2014 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing

Density Classification in Asynchronous Random Networks with Faulty Nodes.

Alexander Gogolev*†, Lucio Marcenaro†
*Institute of Networked and Embedded Systems
University of Klagenfurt
Klagenfurt, Austria
first.last@aau.at
†DITEN
Department of Naval, Electric,
Electronic and Telecommunication Engineering
University of Genoa
Genoa, Italy
first.last@unige.it

***Abstract*—This paper investigates distributed consensus for density classification in asynchronous random networks with faulty nodes. We compare four different models of faulty behavior under randomized topology. Using computer simulations, we show that (a) faulty nodes' impact depends on their location and (b) faulty nodes with persistent failures inhibit consensus stronger than commonly-used Byzantine faulty nodes with random failures. We also show that (c) randomization by Byzantine faulty nodes can be strongly beneficial for binary consensus and (d) topology randomization can increase robustness towards faulty node behavior.**

***Keywords*-Byzantine failure; self-organization; distributed consensus; density classification; randomized consensus; majority sorting; binary consensus;**

I. Introduction

Distributed consensus algorithms can be used in systems, where centralized decision making is difficult or impossible. Such conditions often reported in distributed mission planning [1], target tracking [2] or database management [3].

In a distributed consensus system, every node follows simple decision rules, utilizing local node-to-node communication to perform global coordination. Fault-tolerance is an essential property for such tasks. Studies on fault-tolerance of distributed consensus consider influence of noise [4], [5], system-wide synchrony [6], topological changes [7] and faulty nodes [8], [9]. Faulty node behavior is one of the main impediments to consensus [8]. A faulty node is generally defined as a Byzantine node, a node that can have arbitrary failures. Such node participates in consensus, but floods the network with faulty information. In the asynchronous network of N nodes M of them being faulty, already $M = 1$ can prevent consensus [9]. System-wide synchrony is another substantial factor: increasing synchrony promotes consensus [10], [11], and can increase robustness towards faulty nodes: synchronous systems can tolerate $M \geq \frac{N-1}{3}$ [8].

Moreira *et.al.* [12] show that a simple consensus rule, namely Simple Majority (SM) randomized by errors and topology, can outperform one of the best rules — Gacs-Kurdyumov-Levin (GKL) consensus [13].

In this paper we investigate different models of faulty node behavior in asynchronous networks with randomized topologies. Using GKL and SM we show that: (a) faulty nodes with persistent failure have stronger impact on consensus that commonly used Byzantine faulty nodes, (b) impact of faulty nodes depends on their position on the network, (c) randomization by topology can promote robustness towards faulty nodes, and (d) Byzantine faulty nodes can provide randomization, beneficial for consensus.

II. Setup and Notation

For network modeling we use Watts-Strogatz (WS) graph [14]. WS networks are widely used to study systems interactions, spanning from technical systems [15] to natural [16] and social networks [14]. WS network is initially modeled as a ring of N nodes, where each node is connected with the next K nodes. Further, with rewiring probability P

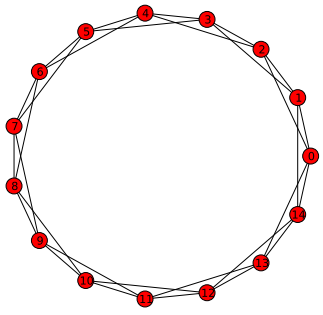


Figure 1. WS graph at $P = 0$, a network is a $2K$ — connected ring, $N = 15, K = 2$.

each link is substituted with a link to a random node on

1066-6192/14 $31.00 © 2014 IEEE
DOI 10.1109/PDP.2014.62

the network. I.e., at $P = 0$ a network is a $2K$-connected ordered lattice (Figure 1). At $P = 0.5$ a network is a Small-World where approximately half of the links are random (Figure 2). At $P = 1$ a network is a fully random graph (Figure 3). Thus, with varied P WS graph can produce networks, ranging from an ordered grid to a fully random network.

Nodes, connected to the node i form the node's i vector of neighbors N_i. Nodes $j \in N_i$, satisfying $j = (i + z) \, mod \, N$, $\forall \, z \in \{0..K\}$, sorted in ascending order form the vector of right- side neighbors of node i denoted as R_i, remaining nodes $j \in N_i$ form vector of left-sided neighbors L_i. We model networks with $\|N_i\| = 2K$, $\|L_i\| = \|R_i\| = K$.

