Consensus in Multi-Agent System

Yi Li

CS6366 Distributed Artificial Intelligence

1 Introduction

Due to broad applications of multi-agent systems in many fields including cooperative control of UAVs, flocking, distributed sensor networks and congestion control in communication networks, distributed coordination of network of dynamic agents have been studied extensively in the literature. In many application related to multi-agent systems, agents groups need to reach consensus, that is to converge to a same value which might or might not be related to the motion of the individual agents. This kind of problems called consensus problems. Although the consensus problem has a long history in the field of computer science, in particular automaton theory and distributed computation, it becomes an important issue that need to be addressed in distributed artificial intelligence.

The project is about studying consensus in multi-agent systems. The core idea is that the consensus problem can be established under the stochastic game framework, where the dynamic agents are players. The consensus can be formally represented by a protocol design problem including the objective functions such that if the rational agents use their best-response strategies, then such objective functions lead the whole system to a consensus value. Moreover, we assume that neighborhood relations are defined in the players set and each player has a optimal, distributed, and stationary control policy making decisions based on its neighbors states.

2 System Description

In this section, firstly, we introduce the consensus algorithm including the generic stochastic game framework and the optimal consensus protocol. Then, a case study is presented.

2.1 Consensus algorithm

$\mathcal{M} = (\Gamma, S, \hat{s}_0, A, P, g)$ is a stochastic game. where A_i is the action set of player i and $A = \times_i A_i$; P is a transition probability from $S \times A$ to S; A payoff function $g : S \times A \rightarrow R^I$ whose i-th coordinate, $g^{(i)}$, is a function of the state $s^{(i)} \in S_i$ and the action profile $a^{(i)} \in A_i$.

Now the consensus problem in a network can be defined based on such stochastic game framework. $G = (V, E, \mathcal{A})$ is a weighted directed graph, where $\mathcal{A}$ is a weighted adjacency matrix. Let $x_i \in R$ denotes the value of node i. $N = (V, E, \mathcal{A}, x)$ is a network with value $x \in R^n$ and information flow $G = (V, E, \mathcal{A})$. We define that two nodes i and j in a network agree if and only if $x_i = x_j$ such that all the nodes in a network reach a consensus if and only if $\forall i, \forall j, x_i = x_j, i \neq j$.

To solve the consensus problem under the stochastic game framework, we introduce a generic cost function for i-th agent: $C_i(x_i, N(x_i), a_i) = \lim\limits_{T\rightarrow\infty} \sum\limits_{i=0}^{T} F(x_i, N(x_i), a_i)$. We say that a protocol is optimal, a_i minimizes C_i. We say that a protocol is a consensus protocol if it is solution of the consensus problem. Furthermore, a consensus protocol is said to be optimal, if the controls a_i minimize C_i. At this point, we can convert the consensus problem into the corresponding mechanism design problem, which is designing a penalty function F such that there exists an optimal consensus protocol a with respect to the consensus value for any initial state.

Consider that agents with discrete time model under stochastic game framework

$$x_i(t + 1) = x_i(t) + a_i(t)$$

where ϵ is the step size. Thus, we get

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

(1) where P is a non-negative and stochastic matrix related to $\epsilon \in (0, 1/\Delta(G))$ called Perron matrix. (*1*) posted a linear consensus protocol for such discrete time model:

$$a_i = \sum_{j \in N_i} a_{ij}(x_j - x_i)$$

which is proved to be a consensus protocol when the underlying graph of the problem network is strongly connected. However, it may or may not be a optimal consensus protocol depending on the problem domain.

