DISTRIBUTED NETWORK TIME SYNCHRONIZATION: SOCIAL LEARNING VERSUS CONSENSUS

A Thesis by

Ian Ellis L. Hulede

Bachelor of Science, Kwame Nkrumah University of Science and Technology, 2014

Submitted to the Department of Electrical Engineering and Computer Science
and the faculty of the Graduate School of
Wichita State University
in partial fulfillment of
the requirements for the degree of
Master of Science

July 2020

DISTRIBUTED NETWORK TIME SYNCHRONIZATION: SOCIAL LEARNING VERSUS CONSENSUS

The following faculty members have examined the final copy of this thesis for form and content, and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

Hyuck M. Kwon, Committee Chair

Remi Chou, Committee Member

Xiaomi Hu, Committee Member

DEDICATION

I dedicate this thesis to my awesome family; to my Dad, Mr Gove Aryee Hulede, to my Mom, Mrs Dinah Hulede, my Brother-in-law, Mr. Kwaku Danso-Dapaah, Patience Danso-Dapaah and Salomey Lamiorkor Hulede, my sweet and loving sisters, Foster Yeboah Kesse, my Brother-in-law, for their relentless support and motivation throughout my life. They have been so much inspiration to me throughout my life and my studies and to my cute Niece and Nephew; Nana Ama Danso-Dapaah and Gabriel Yeboah Kesse.

Not that we are competent in ourselves to claim anything for ourselves, but our competence comes from God. "2 Corinthians 3:5"

ACKNOWLEDGEMENTS

First I will like to thank the Almighty God for how far He has brought me, for his tender mercies and loving kindness. Special thanks goes to Dr. Hyuck M. Kwon, for his guidance and input in much of this work. His great advice is what has got me far in this work and in my studies in this institution. As an advisor and my lecturer, I appreciate every bit of his advice and lectures. I would also like to thank Dr. Remi Chou and Dr. Yanwu Ding for their lectures in courses I took under them. I enjoyed evey bit of them. This work was supported under Air Force Research Laboratory contract FA9453-19-2-0003. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the United States Air Force.

ABSTRACT

The objective of this thesis is to investigate social learning-based distributed network time synchronization (SLDNTS) and compare it to a classic approach: consensus DNTS (CDNTS). To achieve this objective, the thesis introduces a method for generating a practical observation random variable (ORV) for SLDNTS and presents both the worst and best ORV. Then, this thesis shows, through simulations, that SLDNTS is more robust than CDNTS in convergence. CDNTS fails when timing measurement error is nonzero, i.e.,a Gaussian random variable with mean equal to true time and variance equal to one is applied as an observation random variable (ORV) at each node. On the other hand SLDNTS shows quick convergence with small number of iterations even if nonzero timing measurement error, i.e., a Gaussian random variable, is applied.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS

k	Node
l	Neighbouring node
t	Time
a	Weight
v	Velocity
i	Iteration index
N	Number of nodes
Th	Threshold
L	Likelihood
Pr	Probability
Θ	Set of possible node clock time events
ψ	Intermediate belief
μ	Updated belief
ξ	Observation signal
ζ	Observation signal event
$\mathcal{N}$	Gaussian Random Variable
σ	Standard deviation
ε	Convergence parameter for SL
ϵ	Convergence parameter for Consensus
c	Speed of Light
$t_{k,i}^{dil}$	Time dilation

CHAPTER I

INTRODUCTION

1.1 Background

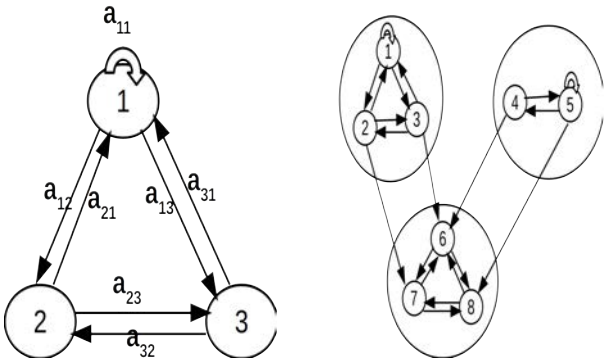
The purpose of distributed network time synchronization (DNTS) is to acquire and track a common time among the distributed and independent node clocks locally with no master-and-slave commands. Here, a node represents a clock device. The DN node clocks might initially be synchronized to a common time, but due to individual clock drifts and instabilities, they may lose synchronization as time increases. Time synchronization is necessary among distributed node clocks, especially when a Global Positioning System (GPS) signal is not available. A classic approach for the DNTS is a consensus-based DNTS (CDNTS) [2].

1.2 Research Objectives and Solutions

This paper investigates a social learning (SL) algorithm [1] for DNTS and compares its performance with CDNTS [2]. The performance criteria will be the number of iterations to reach a steady state event, and then the mean square error (MSE) after reaching the steady state event in order to measure the clock stability of the proposed SLDNTS. Only the time acquisition is studied; tracking is not investigated in this paper.

1.3 Organization of Thesis

The rest of this paper is organized as follows: Section II describes the network model. Section III presents how to generate an ORV and use the SL for the DNTS. Section IV describes how to use the consensus for the DNTS. Section V presents simulation results, and finally Section VI concludes the paper.



(a) (b)

Figure 1: Examples of networks: (a) strongly connected, and (b) weakly [1].

CHAPTER II

NETWORK MODEL

Consider a network of N nodes. Let $\mathcal{N} = \{1, 2, \ldots, N\}$ denote the set of the node indices. Also, let a_{lk} and a_{kl} denote the weight of the edge from node l to node k, and vice versa, respectively. Let $\mathcal{N}_k$ denote the set of neighbor nodes connected to node k. And let $N \times N$ matrix A represent the weights $\{a_{lk}\}$. Matrix A is a left-stochastic matrix if

$$a_{lk} \geq 0, \sum_{l \in \mathcal{N}_k} a_{lk} = 1, \text{and } a_{lk} = 0 \quad if \quad l \notin \mathcal{N}_k. \tag{1}$$

This left-stochastic, or doubly stochastic, network model will be used in this paper.

If there is a path from node k to node l for any pair (k, l), $k, l = 1, ..., N$ and at least one node has a self-loop $a_{kk} > 0$ for some k, then the network is strongly connected. Otherwise, the network is weakly connected, where information flows in only one direction over some edges in the network. Fig. 1(a) and Fig. 1(b) show examples of a strongly connected and a weakly connected network, respectively. The subnetwork consisting of nodes {6,7,8} is strongly connected if isolated and a self-loop exists, but the entire network in Fig. 1(b) is weakly connected. For example, there is no connection from node 1 to node 4 due to no feedback connection from node 6 to node 4.