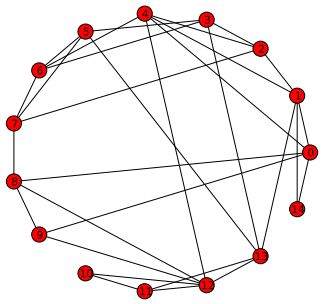


Figure 2. WS graph at $P = 0.5$, $\sim 50\%$ of the initial links are substituted with random links, $N = 15, K = 2$.

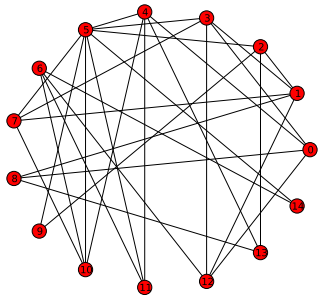


Figure 3. WS graph at $P = 1$, all links are random and network is a small-world, $N = 15, K = 2$.

III. Distributed Density Classification

Below we describe the distributed density classification, also known as majority sorting or binary consensus.

Consensus is performed in a network of N nodes. At a first time step $t = 0$ every node i is randomly assigned with a binary state $\sigma_i \in \{-1, 1\}$. A set of σ_i at $t = 0$ is called *Initial Configuration* and denoted as I. The sum of all σ_i at a time $t = 0$ is called the density of I and is denoted by ρ, $\rho \in \{-N \ldots N\}$. At every time step $t > 0$ each node updates its state following a given consensus rule, based on its own state information and state information, received from neighboring nodes. We focus on wait-free consensus: such algorithms are terminated after T time steps, whether consensus was reached or not. In a task of density classification all nodes are expected to converge to a single state, corresponding to initial majority of state distribution within T time steps. I.e., network is considered as converged if there exists time $t \in \{0, \ldots, T\}$, such that $\sum_{i=0}^{N} \sigma_i[t] = -N$ for $\rho < 0$, and $\sum_{i=0}^{N} \sigma_i[t] = -N$ for $\rho > 0$. We use $T = 2N$ originally defined in [17].

A. Simple Majority Consensus

Following Simple Majority consensus, each node on the network calculates its new value on basis of its current state and the state of its closest neighbors, i.e.:

$$\sigma_i[t+1] = G\left(\sigma_i + \sum_j \sigma_j[t]\right), \tag{1}$$

where $j \in N_i$. The update function G is [12]:

$$G(x) = \begin{cases} -1 & \text{for } x < 0 \\ +1 & \text{for } x > 0 \end{cases}. \tag{2}$$

B. Gacs-Kurdyumov-Levin Consensus

Gacs-Kurdyumov-Levin Consensus is known as one of the best human-designed rules for density classification [17], [18]. GKL is defined as follows. Each node i on the network calculates its new value depending on its own current state: if $\sigma_i < 0$ it accounts for its first and third neighbors to the left, and its own state. If $\sigma_i > 0$ node considers its first and third neighbors to the right, and its own state, i.e.:

$$\sigma_i[t+1] = \begin{cases} G\Big(\sigma_i[t] + \sigma_{l_1}[t] + \sigma_{l_3}[t]\Big) & \text{for } \sigma_i[t] < 0, \\ G\Big(\sigma_i[t] + \sigma_{r_1}[t] + \sigma_{r_3}[t]\Big) & \text{for } \sigma_i[t] > 0. \end{cases} \tag{3}$$

Here, l_1 and l_3 are the first and the third neighbor to the left and r_1 and r_3 are the first and the third neighbor to the right, respectively.

One can see that GKL is a special case of SM with a value-dependent direction bias: in GKL nodes select neighbors based on their own value.

C. Update Mode

We implement synchronous and asynchronous update functions [19]. With synchronous update all nodes in the network update to the new state simultaneously. With asynchronous update nodes are updated one after another, according to their indices, sequentially.