Now, let's consider the hidden consensus value (consensus hypotheses, or more generic framework called belief consensus) problem on Bayesian Networks. We demonstrate that such problem can be posed as an average-consensus problem presented in (*2*). Posterior in a Baysian network can be performed based on the following equation:

$$P(h_a|Z) = \alpha \prod P(z(i)|h_a)$$

where $z(i)$ is a set of observation from agents and h_a are the hidden consensus values (hypotheses). Let $\pi_i = P(z(i)|h_a)$ denotes the belief of agent i. Actually, we hope to use distributed algorithm to compute above product $\prod_i \pi_i = exp(n \cdot Ave(log(\pi))) = exp(\sum_i \log(\pi))$. Thus, we can reuse the previous linear consensus protocol to compute the hidden consensus by defining the likelihood of agent i as $l_i = log(\pi_i)$, which is

$$x_i(t+1) = x_i(t) + \epsilon \sum_{j \in N_i} (x_j - x_i), x_i(0) = log(\pi_i)$$

2.2 Case study: target tracking with distributed sensors

In this application, there are multiple static sensor nodes with known positions and a target in a bounded space. The target is initially unobservable for all the sensors. A sensor can periodically make observations and get the position of target with noise. The noise level is in direct proportion to the distance between the sensor and the target, which means that the more a sensor is close to the target, the more accurate results it can get. The goal of the sensor system is to track the target.

2.2.1 Model

There are N static sensors with known two-dimensional positions, p_n ($n = 1, 2, ..., N$) and one target with an unknown position p_t at a certain time point. For simplicity, we assume the observation noise of each sensor follows zero mean two-dimensional Gaussian distribution ($\sigma_1 = \sigma_2 = \sigma', \rho = 0, (\mu_1, \mu_2) = p_t$), and $\sigma' = \sigma + rd(p_n, p_t)$, where r is the discount factor and $d(p_n, p_t)$ is the distance between sensors and the target. We further assume that the sensor network is static and connected.

To apply the consensus algorithm, we assign the sensor set to the player set, use the sensor network as the information flow, and treat the believed target position of each sensor as the node value. In this scenario, we consider both the centralized approach and the distributed approach. The centralized approach collects all the observations from sensors and generates a fusion center, while the distributed approach makes all the sensors have a common view of the position of the target by using the consensus algorithm.

3 Experiment

3.1 Design and implementation

To reveal the relation between the uncertainty level and the accuracy of sensors for tracking the target, we choose following independent variables for the experiments:

- the number of sensors

- the connectivity of information flow
- the value of σ (which represent the uncertainty level of observations)

Due to the lack of computing resource, the experiments only test single step of the stochastic game stage so that we further assume that the target is stationary during the simulation process. (Mobile target tracking has been implemented and will be demonstrated on the presentation. However, we fail to get the complete results.)

The experiments are divided into two groups, which are centralized approaches with single iteration and distributed approaches with multiple iterations. For each configuration of information flow, five test scenarios are semi-randomly generated (random positions of sensors and semi-random edge set which guarantee the generated graph is connected). For the outcomes, we measure the rate of convergence and the deviation of consensus value. The deviations are represented by root mean square error (RMSE) with N number of sensors, $R = 300$ Monte Carlo runs and $S = 15$ iterations. Thus, for each iteration, the error function