CHAPTER III

SOCIAL LEARNING ALGORITHM

Social learning, e.g., tweeting, is a method used by network nodes to influence their beliefs about the true event or opinion of an issue of interest. In [1], a weakly connected network was considered, where nodes of a strongly connected subnetwork impose their beliefs on nodes in a weakly connected subnetwork, thereby causing them to follow the belief of a strongly connected subnetwork. The relation between a strongly connected subnetwork and a weakly connected subnetwork within a weakly connected network is similar to a master-and-slave relation [3] in DNTS.

The approach considered in [1] is a diffusion learning approach whereby nodes continually update their beliefs about a true event through a cooperative process. A node computes its intermediate belief $\psi_{k,i}(\theta_j)$ on event θ_j using its observed private signal, i.e., ORV $\xi_{k,i}$, by means of a Bayesian rule and then updates its belief $\mu_{k,i}(\theta_j)$ by cooperating with the neighbors [eq. (8), [1]]. In other words, node k in a SL network updates its belief on event θ_j as [1]

$$\psi_{k,i}(\theta_j) = \frac{\mu_{k,i-1}(\theta_j) L_k(\xi_{k,i}|(\theta_j)}{\sum_{\theta' \in \Theta} \mu_{k,i-1}(\theta') L_k(\xi_{k,i}|(\theta')} \tag{2}$$

$$\mu_{k,i}(\theta_j) = \sum_{l \in \mathcal{N}_k} a_{lk} \psi_{l,i}(\theta_j) \tag{3}$$

where i is the iteration index, $\psi_{k,i}(\theta_j)$ is the intermediate belief on θ_j, $\mu_{k,i}(\theta_j)$ is the updated belief, and $L_k(\xi_{k,i}|\theta_j)$ is the likelihood of an observation signal $\xi_{k,i}$ for a given conditional event θ_j.

To use SL for DNTS, let θ_j represent a possible time event at iteration i, and let Θ denote a set of such possible node clock time events. Also, let θ^o denote the true event (i.e., true time), which is unknown to the nodes. Let the true time event θ^o be equal to 0 seconds without loss of generality. The goal of the network is for all nodes to synchronize

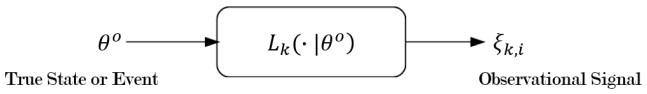


Figure 2: Generation of observational signals.

to the true event $\theta^o = 0$ seconds. Each node will keep updating its belief probability on the true event by cooperating with its neighbors. Note that each node belief probability is updated instead of each node clock time. Once the belief probability on a certain event reaches one, the node clock time can be updated as the sum of the true event time plus the updating block time interval. Clock drifting during each updating block is neglected for simplicity in this paper.

A finer time synchronization can be achieved during the next updating block period by making the time interval between the adjacent time events smaller for a fixed number of events per updating block. The set of possible clock time events in Θ will consist of finer possible time events. This paper considers only the first updating block, and Θ is time invariant for simplicity.

At the initial iteration $i = 0$, each node starts with an initial belief $\mu_{k,i=0}(\theta_j) \in [0, 1]$, which represents its probability distribution over $\theta_j \in \Theta$. At each iteration, every node makes an observation $\xi_{k,i}$ from a signal realization. This $\xi_{k,i}$ is generated from a likelihood function based on the true event θ^o.

Fig. 2 shows a block diagram for the ORV generation $\xi_{k,i}$ [1]. In this paper, each node will use a *Gaussian distribution* $\sim \mathcal{N}(\theta^o = 0, \sigma_k^2 = 1)$ for the likelihood function $L_k(\xi_{k,i}|\theta^o)$. Another variance value σ_k^2 can be used to represent each node's different clock quality. For simplicity, this paper assumes the same clock quality for all nodes. In other words, let $\xi_{k,i} \sim \mathcal{N}(\theta_k^o, 1)$ for all $k = 1, ..., N$. Then, apply a soft-decision function, e.g., a sigmoid function [4] to quantize from the ORV $\xi_{k,i}$ to a Bernoulli RV $Z_k = \{H, T\}$:

$$\zeta_{k,i}(\xi_{k,i}) = 1/(1 + e^{-\xi_{k,i}}) = e^{\xi_{k,i}}/(1 + e^{\xi_{k,i}}) \tag{4}$$

where $\zeta_{k,i}(\xi_{k,i}) = H$ if $|\xi_{k,i}| < Th$, else $\zeta_{k,i}(\xi_{k,i}) = T$. Fig. 3(a) shows the sigmoid function.

This is similar to a neuron behavior [4] . If the stimulus force of an ORV sample is smaller than $-Th$ or larger than Th, then the neuron does not respond as the tail event T, and if the stimulus force is between $-Th$ and Th, then the neuron responds as the head event H. The stimulus force random variable is always positive. However, the corresponding ORV $\xi_{k,i}$ can be negative. This paper uses the magnitude of $\xi_{k,i}$ as an input to the neuron.

Fig. 3(b) shows the probability $Pr[|\xi_{k,i}| \geq Th] = \varepsilon$, which is the probability of a no transition region (no response region) in the sigmoid function, and the probability $Pr[|\xi_{k,i}| < Th] = 1 - \varepsilon$, which is the probability of the transition in the sigmoid function. The ε is the tail probability and can be written as

$$\varepsilon = 2Q(\frac{Th - \theta_k^o}{\sigma_k}) \tag{5}$$

where $Q(\alpha) = \int_\alpha^\infty e^{-t^2/2}/\sqrt{2\pi}dt$.

In [1], there are two conditions: (a) identifiable condition (IC) and (b) prevailing observation signal existence condition (POSEC), which should be satisfied to guarantee that the true event θ^o is identifiable by all nodes, and all nodes asymptotically can learn the true event. A set of indistinguishable events Θ_k is a subset of Θ and is defined as

$$\Theta_k = \{\theta_j : L_k(\zeta_k|\theta_j) = L_k(\zeta_k|\theta^o)\} \tag{6}$$

for a quantized ORV $\zeta_k \in Z = \{H, T\}$, and the IC is written as

$$\bigcap_{k \in \mathcal{N}} \Theta_k = \{\theta^o\}. \tag{7}$$

The POSEC is written as

$$L_k(\zeta_k^o|\theta^o) - L_k(\zeta_k^o|\theta_j) \geq 0, \theta_j \in \bar{\Theta}_k \tag{8}$$

where $\bar{\Theta}_k$ is the complementary set of Θ_k. To meet these two conditions, ε is chosen to be

smaller than 0.5, and the likelihood of the head event H for node k for a given possible event $\theta_j \in \Theta$, i.e., $L(H|\theta_j)$ is $1-\varepsilon$, if $\theta_j = \theta^o$. Otherwise, $L(H|\theta_j)$ is ε, if $\theta_j \neq \theta^o$. For example, if $N = 3$ nodes, the number of events is $|\Theta| = 3$, and the true event $\theta^o = \theta_1$, then the following likelihood matrix meets the two conditions:

$$L(H) = \begin{bmatrix} 1-\varepsilon & 1-\varepsilon & 1-\varepsilon \\ \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon \end{bmatrix}. \tag{9}$$

