Innovations in Computer Science 2010

Reaching Consensus on Social Networks

Elchanan Mossel[1] Grant Schoenebeck[2]
[1]Weizmann Institute of Science and UC Berkeley
[2]UC Berkeley
mossel@stat.berkeley.edu grant@cs.berkeley.edu

Abstract: Research in sociology studies the effectiveness of social networks in achieving computational tasks. Typically the agents who are supposed to achieve a task are unaware of the underlying social network except their immediate friends. They have limited memory, communication, and coordination. These limitations result in computational obstacles in achieving otherwise trivial computational problems.
One of the simplest problems studied in the social sciences involves reaching a consensus among players between two alternatives which are otherwise indistinguishable.
In this paper we formalize the computational model of social networks. We then analyze the consensus problem as well as the problem of reaching a consensus which is identical to the majority of the original signals. In both models we seek to minimize the time it takes players to reach a consensus.

Keywords: consensus; social networks

1 Introduction

We formalize social networks as restricted computational models and study their computational power. Our goal is to analyze their computational complexity and find optimal algorithms for some natural computational tasks. This may provide a comparison to the actual behavior displayed by people on social networks. In particular: do real behaviors match the performance of the best possible algorithms on social networks? What properties of a social network dictate the time it takes for the network to perform certain simple, computational tasks both theoretically and in practice?

To this end, we define a model of social network computation parameterized by the amount of memory allocated to each agent and the amount of communication with its neighbors. We then study two computational tasks. The first we call the *coordination problem*, where the agents must all agree on one of two otherwise indistinguishable colors. This problem is sometimes called the consensus problem. The second problem, which we call the *majority coordination problem* is a strengthening of the first, where the consensus reached must be identical to the majority of the original signals. These problems are both trivial from game theoretic point of view. From a (global) computational point of view the coordination problem is trivial and the majority coordination problem is not very difficult either. Both game theory and computational complexity view these problems as trivial. However they do not predict how the agents will arrive at solutions is the social network setup.

To motivate these problems consider a group of loosely connected friends who would like to go to a movie tonight. The problem is that there are two movie theaters, and they will only have fun if everyone goes to the same theater. We assume no one cares which theater they go to, nor is there any precedent for one theater over the other. The friends will happen into each other throughout the day at school (or online) and need to all know where to go by the end of school. What protocol should they run?

Alternatively the friends may have a true preference between the two theaters and their wish now is to go to the theater that is preferred by the majority. What protocol should be used in this case?

Variants of the example above are natural in modern social networks where the friendship graph is well defined and interactions are limited by the underlying medium.

1.1 Motivating and Related Work

Our work is partially motivated and informed by past and on-going works studying related problems from different angles. Most relevant is the work from experimental sociology. Latané and L'Herrou [15] study clustering, consolidation, and minorities groups using a similar problem they call the *conformity game* played by human subjects in four stylized networks. In the conformity game, each player is paid a fixed amount if his final label agrees with the most popular final label. The 24 players are unaware of structure of the network except for their immediate neighborhood. In their experiment there are 4 rounds where players

can update their color. In 42 of 48 games, the players failed to all converge uniformly. Instead they often clustered into two groups. Even though the number of rounds was small (4), players often got stuck in a local minimum before the final round–situations where all people believed they were in the majority. The authors note: "Participants failed to win a bonus more then 25% of the time, which seems particularly inept, considering that at least half were guaranteed a win by the rules of the game." By understanding what are the optimal algorithms for the problem, we can make more accurate statements about how well people do.

Kearns, Judd, Tan, and Wortman [9] study the coordination problem with human subjects. In their experiment, some players are paid an additional bonus when the consensus reached is "blue", and others are paid the bonus when the consensus is "red" (still no one is paid anything unless a consensus is reached). They study how different graph structures, payment schemes, and placement of like agents on the network influence whether a consensus is reached, and if so, which one. In their experiment, each player can switch between two possible labels as many times as he likes and can see the currently chosen labels of each of his neighbors (plus a few other pieces of information, including the degrees of each neighbor). The players have 60 seconds to converge, or they will all receive no payoff for that round.

Both Kearns, Suri, and Montfort [10] and a follow-up work by Enemark, McCubbins, Paturi, and Weller [5] studied a coloring game with human subjects. In the coloring game, each agent must play labels (colors) different from the labels (colors) of each of his neighbors. The subjects were paid a fixed amount if *all* the agents were successful, and received nothing otherwise. The problem of finding such a coloring with c colors of a general graph is **NP**-complete in the worst case when $c \geq 3$. Obviously this makes the comparison between human behavior and the "best known algorithm" somewhat problematic. In the conclusion section we make some comments relating the work on coloring social networks and work in algorithms and statistical physics on natural dynamics for coloring.

The aforementioned results of [15], [9], and [5] were all in close agreement to the computational simulations using very simplified models.

Other than the experimental work in sociology, problems such as the one studied here were studied in a number of different areas.