$$e_s = \sqrt{\frac{\sum^N \sum^R e_{s,n,r}^2}{NR}}$$

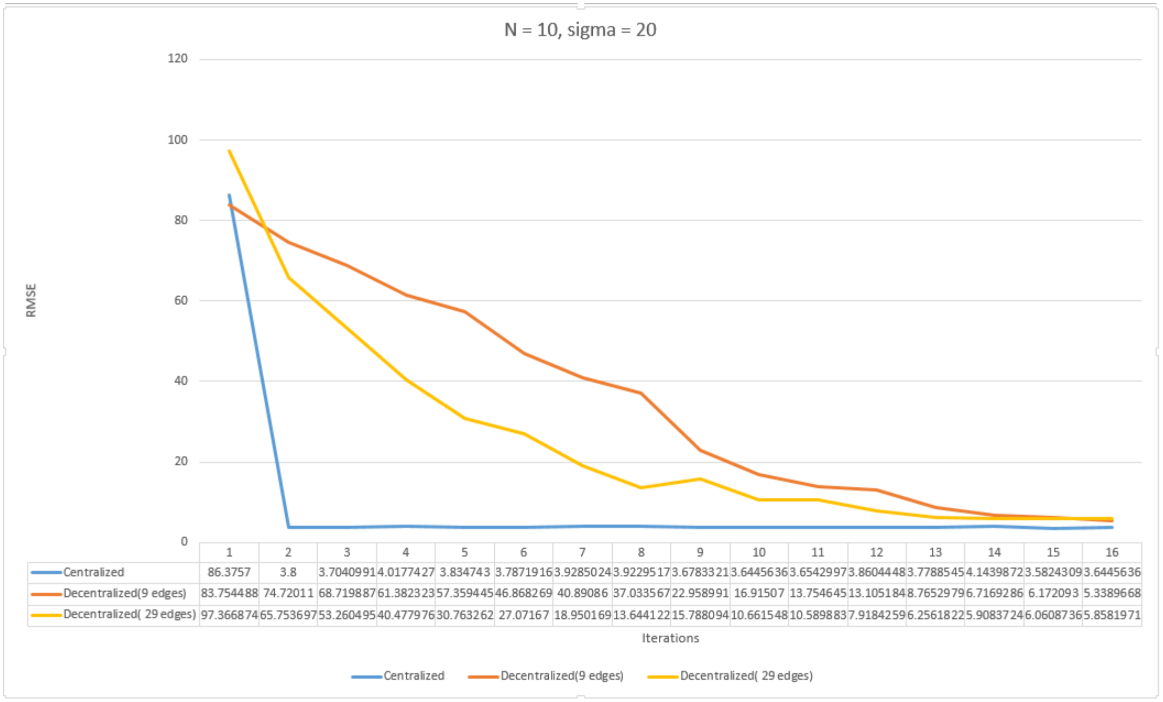
and the total error

$$e = \sqrt{\frac{\sum^S e_s^2}{S}}$$

where e is L2 distance between estimated target and the true target.

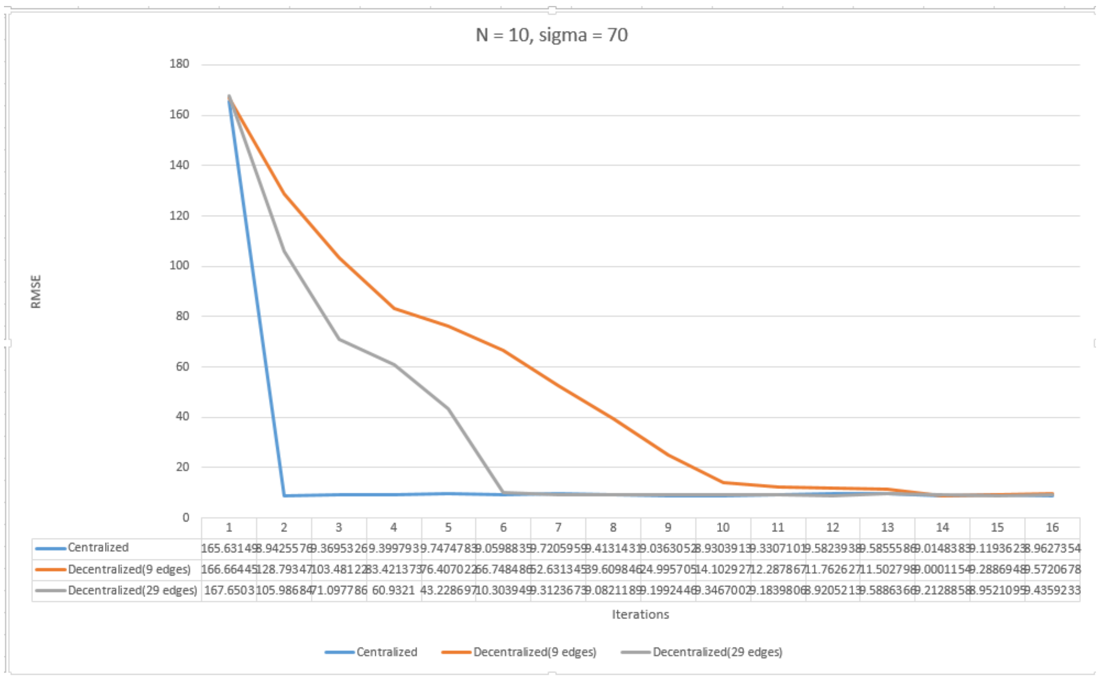
The simulation program is implemented by python and Scipy.stats. We consider a 1000*1000 bound discrete plane and use Monte Carlo approach to sample the PDF of the random variables.