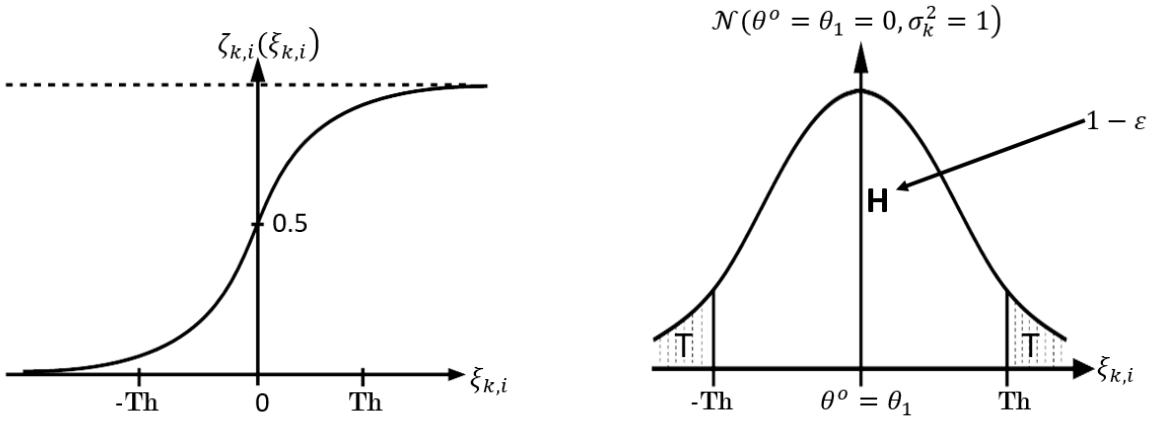
Each row index j in (9) corresponds to the possible event θ_j, $j = 1, ..., |\Theta|$, and the column index k does the node index, $k = 1, ..., N$. The column sum of $L(H)$ is not necessarily 1. It only requires $L(T) = 1 - L(H)$. For example, if $[L(H)]_{j=1,k=2} = L_k(\zeta_{k=2} = H|\theta_{j=1}) = 1-\varepsilon$, then $[L(T)]_{j=1,k=2} = \varepsilon$. Note that each component in the first row of $L(H)$ is equal to $1-\varepsilon$, which is larger than each component of the 2nd and 3rd rows. This is because $\theta^o = \theta_1$, and the prevailing event exists as $L_k(\zeta_k^o = H|\theta^o = \theta_1) - L_k(\zeta_k^o = H|\theta_j) \geq 0$ for $j = 2, 3$.

Observation 1 (Best ORV): The best quantized ORV is achieved when ε is close to 0. As ε decreases to 0, $0 < \varepsilon < 0.5$, the convergence speed increases. This is because the first row represents the true event, and the higher likelihood yields the faster convergence.

Observation 2 (Worst ORV): The worst quantized ORV is achieved when ε is close to 0.5. As ε increases to 0.5, the convergence speed slows, and the SL fails to converge when ε is close to 0.5. This is because the second and third rows represent the incorrect events, and the higher likelihood of an incorrect event will yield the slower convergence.

Observation 3: Once the IC and POSEC are met, all nodes in a strongly connected network converge to a true event θ^o asymptotically as iteration i increases.

If multiple strongly connected subnetworks are connected to a weakly connected subnetwork, then each strongly connected subnetwork converges to its own true event, but the nodes in the weakly connected subnetwork can alternate from a true event of a strongly connected network to a true event of another strongly connected network, or converge to a



(a) Sigmoid function

(b) Gaussian tail probability observation

Figure 3: ORV generation using the sigmoid function and tail probability.

dominant strongly connected true event depending on the weight matrix.

As a side observation, if all nodes in a strongly connected network have the maximum number of connections to N nodes with equal weights, then the convergence speed becomes the fastest. This is because the network cooperation is the most efficient in this case.

CHAPTER IV

CONSENSUS ALGORITHM

In a classic consensus algorithm [2], all nodes cooperate in order to agree on a certain quantity of interest, i.e., a common sync time with others, by exchanging individual node clock times. Therefore, CDNTS with zero timing measure error is updated as

$$t_k(i+1) = t_k(i) + \epsilon_{cons} \sum_{l \in \mathcal{N}_k} a_{lk}(t_l(i) - t_k(i)) \tag{10}$$

where $t_k(i+1)$ is an updated time for node k at iteration $i+1$, a_{lk} is the weight from node l to node k, $t_l(i)$ and $t_k(i)$ are the exact clock times of node l and k at iteration i, respectively, ϵ_{cons} is a convergence parameter between 0 and $1/\triangle$, and $\triangle$ is the maximum degree of the network connection. A higher ϵ_{cons} yields faster convergence but a larger MSE. Hence this paper uses $\epsilon_{cons} = 1/\triangle$.

The same simulation environment used in SLDNTS is used for CDNTS: each node's ORV $\sim \mathcal{N}(\theta^o = 0, \sigma_k^2 = 1)$. A practical CDNTS with a nonzero timing measure error is updated as

$$t_k(i+1) = t_k(i) + \epsilon_{cons} \sum_{l \in \mathcal{N}_k} a_{lk}(obs_l(i) - obs_k(i)) \tag{11}$$

where $obs_l(i) \sim \mathcal{N}(t_{true} = \theta^o = 0, \sigma_{obs}^2 = 1)$.

CHAPTER V

SIMULATION RESULTS

In this section, simulation results are presented to compare SLDNTS and CDNTS.

First, a strongly connected network of three nodes, $\mathcal{N} = \{k = 1, 2, 3\}$ is considered for SLDNTS. And a set of three possible events (i.e., times),
$\Theta = \{\theta_j\} = \{\theta_1 = 0, \theta_2 = 3, \theta_3 = -3\}$, is assumed for all nodes k. The true event is $\theta^o = \theta_1 = 0$ seconds. The proposed SLDNTS algorithm is summarized as follows:

STEP 1: Call a doubly-stochastic or left-stochastic weight matrix A in (1) with random connection weights.

STEP 2: Assume a uniform initial belief probability for all events at all nodes: $\mu_{k,i=0}(\theta_j) = \frac{1}{3}$ for all $j = 1, 2, 3$ and for all nodes $k = 1, 2, 3$.

STEP 3: Call an ORV $\xi_{k,i}$ at each node k and iteration i using the Gaussian distribution: $\xi_{k,i} \sim \mathcal{N}(\theta^o = \theta_1 = 0, \sigma_k^2 = 1)$.

