fail by simply ceasing to communicate with other agents.
- Failure mode 2 : An agent fails by setting its state value $x_i(k)$ to a constant in the set X_i.
- Failure mode 3 : The agent alters the control input to set its state at every time step k to an arbitrary value in the set X_i.

Definition V : Robustness of an Algorithm

⇒An algorithm is said to be worst-case robust to a particular failure up to p agents if

$$PC_{wc}(N,p) = O\left(PC_{wc}(N-p)\right)$$

Note : $f(x)=O(g(x))$ iff there exists numbers x_0 and M >0 s.t. $|f(x)| \le M|g(x)|$ For $x>x_0$

- If $PC_{wc}(N,p) = \Omega(PC_{wc}(N-p))$ but $PC_{wc}(N,p) \neq \Theta(PC_{wc}(N-p))$

The algorithm is said to be worst-case non-robust

Basic Framework

- Worst case robustness ⇒tells us if the algorithm will perform correctly for any set of initial condition.
- Average case robustness ⇒guarantees that the algorithm will perform correctly on an average.
- Almost sure robustness ⇒worst case robust except on a region with measure zero.

Proposition I :

If an algorithm is non-robust for p failed agents to failure mode 2, it is non robust to p failed agents to mode 3.

Proof :

Let the control input used in the calculation of PC for failure mode 3(2) be given by $\{u_i(k)\}_{3(2)}$ for agent I and messages sent be given by $\{m_i(k)\}_{3(2)}$. Consider the choice of the control inputs. The set in which the control inputs are allowed to vary for mode 3 also contains as a particular element $\{u_i(k)\}_2$.

Proof :

Since by definition, the cost in mode 3 is maximized by $\{u_i(k)\}_3$;in particular, the cost achieved by using $\{u_i(k)\}_2$ is not more than when $\{u_i(k)\}_2$ is used. Thus,

$$PC_{wc}(N,p)_{\text{failure mode 3}} \geq PC_{wc}(N,p)_{\text{failure mode 2}}$$

If the algorithm is non-robust to failure mode 2, there exists a constant c s.t.

$$PC_{wc}(N,p)_{\text{failure mode 2}} \geq cPC_{wc}(N)$$

Proposition II :

If an algorithm is non-robust to failure of p agents in failure mode 3, it is also non-robust to failure of t agents in failure mode 3 where $t \geq p$.

Proof :

Consider the case when p agents fail and the choice of init. Conditions, control inputs and messages for failed agents that corresponds to the worst case of PC. Choose a set S of t-p func. Agents which control inputs and messages are $\{u_i(k)\}$ and $\{m_i(k)\}$. Now, consider the case when t agents can fail.

Basic Framework

Proof :

Choose the same init. Cond. as the previous case. Let the t agents that fail be chosen s.t. they consist of the p agents that failed in the previous case and t-p agents in the set S. Also, let the p agents apply the same control inputs and transmit the same messages as the previous case. Thus, the evolution of the system will be identical to the case when only p agents failed. Hence,

$$PC_{wc}\left(N, t\right) \geq PC_{wc}\left(N, p\right)$$

Example (Average Consensus)

- Since the algorithm requires connected graphs, we will assume that to be the case as long as no agents fail.
- Consider that the initial conditions to be chosen uniformly over the set [-1,1].
- Assume that p agents that fail are allowed to be chosen so that the graph of the remaining N-p agents is disconnect. Then the algorithm is worst case non-robust to failure mode 1. If the graph remains connected, then the algorithm is worst-case, average case and a.s. robust to failure mode 1.

Example (Average Consensus)

Proof :

Consider the case that we allow the graph of the remaining agents to be potentially disconnected. Let p=1, N=2m+1 and choose the graph of N agents as a line. Let the agent I fail s.t two distinct connected sub-groups of agents are formed, each with m agents. Also, the initial cond. Are chosen s.t. every agent in the first group has value 1 and in the other group has value –1. Thus,

$$PC_{wc}(N,1) \geq \sum_{i=1}^{m}(1)^2 + \sum_{i=1}^{m}(-1)^2 = N-1$$

Example (Average Consensus)

Proof :

$PC_{wc}(N)=0$ as long as the graph was connected. Thus, the algorithm is worst case non-robust. If the graph remains connected, $PC_{wc}(N,p)=PC_{wc}(N)=0$. Hence, the algorithm is worst case robust.

- The algorithm is worst-case, average-case and a.s non-robust to failure mode 2.

Proof :

Consider the case when p=1. Let the initial cond. be s.t. the non-faulty N-1 agents have values 0 while, the faulty agent has value 1.