D. Faulty Nodes

We implement two schemes of faulty behavior: Byzantine random failure [9], [20] and non-Byzantine persistent failure. We model faulty nodes as follows. At a starting time M nodes are added to the network and labeled as "faulty". Further, faulty nodes counter consensus according to one of the following failure model.

1) Byzantine Failures: In Byzantine failure scheme, a faulty node i randomly changes its state $\sigma_i \in \{-1, 1\}$, independent from received messages, and broadcasts it among its neighbors. Such faulty nodes are not excluded from the consensus, i.e., consensus is considered as reached when all $N + M$ nodes converged to a single state.

2) Non-Byzantine Failures: In non-Byzantine, persistent failure model faulty nodes are initially assigned with a state σ_M opposite to the initial majority:

$$\sigma_M = \begin{cases} -1 & \text{for } \rho > 0 \\ +1 & \text{for } \rho < 0 \end{cases} . \tag{4}$$

In such a scheme, faulty nodes broadcast their state to the neighbors but do not update it. Non-Byzantine faulty nodes are excluded from the consensus, i.e., consensus is considered as reached if within $2N$ time steps all N non-faulty nodes have reached the same state.

For each scheme we use distributed and clustered positioning of faulty nodes in the network. With clustered layout all faulty nodes are arranged next to each other, and position of the cluster is randomly selected for each simulation run. In distributed positioning faulty nodes are randomly distributed over the network.

IV. Performance Study

As performance metric we measure convergence rate R — a fraction of a $10,000$ initial configurations that an algorithm successfully classifies. We simulate over 30 sets combined of $10,000$ I each and plot average values with 95% confidence intervals.

We use test sets of initial configurations with ρ distribution approximated by a uniform discrete distribution, $\rho \sim \mathcal{U}(-N, N)$. Simulation engine is built in *C++* using boost libraries (`www.boost.org`).

A. Update Mode Impact

Figure 4 shows convergence rate of GKL and SM in ordered networks of $N \in \{29, \ldots, 149\}$ with different update modes. It shows that test sets with uniform density distribution result in higher R than some other approaches [12], [13]. It also shows that update mode has different influence: lack of synchrony inhibits convergence rate for GKL, while for SM lack of synchrony promotes consensus. This can be explained by the fact that asynchronous updates inhibit the direction bias of GKL, while for SM random state updates create a temporal bias, similar to that of GKL. Figure 4 indicates that R is decreased with growth of N.

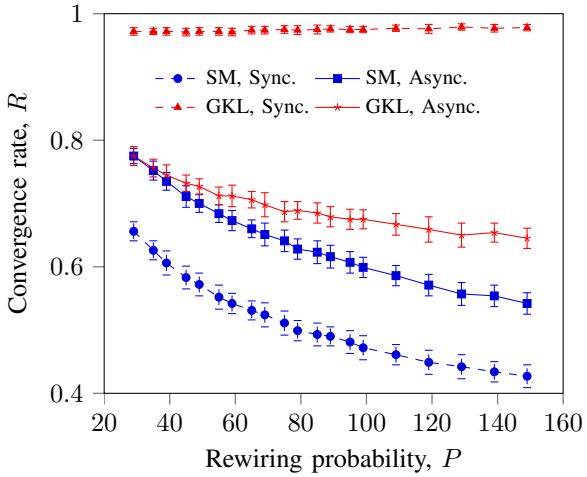


Figure 4. GKL and SM in ordered networks of $N \in \{29, \ldots, 149\}$ nodes. $\rho \sim \mathcal{U}(-N, N)$, $K = 3, M = 0$.

Such decrease can be related to the fact that systems with higher ratio $\frac{N}{K}$ are more prone to clustering [16] which can inhibit binary consensus. Further we focus on asynchronous networks which generally present a more difficult case for consensus [10].