STEP 4: Quantize the ORV $\xi_{k,i}$ into a Bernoulli RV $\zeta_{k,i} \in \{H, T\}$ using threshold Th in the sigmoid function (4) at each node.

STEP 5: Compute the tail probability ε in (5) using the threshold Th.

STEP 6: Build the likelihood ORV matrix in (9) using ε in (5).

STEP 7: Using the likelihood matrix in (9), compute the intermediate probability in (2), and update the belief probability in (3) on event θ_j at iteration i and node k, $j = 1, 2, 3$ and $k = 1, 2, 3$.

STEP 8: Stop, if the belief probability converges to 1. Else, increase $i = i + 1$, and go to Step 3.

Since the network is strongly connected, the belief probability on the true event converges to one and the belief probabilities on false events converge to zero, as iteration i

increases at all nodes k.

$$\lim_{i\to\infty} \mu_{k,i}(\theta^o = \theta_1) = 1, \lim_{i\to\infty} \mu_{k,i}(\theta_2) = \lim_{i\to\infty} \mu_{k,i}(\theta_3) = 0. \tag{12}$$

For CDNTS simulation, use the same initial time events as those used for SLDNTS. Here, $t_k(i)$ denotes the time of node k at iteration i. Let $t_1(i=0) = -3$, $t_2(i=0) = 0$, and $t_3(i=0) = 3$ seconds. Assume that the true time event is $\theta^o = t_2 = 0$ seconds. Then, call at each node a random Gaussian RV $\sim \mathcal{N}(\theta^o = 0, \sigma_k^2 = 1)$, and use this ORV $obs_k(i)$ in (11). A doubly stochastic connection weight matrix A in (1) is used.

Figs. 4(a) and (b) show the convergence of SLDNTS to the true event $\theta^o = 0$ with the tail probability as a parameter equal to $\varepsilon = 0.001$, and 0.45, respectively, for a strongly connected network, doubly stochastic connection matrix, random connection weights, and $N = 3$ nodes in Fig. 1(a). Fig. 4(c) shows the results for a strongly connected network, doubly stochastic connection matrix, random connection weights, $N = 30$ nodes, and $\varepsilon = 0.001$.

Observe that all nodes in a strongly connected network converge to the true time event $\theta^o = 0$ seconds within only two iterations for both cases involving a small number of nodes $N = 3$ and a high number of nodes $N = 30$. This can be achieved by controlling the tail probability to a small number such as $\varepsilon = 0.001$. Also, observe that the convergence speed becomes slower as ε approaches 0.5. Furthermore, observe that the MSE is zero once it reaches the steady state belief probabilities.

Figs. 5(a), and (b) show the convergence of SLDNTS to the true event $\theta^o = 0$ with the tail probability as a parameter: $\varepsilon = 0.001$ and 0.45, respectively, for a weakly connected network, random connection weights, and $N = 8$ nodes in Fig. 1(b). It was assumed that both strongly connected subnetworks {1,2,3} and {4,5} have a common true event of $\theta^o = 0$ seconds. The weakly connected subnetwork {6,7,8} can have a different true event from that of the strongly connected subnetworks, e.g., $\theta^o = -3$ seconds. Observe that all nodes in the weakly connected subnetwork do not converge to their own true event but converge to that of the strongly connected subnetworks. Observe also that ε can

control the convergence speed. As ε gets smaller, the convergence speed increases.

Figs. 6(a) and (b) show CDNTS convergence for $N = 3$ and $N = 30$, respectively, using the same network conditions as in SLDNTS shown in Fig. 3 when there is zero timing measurement error. Figs. 6(c) and (d) show the corresponding CDNTS time update for $N = 3$ and $N = 30$, respectively for zero timing measurement error. Observe that the CDNTS does not converge if timing measurement error is not zero. And the CDNTS performs worse in both convergence speed and MSE than SLDNTS for $N = 3$ when timing measurement for the CDNTS is zero. Note also that a nonzero timing measurement error ORV either $\sim \mathcal{N}(\theta^o, \sigma_k^2)$ or $\sim \mathcal{N}(\theta^o + t_{k,i}^{dil}, \sigma_k^2)$ is always applied for the SLDNTS.

Time Dilation: When the speed of a node is high relative to a reference node, the time of the node observed by the reference node gets dilated. The true time θ^o will be shifted by this time dilation amount. Therefore, to include this time dilation effect the ORV $\xi_{k,i}$ is generated as $\xi_{k,i} \sim \mathcal{N}(\theta^o + t_{k,i}^{dil}, \sigma_k^2)$. The time dilation shift amount $t_{k,i}^{dil}$ is written as

$$t_{k,i}^{dil} = \frac{\theta^o}{\sqrt{1 - \frac{v^2}{c^2}}} \tag{13}$$

where θ^o is the true time when the node is stationary, v is the velocity of the moving node and c is the speed of light.

For a speed of a geostationary earth orbit satellite equal to $v = 3,075.4m/s$, the time dilation did not change the SLDNTS belief probability. When $v \leq 0.94c$, the time dilation does not change the SLDNTS probability of belief on true time event θ^o. However, as shown in Fig. 7, the SLDNTS fails to converge when $v \geq 0.96c$. Note that when $v = 0.94c$, the time dilation shifting amount is 2.93 seconds which is within the gap of 3 seconds between the true event and a false event. Hence, the SLDNTS is still working for $v = 0.94c$ correctly. However, when $v = 0.96c$, the time dilation shifting amount is 12.755 seconds which is outside the gap of 3 seconds between the true event and a false event. Hence, the SLDNTS fails for $v = 0.96c$.

To demonstrate the effects of time dilation, the time difference between $\theta^o = \theta_1 = 1$

and $\theta_2 = -2$ or $\theta^o = \theta_1 = 1$ and $\theta_3 = 4$ was chosen to be 3. This time gap between one hypothesis and another neighbor hypothesis is related to time resolution of the time synchronization. The difference should be scaled depending on the desirable time resolution. And for simulation, the true event $\theta^o = 1$ instead of $\theta^o = 0$ was used. And the false events $\theta_2 = -2$ and $\theta_3 = 4$ instead of $\theta_2 = -3$ and $\theta_3 = 3$ were used. This is because the time dilation shifting amount will be zero if $\theta^o = 0$ from (13).