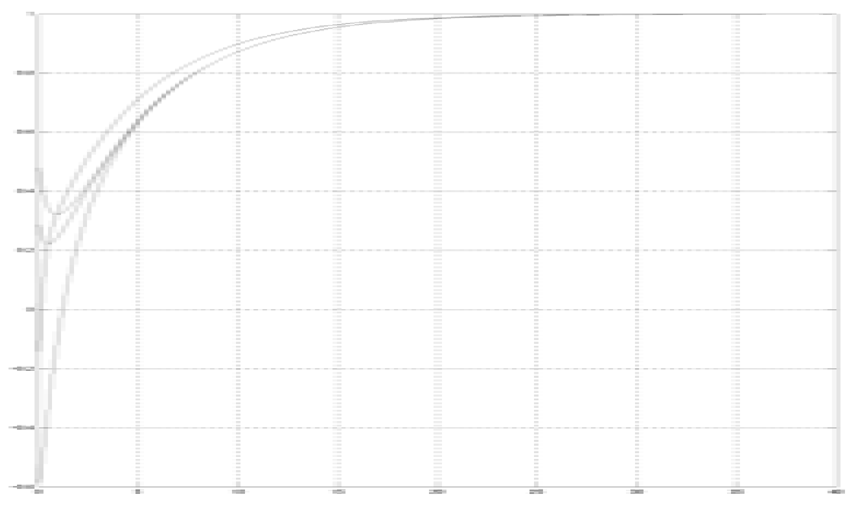
Example (Average Consensus)

Proof :

Thus, the algorithm will converge with each agent achieving the value 1, as against converging to the correct mean for N-1 agents, which is 0. Thus,

$$PC_{wc}\left(N,1\right) \le \sum_{i=1}^{N}\left(1-0\right) = N$$

Since $PC_{wc}(N)=0$, the algorithm is non-robust.



How to make Algorithm Robust

Byzantine Generals problem :

- A general needs to transmit a value v to N commanders s.t. when the algorithm terminates,

1. All the functional(loyal) commanders make the same decision about the value. The final value of the non-loyal commanders is not concerned.
2. If the General is functional, all functional commanders receive the correct value.

- Assume that the General is Functional.

How to make Algorithm Robust

- Consider the Cost

$$C = \lim_{k\to\infty} \sum_{i=1}^{N} \left(x_i(k) - v \right)^2$$

- x_i is the final decision of the i-th loyal commander and N is the number of loyal commanders.
- Let’s study the robustness properties of three algorithms that solve the problem.
- In the First one, assumed the general to be node 1 whose state remains at v. Every other agent updates its state as

How to make Algorithms Robust

$$x_i(k+1) = x_i(k) - h\sum_{j \neq i}\left[x_i(k) - x_j(k)\right]$$

- The algorithm solves for all agents are functional.
- When p agents fail according to failure mode 2, as long as a node has a path from the failed agent that does not include the general, it does not converge to the value v. Thus the algorithm is worst-case and average-case non-robust for any p.
- It is also non-robust for failure mode 3.

How to make Algorithms Robust

- In the second algorithm, assume that less than one-third agents fail. For simplicity, the comm.graph is assumed to be fully connected.
- The algorithm proceeds as :

1. At time step 1, the general transmits its value v to all the commanders.
2. At time step 2, every commander transmits its estimation to every other commander.
3. At time step 3, every commander calculates a majority of what it has heard and outputs its estimate of the decision.

How to make Algorithms Robust

- The algorithm is both worst-case and average-case robust to failure mode 3.
- The third algorithm involves including a fault detection step in algorithm 1.
- For node i, let Ni be the neighbor set of i. When agent I communicates with agent j at time k, ot transmits 4 quantities: $x_i(k)$ (denoted by $a_i(k)$), $x_i(k-1)$ (denoted by $b_i(k)$), $\sum_{l \in N_i} x_l(k-1)$(denoted by $d_i(k)$) and N_i.

How to make Algorithms Robust

- Given these quantities, each node carries out the following checks :

1. It checks if $a_i(k-1)=b_i(k)$.
2. It checks if $a_i(k)=(1-hN_i)b_i(k)+hN_id_i(k)$.

- If both checks are successful, it carries out the same step as average consensus algorithm, otherwise it identifies the node i as faulty agents and disregards it from that time on.

Conclusions

- Distributedness in algorithms does not inherently lead to robustness.
- To make algorithms robust, in general, we need to ensure the agents receive enough information from their neighbors to be able to detect and isolate faulty agents.