3.2 Result and discussion



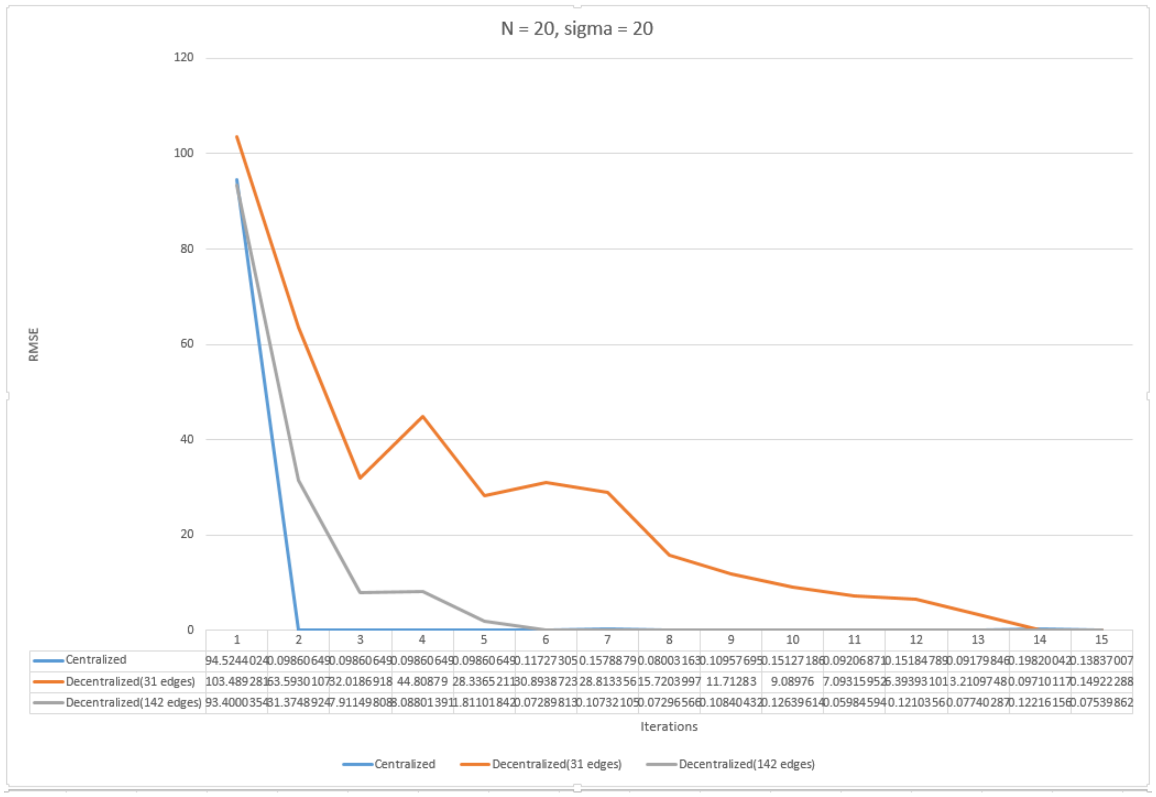
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Centralized	86.3757	3.8	3.7040991	4.0177427	3.834743	3.7871916	3.9285024	3.9229517	3.6783321	3.6445636	3.6542997	3.8604448	3.7788545	4.1439872	3.5824309	3.6445636
Decentralized(9 edges)	83.754488	74.72011	68.719887	61.382323	57.359445	46.868269	40.89086	37.033567	22.958991	16.91507	13.754645	13.105184	8.7652979	6.7169286	6.172093	5.3389668
Decentralized(29 edges)	97.366874	65.753697	53.260495	40.477976	30.763262	27.07167	18.950169	13.644122	15.788094	10.661548	10.589883	7.9184259	6.2561822	5.9083724	6.0608736	5.8581971