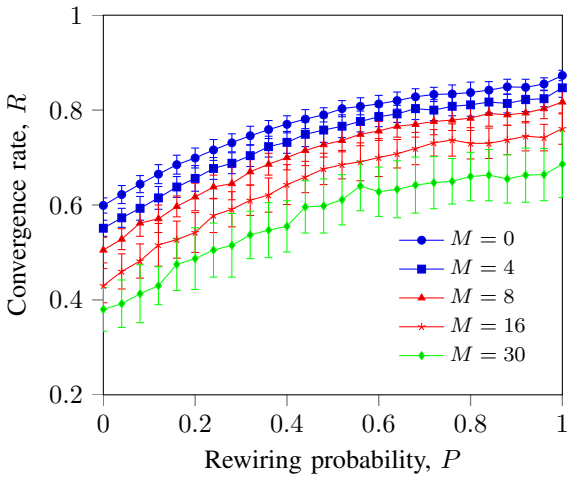
B. Byzantine Faulty Nodes

Figures 5 and 6 illustrate how Byzantine faulty nodes influence asynchronous SM and GKL under topology randomization.

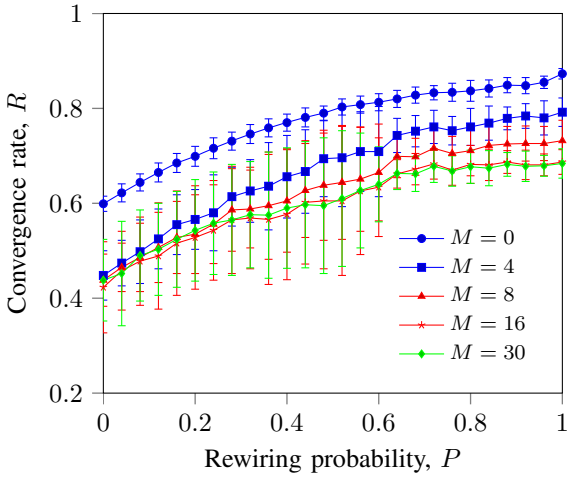
1) Simple Majority with Byzantine Faulty Nodes: Figure 5 shows that Byzantine faulty nodes always decrease the convergence rate of SM. It also shows negligible difference in impact between faulty nodes located in a single cluster (Figure 5b) and nodes, randomly distributed over the network (Figure 5a). Figure 5 indicates that topology randomization (growing P) promotes consensus — the effect that can be observed in all studied setups.

2) Gacs-Kurdyumov-Levin Consensus with Byzantine Faulty Nodes: Figure 6 shows that Byzantine faulty nodes can promote consensus. Figures 6a and 6b show that in ordered networks ($P = 0$) Byzantine faulty nodes can increase R. Topology randomization ($P \leq 0.04$) lowers this relative gain, but with further growth ($P > 0.04$) promotes consensus. Impact of faulty nodes location is weakly signified. Observations on impact of Byzantine faulty nodes can be generalized as follows:

1) Byzantine faulty nodes inhibit SM consensus.
2) Asynchronous GKL consensus in ordered grids, randomized by Byzantine faulty nodes can reach $R \simeq 100\%$.
3) Topology randomization can increase convergence rate of asynchronous GKL and SM.
4) Topology randomization can increase robustness towards faulty nodes.

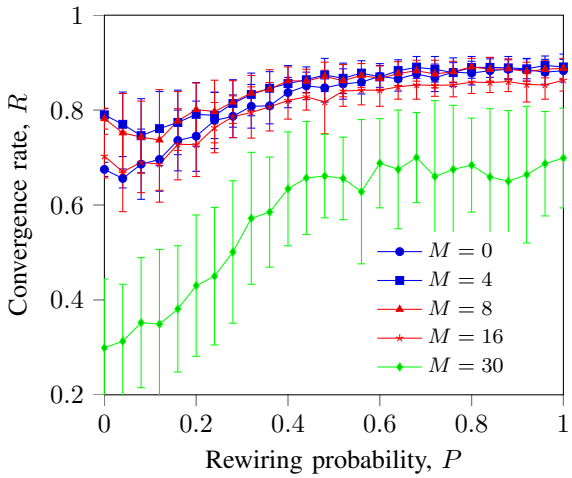


a) Faulty nodes are randomly distributed over the network.