The time dilation effect on the CDNTS was also observed. For zero timing measurement error, the initial conditions in (10) is changed to include the time dilation effect as

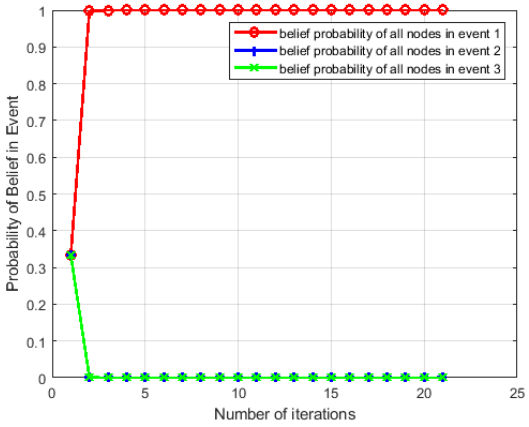
$$t_k(1) = t_k(1) + t_{k,i}^{dil}. \tag{14}$$

For nonzero timing measurement error, the CDNTS in (11) assumes that its true time is also shifted by the time dilation amount in (13) as

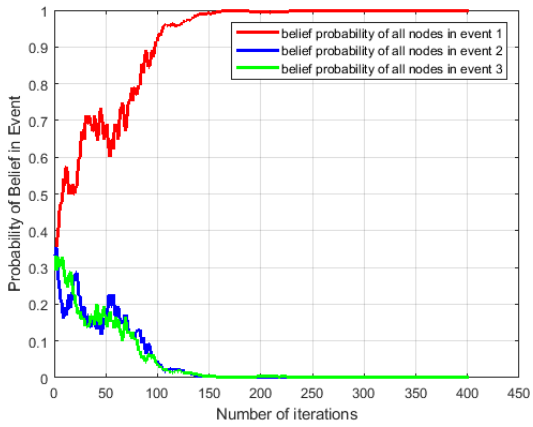
$$t_{true} = \theta^o + t_{k,i}^{dil} \tag{15}$$

and generates an observation time $obs_k(i)$ in (11) at each node $k = 1, \cdots, N$ using the ORV as $obs_k(i) \sim \mathcal{N}(t_{true}, \sigma_{obs}^2 = 1)$. For CDNTS simulation, the true event $\theta^o = 1$ instead of $\theta^o = 0$ was also used and the false events $\theta_2 = -2$ and $\theta_3 = 4$ were used as the SLDNTS.

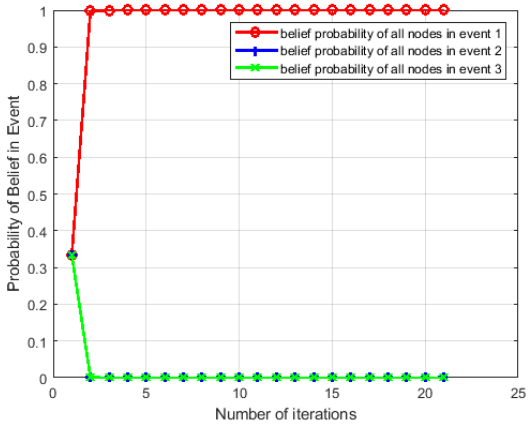
Figs. 8(a) and (b) show the CDNTS updated time versus iteration index for zero- and nonzero-timing measurement error cases, respectively. Observe in Fig. 8(a) that the convergence time is shifted from the true time $t_{true} = \theta^o = 1$ to $\theta^o + t_{k,i}^{dil} = 3.93$ when $v = 0.94c$ even if there is zero timing measurement error. Observe in Fig. 8(b) that the true event $\theta^o = 1$ cannot be achieved for any velocity v even for a strongly connected network in Fig. 1(a) if the timing measurement error is nonzero. The CDNTS in Fig. 8(b) performs worse than the CDNTS in Fig. 6(c) when the timing measurement error is nonzero and time dilation effects are included.



(a) SL for $N = 3$, Strongly Connected, $\varepsilon = 0.001$

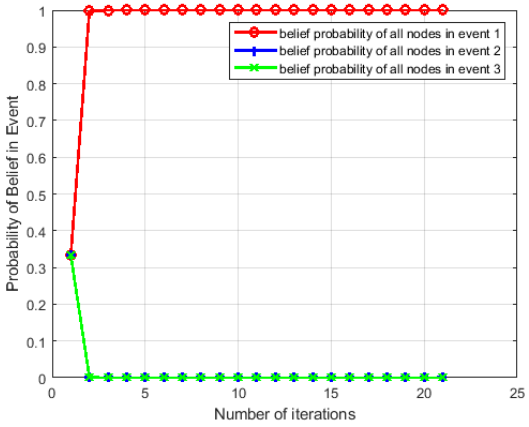


(b) SL for $N = 3$, Strongly Connected, $\varepsilon = 0.45$

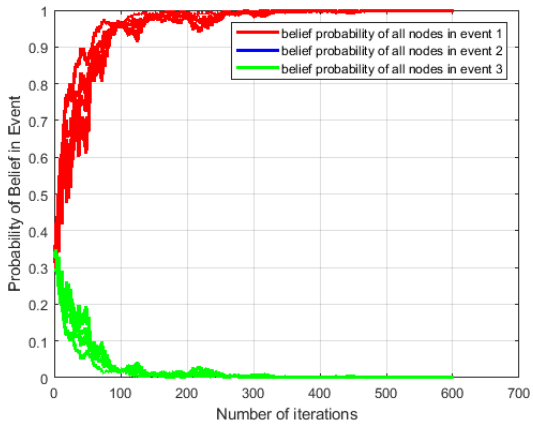


(c) SL for $N = 30$, Strongly Connected, $\varepsilon = 0.001$

Figure 4: SLDNTS convergence to true event 1: $\theta^o = 0$ seconds with different ε for strongly connected network in Fig. 1(a).