Figure 1: 10 sensors, $\sigma = 20$



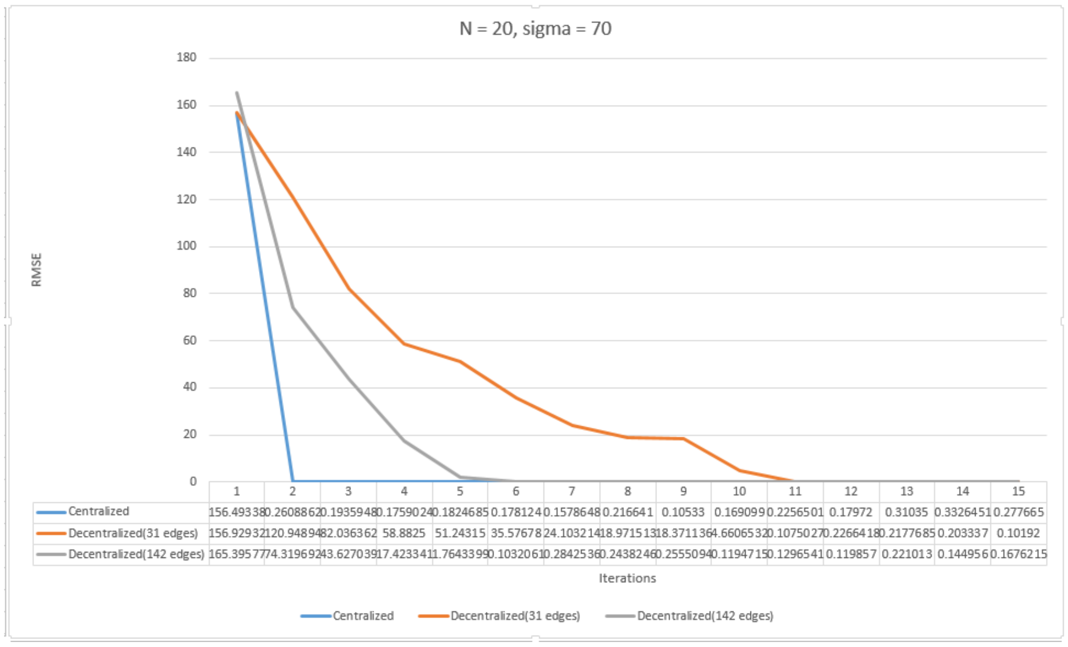
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Centralized	165.63149	8.9425576	9.3695326	9.3997939	9.7474783	9.0598835	9.7205959	9.4131431	9.0363052	8.9303913	9.3307101	9.5823938	9.5855586	9.0148383	9.1193623	8.9627354
Decentralized(9 edges)	166.66445	128.79347	103.48122	83.421373	76.407022	66.748486	52.631345	39.609846	24.995705	14.102927	12.287867	11.762627	11.502798	9.0001154	9.2886948	9.5720678
Decentralized(29 edges)	167.6503	105.98684	71.097786	60.9321	43.228697	10.303949	9.3123673	9.0821189	9.1992446	9.3467002	9.1839806	8.9205213	9.5886366	9.2128858	8.9521095	9.4359233