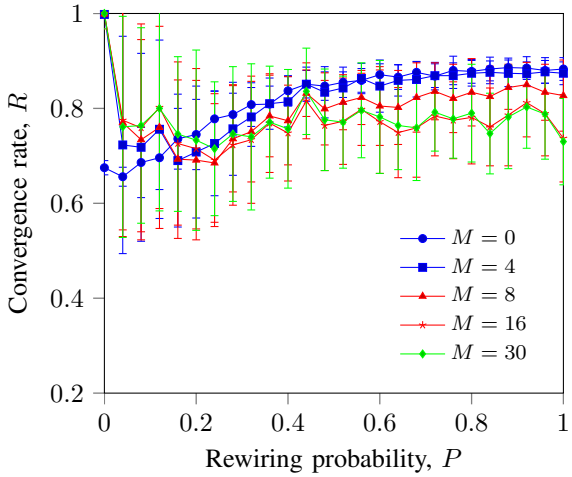


b) Faulty nodes are located in a single cluster.

Figure 5. Asynchronous SM with Byzantine faulty nodes. Topology randomization in a network of $N = 99$ nodes.



a) Faulty nodes are randomly distributed over the network.



b) Faulty nodes are located in a single cluster.

Figure 6. Asynchronous GKL with Byzantine faulty nodes with topology randomization, $N = 99$.

$R \simeq 100\%$ achieved by an asynchronous GKL in ordered grids with randomization by Byzantine faulty nodes is higher than R of any other algorithm to our knowledge, and should be studied in more detail. Such positive impact of faulty nodes can be explained by randomization they impose on information exchange. It was previously shown that similar effects can promote consensus [21], [12]. Increase of R under growing P can be explained by both topology randomization [12] and by the fact that in WS model growth of P increases the average link length [14], which contributes to higher R.

C. Non-Byzantine Faulty Nodes

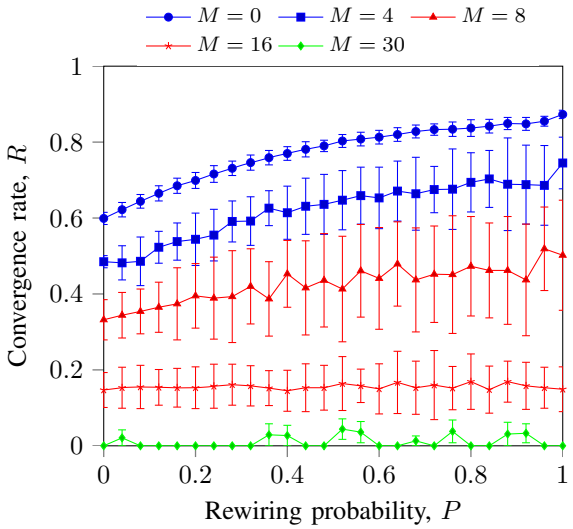
Figures 7 and 8 show convergence rate of GKL and SM with non-Byzantine faulty nodes.

1) Simple Majority with Non-Byzantine Faulty Nodes: Pairwise comparison of Figures 5 and 7, shows that non-Byzantine nodes inhibit consensus stronger than Byzantine nodes. Figure 7 shows that faulty nodes located in a single cluster, have higher impact on R than nodes randomly distributed around network.

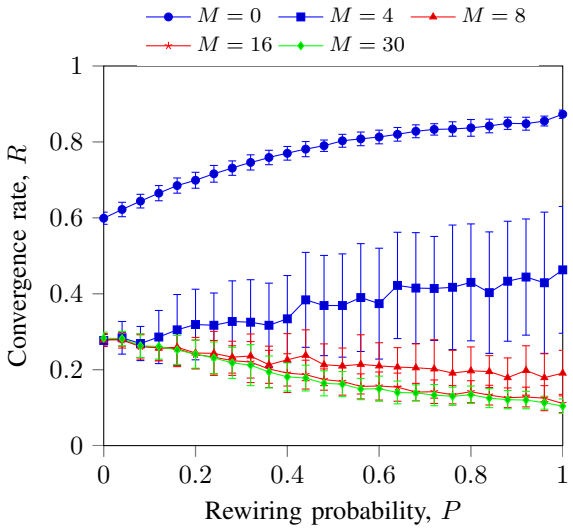
2) Gacs-Kurdyumov-Levin Consensus with Non-Byzantine Faulty Nodes: Figure 8 shows that similarly to SM, non-Byzantine faulty nodes inhibit GKL stronger than Byzantine faulty nodes (Figure 6). Nodes located in a single cluster, decrease R larger than nodes randomly placed over the network.