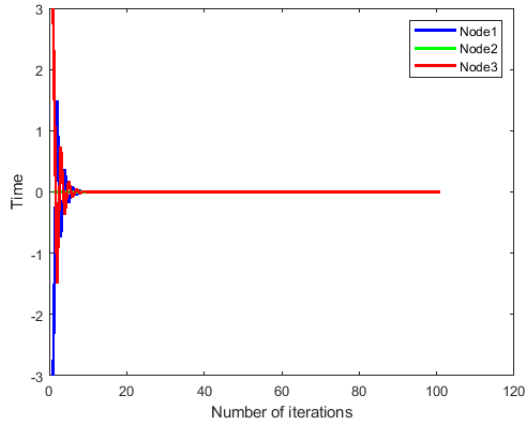


(a) SL, $N = 8$, Weakly Connected, $\varepsilon = 0.001$

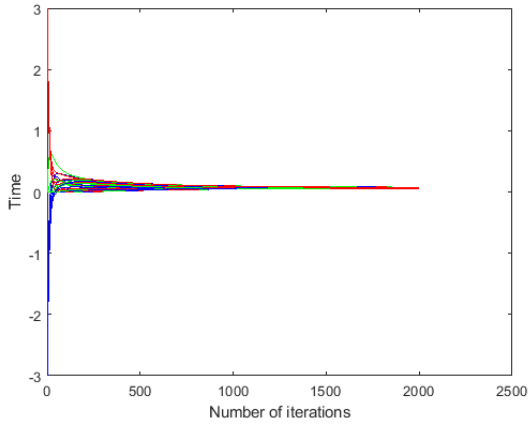


(b) SL for $N = 8$, Weakly Connected, $\varepsilon = 0.45$

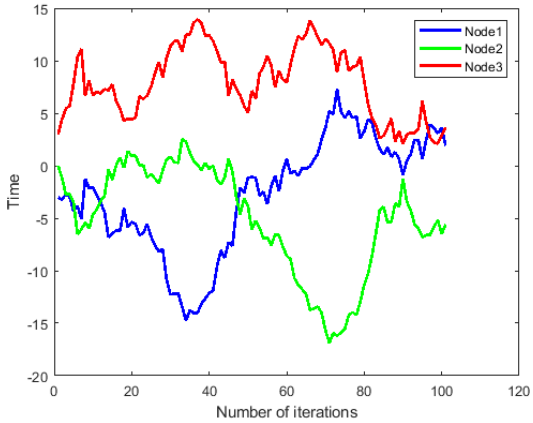
Figure 5: SLDNTS convergence to true event 1: $\theta^o = 0$ seconds with different ε for weakly connected network of $N = 8$ nodes in Fig. 1(b).



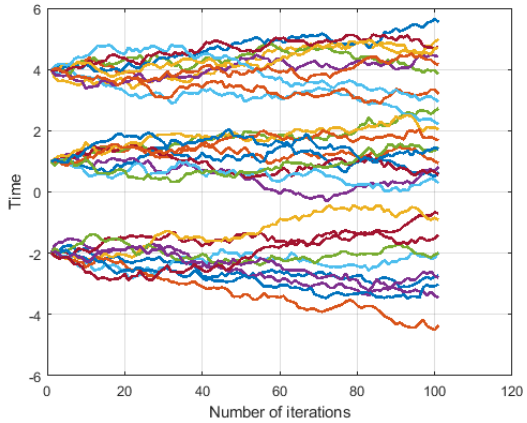
(a) Consensus with zero observation error for $N = 3$



(b) Consensus with zero observation error for $N = 30$

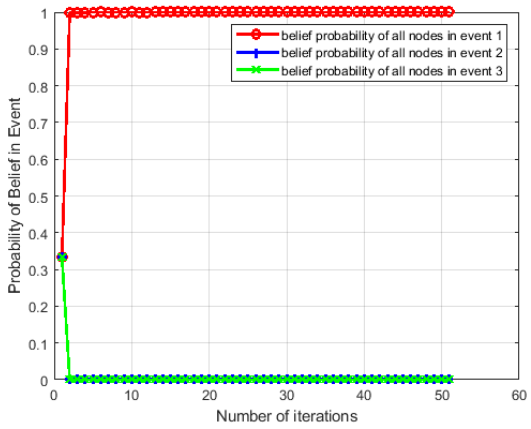


(c) Consensus with nonzero observation error in (11) for $N = 3$

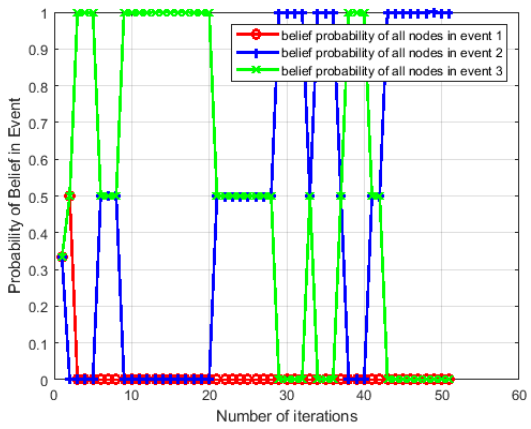


(d) Consensus with nonzero observation error for $N = 30$

Figure 6: Consensus algorithm with zero- and nonzero-timing measurement error in (10) and (11), respectively for $N = 3$ and $N = 30$ for the same network used in SL in Fig. 4.

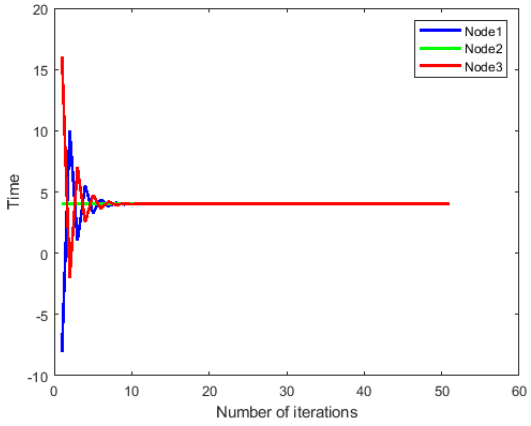


(a) $v = 0.94c$

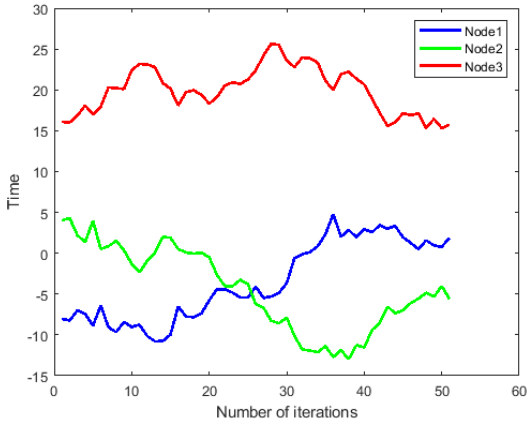


(b) $v = 0.96c$

Figure 7: Time dilation effect on SLDNTS when (a) $v = 0.94c$ and (b) $v = 0.96c$ for a strongly connected network in Fig. 1(a).



(a) $v = 0.94c$



(b) $v = 0.94c$

Figure 8: Time dilation effect on (a) CDNTS with zero timing measurement error, $v = 0.94c$ and (b) CDNTS with nonzero timing measurement error, $v = 0.94c$ for a strongly connected network in Fig. 1(a).

CHAPTER VI

CONCLUSION

SLDNTS is more robust than CDNTS in convergence. CDNTS fails when timing measurement error is nonzero, i.e., a Gaussian random variable with mean equal to true time and variance equal to one is applied as an observation random variable (ORV) at each node. On the other hand SLDNTS shows quick convergence with small number of iterations even if nonzero timing measurement error, i.e., a Gaussian random variable, is applied. A tail probability ε of an ORV can control the performance of SLDNTS. As ε decreases to zero, the convergence speed of SLDNTS increases. And as ε approaches to 0.5, the convergence speed elongates. The MSE of SLDNTS is zero once it reaches the belief probability of 1. SLDNTS shows robustness in convergence even under time dilation environment of a significant relative velocity such as $v = 0.94c$ and time event step size is 3 seconds.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] H. Salami, B. Ying, and A. H. Sayed, "Social learning over weakly connected graphs," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 3, no. 2, pp. 222–238, 2017.