Figure 2: 10 sensors, $\sigma = 70$



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Centralized	94.5244024	0.09860649	0.09860649	0.09860649	0.09860649	0.11727305	0.1578879	0.08003163	0.10957695	0.15127186	0.09206871	0.15184789	0.09179846	0.19820042	0.13837007
Decentralized(31 edges)	103.489281	63.5930107	32.0186918	44.80879	28.3365211	30.8938723	28.813356	15.7203997	11.71283	9.08976	7.09315952	6.39393101	3.2109748	0.09710117	0.14922288
Decentralized(142 edges)	93.4000354	31.3748924	7.91149808	8.08801391	1.81101842	0.07289813	0.10732105	0.07296566	0.10840432	0.12639614	0.05984594	0.1210356	0.07740287	0.12216156	0.07539862

Figure 3: 20 sensors, $\sigma = 20$



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Centralized	156.49338	0.2608862	0.1935948	0.1759024	0.1824685	0.178124	0.1578648	0.216641	0.10533	0.169099	0.2256501	0.17972	0.31035	0.3326451	0.277665
Decentralized(31 edges)	156.92932	120.94894	82.036362	58.8825	51.24315	35.57678	24.103214	18.971513	18.371136	4.6606532	0.1075027	0.2266418	0.2177685	0.203337	0.10192
Decentralized(142 edges)	165.39577	74.319692	43.627039	17.423341	1.7643399	0.1032061	0.2842536	0.2438246	0.2555094	0.1194715	0.1296541	0.119857	0.221013	0.144956	0.1676215

Figure 4: 20 sensors, $\sigma = 70$

The experiment results are shown by Figure 1 4. In here, we consider 4 different scenarios with different number of sensors and value of sigma. In each scenarios, we choose the centralized approach as the base line and measure the performance of decentralized approach with different connectivity of information flows. (Since the underlying graphs are semi-randomly generated, we use the number of edges to estimate the graph connectivity. The more edges one graph has, the more dense such graph will be.)

In general, the experiment results follow our expectation:

- The results of centralized approach directly reach to the optimum.(no problem for the cases with 10 sensors. The valley of RMSE is usually around $\sqrt{\sigma}$. However, the value seems too small for the cases with 20 sensors, which may be caused by bugs.)

- The rate of convergence of decentralized approach is in direct proportion to the graph connectivity and the number of sensors. However, if we consider the cost of communications, it will become a trade off between the cost and the performance.

About the project process, in general, I just meet every milestone and finally made it. The consensus problem is a really exciting and difficult field. One problem is that I did feel time is not enough for running the experiment. The simulation for stochastic processes takes more time than I expected.

4 Future Work

- One possible reason why the experiment results seem not very impressive is we only considered the standard Gaussian distribution in the model. Some different prior assumptions may cause different results.
- Since the currently hidden consensus problem is defined under the stochastic game framework, we can further study this problem with some adversarial assumptions.(For example, if sensors can freely move around, they need to consider the risk of collision)
- Another point is that the current model can be extended by considering more complicated communication style. For example, wireless communication and the communication with time-delay.

5 Conclusion

In the project, we refer and study the consensus problem under the stochastic game framework and the hidden consensus problems. Then, we demonstrate a target tracking model, implement the simulation program, and analyze the experiment results.

References and Notes

1. R. Olfati-Saber, R. M. Murray, *IEEE Transactions on automatic control* **49**, 1520 (2004).

2. R. Olfati-Saber, E. Franco, E. Frazzoli, J. S. Shamma, *Networked Embedded Sensing and Control* (Springer, 2006), pp. 169–182.

3. B. Mohar, Y. Alavi, G. Chartrand, O. Oellermann, *Graph theory, combinatorics, and applications* **2**, 12 (1991).

4. A. Jadbabaie, J. Lin, A. S. Morse, *IEEE Transactions on automatic control* **48**, 988 (2003).

5. V. Savic, H. Wymeersch, S. Zazo, *Signal Processing* **95**, 149 (2014).