These observations can be summarized as follows:

1) Non-Byzantine faulty nodes inhibit consensus stronger than Byzantine nodes.
2) Faulty nodes randomly distributed over the network, have lower impact than nodes allocated in a cluster.
3) Topology randomization can increase R.

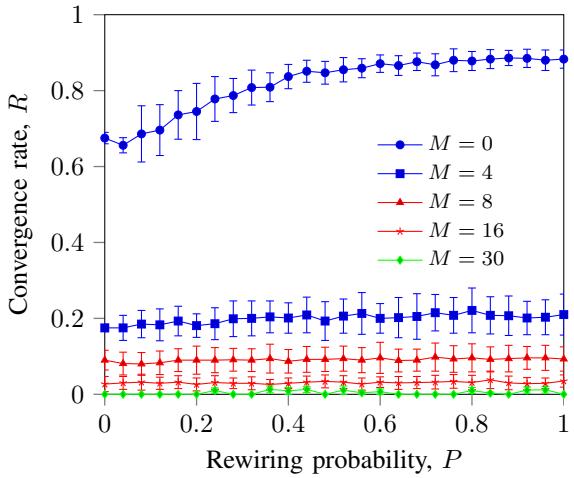


a) Faulty nodes are randomly distributed over the network.

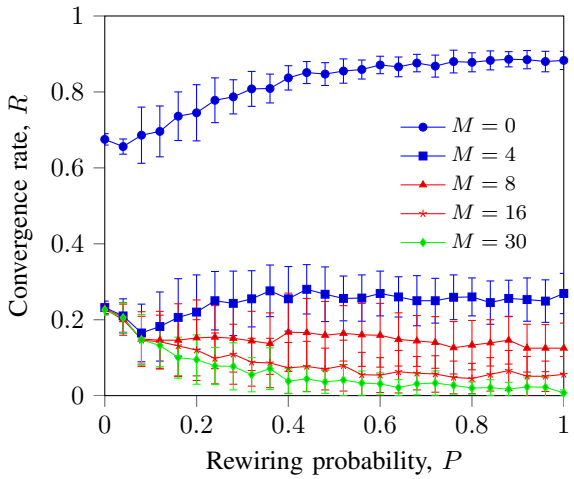


b) Faulty nodes are located in a single cluster.

Figure 7. Asynchronous SM, topology randomization with non-Byzantine faulty nodes. Network of $N = 99$ nodes.



a) Faulty nodes are randomly distributed over the network.



b) Faulty nodes are located in a single cluster.

Figure 8. Asynchronous GKL, topology randomization with non-Byzantine faulty nodes, $N = 99$.

4) Topology randomization can partially restore drop in R induced by faulty nodes for SM.

The first effect can be explained by the fact that nodes with non-Byzantine failure model persistently inhibit consensus, while Byzantine faulty nodes can sometimes provide correct input. The second effect can be explained by dithering of the negative impact of faulty nodes into larger set of inhibited nodes, and thus allowing the latter ones to overcome the disturbance. All models but one positively respond to topology randomization, indicating growth of R even with faulty nodes. The positive influence of topology randomization can be explained by dithering of topological clusters [15]. Growth of rewiring probability P in WS networks not only randomizes the topology, but also increases the average link length in the network [14], which also contributes to gains in R. The only model that does not indicate increase in R under growing P is asynchronously updated GKL with non-Byzantine faulty nodes. This can be explained by the value-dependent direction bias of GKL. This bias provides for GKL's higher R in ordered networks, but it can be inhibited by topology randomization.

V. Conclusions

This paper studies binary density classification in random asynchronous networks with faulty nodes. Presented results indicate that topology randomization can increase convergence rate of algorithms with faulty nodes. We also show that the impact of faulty nodes can be lowered by altering their position: faulty nodes, randomly distributed

over the network have lower impact on consensus than ones clustered next to each other. We show that commonly used Byzantine failure model inhibits binary consensus less than persistent failure model. Further, we show that in some cases Byzantine faulty nodes can provide beneficial randomization that strongly promotes consensus. These observations can be generalized according to model assumptions to a wide range of networks from ordered grids to fully random networks.