[2] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[3] L. Han and N. Hua, "A distributed time synchronization solution without satellite time reference for mobile communication," *IEEE communications letters*, vol. 17, no. 7, pp. 1447–1450, 2013.

[4] V. S. Kumar, S. V. Chandra, and C. S. Kumar, "Neuro-fuzzy function approximations using feedforward networks-an application of sigmoidal signal," in *2010 Ninth International Conference on Machine Learning and Applications*, pp. 895–898, IEEE, 2010.

APPENDIX

Matlab codes

```
%SOCIAL LEARNING ALGORITHM FOR TIME SYNCHRONIZATION

clc;
clear;

T=10; %number of iterations
N=3;    %number of nodes
% A = xlsread('weight matrix1.xlsx');
c={'k','g','r'};
index=1;
m={'o','*','+'};


%hypothesis definition%
theta1=0;
theta2=-3;
theta3=3;

theta_true=theta1; %true event definition


A = bistoch(N); %bistochastic weight matrix or connection matrix
% A= f;
% A=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.1 0.2 0.1 0.2 0.1 0.2 0.2;0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
% A = [0.1 0.1 0.4 0.4; 0.1 0.1 0.4 0.4; 0.1 0.1 0.1 0.1; 0.1 0.1 0.1 0.1];
% A=[0.2 0.2 0.8 0 0 0 0 0;0.5 0.4 0.1 0 0 0.2 0 0.4;0.3 0.4 0.1 0 0 0.1 0 0; 0 0 0 0.4 0.3 0.3 0 0;0 0 0 0.6 0.7
0 0 0 0 0 0.2 0.3 0.2; 0 0 0 0 0 0.1 0.5 0.3;  0 0 0 0 0 0.1 0.2 0.1];

% A=[0.5 0.3 0.2; 0.2 0.5 0.3; 0.3 0.2 0.5]; %weight matrix or connection matrix

% A=[0.5 0.25 0.25 0; 0.25 0.5 0 0.25; 0.25 0 0.5 0.25; 0 0.25 0.25 0.5];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% social learning
%initial beliefs of all agents%
% (agent, theta)
init_belief_theta = (ones(N, 3) * 1/3);    %'i'th row is init belief theta of 'i'th agent [theta1 theta2 theta3]
```

```
%preallocation
% (time, theta, agent)
int_belief_theta = zeros(T, 3, N); %intermediate belief

% (time, theta, agent)
belief_theta = zeros(T+1, 3, N);  %updated belief

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

e=0.01; %epsilon value%
LH=1-e;
var=1;  %observation variance%

Th= (qfuncinv((1-(LH))/2)*var)+ theta_true ; %Threshold calculation%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% set time_offsets
time_offset = rand(1, N) * 0;
% l_weak=[1/3 1/4 1/5 1/6 1/7 1/8 1/9];
% (rows=theta, columns=agent)
L=zeros(3, N);
for t=1:T
    obs=normrnd(theta_true+time_offset(1,1),var); %agent 1 observation(true theta)
    if (abs(obs) <= Th )  %head is observed with this Threshold%
        L(1,1) = 1-e;
        L(2:3,1) = e;
        quantized1(t)=1;
    else
        L(1,1) = e;
        L(2:3,1) = 1-e;
        quantized1(t)=-1;
    end
    obs1(:,t)= obs;
    % Agent iteration
    for n=2:N

        obs=normrnd(theta_true+time_offset(1,n),var); %agent 'n' observation
        if (abs(obs) <= Th)  %head is observed with this Threshold%
```

```
            L(1,n) = 1-e;
            L(2:3,n) = e;
            quantized2(t,n,:)=1;
        else
            L(1,n) = e;
            L(2:3,n) = 1-e;
            quantized2(t,n,:)=-1;
        end
        obs2(t,n,:)=obs;
    end

    % Agent iteration
%     for n=25:N
%         l_weak=[1/3 1/4 1/5 1/6 1/7 1/8 1/9];
%         i=1;
%         obs=normrnd(theta_true+time_offset(1,n),var); %agent 'n' observation
%         if (abs(obs) <= Th)  %head is observed with this Threshold%
%             L(1,n)   = l_weak(i);
%             L(2:3,n) = l_weak(i);
%         else
%             L(1,n)   = 1- l_weak(i);
%             L(2:3,n) = 1- l_weak(i);
%         end
%         i=i+1;
%     end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %social learning
    % Agent loop
    for agent=1:N
        %Theta loop
        for th=1:3
            belief_theta(1,th, agent) = init_belief_theta(agent,th);
        end
    end

    %compute intermediate belief of agents%  %test sum of number of hypothesis%
    % Agent loop
    for agent=1:N
        int_belief_theta(t+1, :, agent)=(belief_theta(t, :, agent) .* L(:, agent)') ./ (belief_theta(t, 1, agent)
    end
```

```
    %compute new belief of agents
    % Agent loop
    for agent_outer=1:N
        for agent_inner=1:N
            belief_theta(t+1, :, agent_outer) = belief_theta(t+1, :, agent_outer) + (A(agent_inner, 1) * int_beli
        end
    end

    t = t+1;
end

figure
j = 1:T+1;
for i=1:N
p1= plot(j,belief_theta(:,1, i),'color','r','marker','o','LInewidth',2,'DisplayName',strcat('belief curve of agen
 hold on
 index=index+1;
end
hold off
% xlabel('iteration')
% % ylabel('belief probability of agents over time 1')
legend;

hold on
index=1;
for i=1:N
p2= plot(j,belief_theta(:,2, i),'color','b','marker','+','LInewidth',2,'DisplayName',strcat('belief curve of agen
 hold on
 index=index+1;
end
hold off
legend;
% xlabel('iteration')
% % ylabel('belief probability of agents over time 2')

hold on
index=1;
for i=1:N
p3= plot(j,belief_theta(:,3, i),'color','g','marker','x','LInewidth',2,'DisplayName',strcat('belief curve of agen
```

```
 hold on
 index=index+1;
end
hold off
xlabel('Number of iterations')
ylabel('Probability of Belief in Event')
grid
legend([p1(1) p2(1) p3(1)],{'belief probability of all nodes in event 1','belief probability of all nodes in even

figure
p4= plot(obs1,'LInewidth',2,'DisplayName',strcat('ORV', num2str(1)));
xlabel('Number of iterations')
ylabel('Observation')
grid
legend([p4],strcat('ORV', num2str(1))');
figure



for i=2:N
p5= plot(obs2(:,i),'LInewidth',2,'DisplayName',strcat('ORV', num2str(i)));
xlabel('Number of iterations')
ylabel('Observation')
grid
legend([p5],strcat('ORV', num2str(i))');
figure
end

figure
p6= stem(quantized1,'LInewidth',2,'DisplayName',strcat('Quantized', num2str(1)));
xlabel('Number of iterations')
ylabel('Head/Tail')
grid
legend([p6],strcat('Quantized', num2str(1))');
figure

for i=2:N
p7= stem(quantized2(:,i),'LInewidth',2,'DisplayName',strcat('Quantized', num2str(i)));
xlabel('Number of iterations')
ylabel('Head/Tail')
grid
```

```
legend([p7],strcat('Quantized', num2str(i))');
figure
end



%CONSENSUS WITH NO OBSERVATION ERROR
clc;
clear;
T=100;

%initial time definition
init_timeA = -3;
init_timeB =  0;
init_timeC = 3;

true_time = init_timeB;
max_degree = 2;
epsilon = 1/max_degree;
timeoffset=0;
delta=3;



A=[0 1 1; 1 0 1; 1 1 0]; %weight matrix
% A = bistoch(N);

timeA = zeros(T+1,1);
timeB = zeros(T+1,1);
timeC = zeros(T+1,1);
for t=1:T

 timeA(1) = init_timeA
 timeB(1) = init_timeB
 timeC(1) = init_timeC

 timeA(t+1) = timeA(t) + (epsilon)*(A(2,1)*(timeB(t)-timeA(t)+A(3,1)*(timeC(t)-timeA(t))));
 timeB(t+1) = timeB(t) + (epsilon)*(A(1,2)*(timeA(t)-timeB(t)+A(3,2)*(timeC(t)-timeB(t))));
 timeC(t+1) = timeC(t) + (epsilon)*(A(1,3)*(timeA(t)-timeC(t)+A(2,3)*(timeB(t)-timeC(t))));

 t = t+1;
```

```
end

iteration=1:T+1;
 plot(iteration,timeA,'b','DisplayName','Node1','Linewidth',2)
 hold on
 plot(iteration,timeB,'g','DisplayName','Node2','Linewidth',2)
 hold on
 plot(iteration,timeC,'r','DisplayName','Node3','Linewidth',2)
 legend
 xlabel('Number of iterations')
 ylabel('Time')



%CONSENSUS WITH  OBSERVATION ERROR
clc;
clear;
T=100;

init_timeA = -3;
init_timeB =  0;
init_timeC = 3;

true_time = init_timeB;
max_degree = 2;
epsilon = 1/max_degree;

N=3;

% A=[0.5 0.3 0.2; 0.2 0.5 0.3; 0.3 0.2 0.5]; %weight matrix
% A = bistoch(N);
A=[0 1 1; 1 0 1; 1 1 0]; %weight matrix

timeA = zeros(T+1,1);
timeB = zeros(T+1,1);
timeC = zeros(T+1,1);

for t=1:T
% consensus with time observation error
 obsA=normrnd(true_time,1^2); %time A observation
 obsB=normrnd(true_time,1^2); %time B observation
```

```
 obsC=normrnd(true_time,1^2); %time C observation

 timeA(1) = init_timeA;
 timeB(1) = init_timeB;
 timeC(1) = init_timeC;


 timeA(t+1) = timeA(t) + (epsilon)*(A(2,1)*(obsB-obsA)+A(3,1)*(obsC-obsA));
 timeB(t+1) = timeB(t) + (epsilon)*(A(1,2)*(obsA-obsB)+A(3,2)*(obsC-obsB));
 timeC(t+1) = timeC(t) + (epsilon)*(A(1,3)*(obsA-obsC)+A(2,3)*(obsB-obsC));

 t = t+1;

 end
iteration=1:T+1;
 plot(iteration,timeA,'b','DisplayName','Node1','Linewidth',2)
 hold on
 plot(iteration,timeB,'g','DisplayName','Node2','Linewidth',2)
 hold on
 plot(iteration,timeC,'r','DisplayName','Node3','Linewidth',2)
 legend
 xlabel('Number of iterations')
 ylabel('Time')



%CONSENSUS WITH OBSERVATION ERROR VECTORIZED(FOR N NUMBER OF NODES)
clc;
clear all;
T=100; %number of iterations%
N = 30; %number of nodes
ci={'r','b','g'};
%every three agents have these initial conditions%
init_time = repmat([-3; 0; 3], N, 1);

% init_time = randi([1,10],1,N)
true_time = init_time(2);

c=3.0*10^8;
v=0.96*c;
```

```
% time_dilation=init_time(2)*(1+(1/(sqrt(1-((v/c)^2)))));


c=3.0*10^8;
v=0;

% time_dilation=init_time(2)*(1+(1/(sqrt(1-((v/c)^2)))));
% time_dilation=9;

max_degree = N-1;
epsilon = 1/max_degree;

A=ones(N);
A = A + diag(0 - diag(A));

% A=[0 1 1; 1 0 1; 1 1 0];
% A = bistoch(N);
% A=[0.5 0.3 0.2; 0.2 0.5 0.3; 0.3 0.2 0.5]; %weight matrix or connection matrix
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%consensus with time observation error
%preallocation

%(time, agent)
time = zeros(T+1, N);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for t=1:T
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % consensus with time observation error

    %observation done for each agent based on same parameters(scalar)%
    for agent=1:N
        obs=normrnd(true_time,1^2, agent, 1); %time 1 observation
    end

    %(time, agent)
    for agent=1:N
        time(1, agent) = init_time(agent);
    end
```

```
    %timeA(t+1) = A(1,1)*timeA(t) + (epsilon)*(A(2,1)*(obsB-obsA)+A(3,1)*(obsC-obsA));
    %(time, agent)
    for agent=1:N
        agent_neighbours = 0;
        if agent == 1
            % (before_agent, after_agent)
            agent_neighbours = [N 2];
        elseif agent == N
            agent_neighbours = [N-1 1];
        else
            agent_neighbours = [agent-1 agent+1];
        end
        temp = A(agent_neighbours(2), agent)* (obs(agent_neighbours(2)) - obs(agent)) + A(agent_neighbours(1),age
        time(t+1, agent) =  time(t+1, agent) + time(t, agent) + (epsilon)*temp;

%         time(t+1, agent) =  time(t+1, agent) + (obs(agent) + (epsilon)*temp);
    end

    t = t+1;
end
 j = 1:T+1;
index=1;
figure
for agent=1:N
%  plot(j,time(:, agent),'color',ci{index},'linewidth',2,'DisplayName',strcat('Node', num2str(agent)))
p= plot(j,time(:, agent),'linewidth',2)
 hold on
 index=index+1;
end
hold off
xlabel('Number of iterations')
ylabel('Time')
% title('Consensus Algorithm for all Nodes(1-30)')
legend off
grid
set(gca,'FontSize',10)
```