ACKNOWLEDGMENT

This work was performed within the Erasmus Mundus Joint Doctorate in "Interactive and Cognitive Environments", which is funded by the EACEA Agency of the EC under EMJD ICE FPA n 2010-0012. The work of A. Gogolev is supported by Lakeside Labs, Klagenfurt, with funding from the ERDF, KWF, and the state of Austria under grant 20214/21530/32606.

REFERENCES

[1] M. Alighanbari and J. P. How, "An unbiased Kalman consensus algorithm," in *Proceedings of American Control Conference*, 2006, pp. 3519–3524.

[2] A. Gee and R. Cipolla, "Fast visual tracking by temporal consensus," *Image and Vision Computing*, vol. 14, no. 2, pp. 105–114, 1996.

[3] R. H. Thomas, "A majority consensus approach to concurrency control for multiple copy databases," *ACM Transactions on Database Systems*, vol. 4, no. 2, pp. 180–209, 1979.

[4] S. Kar and J. M. F. Moura, "Distributed average consensus in sensor networks with random link failures and communication channel noise," *Proceedings of Asilomar Conference on Signals, Systems and Computers*, pp. 676–680, 2007.

[5] J. R. G. Mendonçia, "Sensitivity to noise and ergodicity of an assembly line of cellular automata that classifies density," *Physical Review Letters E*, vol. 83, no. 3, p. 31112, 2011.

[6] G. Bracha and S. Toueg, "Asynchronous consensus and broadcast protocols," *Journal of the ACM*, vol. 32, no. 4, pp. 824–840, 1985.

[7] D. B. Kingston and R. W. Beard, "Discrete-time average consensus under switching network topologies," in *Proceedings of American Control Conference*, 2006, pp. 3551–3556.

[8] M. Pease, R. Shostak, and L. Lamport, "Reaching agreement in the presence of faults," *Journal of the ACM*, vol. 27, no. 2, pp. 228–234, 1980.

[9] M. J. Fischer, N. A. Lynch, and M. S. Paterson, "Impossibility of distributed consensus with one faulty process," *Journal of the ACM*, vol. 32, no. 2, pp. 374–382, 1985.

[10] D. Dolev, C. Dwork, and L. Stockmeyer, "On the minimal synchronism needed for distributed consensus," *Journal of the ACM*, vol. 34, no. 1, pp. 77–97, 1987.

[11] C. Dwork, N. Lynch, and L. Stockmeyer, "Consensus in the presence of partial synchrony," *Journal of the ACM*, vol. 35, no. 2, pp. 288–323, 1988.

[12] A. A. Moreira, A. Mathur, D. Diermeier, and L. A. N. Amaral, "Efficient system-wide coordination in noisy environments." *Proceedings of the National Academy of Sciences of the United States of America*, vol. 101, no. 33, pp. 12 085–12 090, 2004.

[13] M. Mitchell, P. T. Hraber, and J. P. Crutchfield, "Revisiting the edge of chaos : evolving cellular automata to perform computations," *Complex Systems*, vol. 7, pp. 89–130, 1993.

[14] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[15] R. Olfati-Saber and J. S. Shamma, "Consensus filters for sensor networks and distributed sensor fusion," in *Proceedings of IEEE Conference on Decision and Control and European Control Conference*, 2005, pp. 6698–6703.

[16] A.-L. Barabási and A. Reka, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

[17] P. Gacs, G. L. Kurdyumov, and L. A. Levin, "One-dimensional uniform arrays that wash out finite islands," *Problemy Peredachi Informaacii*, vol. 14, pp. 92–98, 1978.

[18] M. Brameier and W. Banzhaf, "Explicit control of diversity and effective variation distance in linear Genetic Programming," in *Genetic Programming*, ser. LNCS, 2002, vol. 2278, pp. 37–49.

[19] C. Gershenson, "Updating schemes in random Boolean networks: Do they really matter," in *Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems*, 2004, pp. 238–243.

[20] J. Aspnes, "Randomized protocols for asynchronous consensus," *Distributed Computing*, vol. 16, no. 2-3, pp. 165–175, 2003.

[21] ——, "Fast deterministic consensus in a noisy environment," *Journal of Algorithms*, vol. 45, no. 1, pp. 16–39, 2002.