The *voter* model has been studied in Statistical physics since the 1970's [3, 7]. Reformulated in the language of the current paper this is an algorithm for the coordination problem which is defined as follows. Whenever two people interact, if their opinions are different then at random one of the two adopts the view of the other. It is immediate to see that this process must converge to a coordinated view. Moreover, the time of convergence is known and it is in general quadratic in the size of the graph. For more details on the model, see [16].

The majority coordination topic was studied before in several works. Kearns and Tan [11] created a protocol where each node requires $O(\log(n))$ memory and converges in time $O(n^7)$ (this protocol is in the "vertex model" to be defined later). This protocol basically runs the voter algorithm repeatedly and takes the majority vote. They show that this is correct with high probability. Benezit, Thiran and Vetterli [2] create an algorithm in the edge model with 2-bits of memory and show that if there is a strict majority that the algorithm will almost surely converge (with no time bound). Meanwhile, a results of Land and Belew [14] shows that no 1-bit memory automata works and so the result of [2] is tight with respect to memory.

Our work is also related to work in several different areas.

1.2 Related Work

There are a number of related but distinct problems that were studied in other areas. This is explained next.

1.2.1 Economics

Our work is naturally related to work in economics on learning on networks. In this work several models are suggested where multiple agents on a network converge to the same value where agents follow strategies aimed at maximizing their utility. There are several differences from the problem studied here. In the economics models the signals given to different players are all biased towards a "true" value while here there is no bias nor a true value. In much of the economics literature the agents know the structure of the graph while here they do not. Finally the economics models require actions which maximize expected utility where here the only goal is to arrive at global coordination.

A different body of work in economics involves graphical games. In these models, adjacency in the graph reflects the fact that a neighbors play effects ones utility. Note that in the problems presented here the utility actually depends on the actions of all players while the graph signifies the underlying information structure.

For references and an overview of the related economics literature, see chapters 8 and 9 in [8].

1.2.2 Distributed Computing

The coordination problem is of natural interest in distributed computing. The basic goal is for all entities in a network to arrive at a coordinated value. Several techniques that we employ are commonly uses in distributed computing. We will point of some of the relationships along the way. However, the standard assumptions in distributed computing are very minimal in terms of the symmetry between different states. For example in the standard setting in distributed computing the only requirement about the value eventually arrived at is that at least one of the nodes had this value to start with. So some of the standard coordination algorithms converge to the minimal value among the original values assigned to nodes. For further references in distributed computing see [19], [12], [1], and [17].

1.3 Our Framework

Our goal is to understand the computational power of social networks compared with more standard computational models. To this end we do the following:

- We do not consider the competitive / game-theoretic aspect of the problem. So all players have the same payoff.
- We consider only the very simple computational tasks of coordination and coordinated majority.
- We assume that except for their immediate neighbors the agents have very rough knowledge about global network parameters such as its diameter or size.
- We require the models to respect a strong symmetry between the signals (see Section 2.1 on Model for a more rigorous definition). This is done in order to model that the two items are indistinguishable (e.g. there may not be a natural ordering), and to exclude some trivial algorithms for the problems suggested here e.g., for the coordination problem, the algorithm that always declares red.

From a different perspective our model is stronger than those introduced in [15] and [9]. Both papers study problems very similar to the coordination games, where the agents have extremely limited communication with each other. In [15] the agents can only send a color and a confidence, and in [9] they can only communicate their current label. Our model allows richer spaces of communication between players. This is motivated by the experiments in [5] where the players often try to enrich their set of signals [4]. For example, they may toggle rapidly between two colors to signal that they are indifferent between those two labels.

1.4 Our Results

Our main technical results show that for many models it is the *broadcast time* of a graph that determines the time to convergence, and in other models it is simply the *diameter*. The broadcast time is the expected the time for a message to reach everyone starting from a single player (assuming that every player broadcasts the to all his neighbors upon receipt). This is novel since previous work in this area arrived at algorithms whose running time depended in a polynomial way on the number of vertices in the graph. As it is widely believed that many social networks are "small world" networks, it is expected that the diameters and broadcast time are (poly) logarithmic in the size of the graph.

Our main results show that

- The coordination problem can be solved in expected time $O(\omega)$ with expected $O(1)$ memory, where ω is the broadcast time.
- The majority coordination problem can be solved in expected time $O(n^3)$, with only 1 bit of additional memory.
- The majority coordination problem can be solved in expected time $O((d + \log(n))\log(n))$ with $O(\log(\Delta))$ memory, where d is the diameter and Δ is the maximum degree.

Some of the faster algorithms make some mild assumptions on the players approximate knowledge of the network such as knowledge of the diameter of the network up to a constant factor, or knowledge of the log of the size of the network up to a constant factor.

Remark: The work in [5, 10] deals with the problem of graph coloring. Even though graph coloring is known to be NP-hard, the computational hardness parameterized by the graph structure is in general poorly understood. At the conclusion section we briefly discuss some speculations regarding the graph coloring problem on social networks.

1) Road Map

In Section 2 we define our model and the problems studied. In Section 3 we present our results on the coordination problem. In Section 4 we present our results on the majority coordination model. In Section 5 we conclude with future directions for research in this area.

2 Schema and Notation

We model evolution on these networks in three ways: asynchronous edge dynamics, asynchronous vertex dynamics, and synchronous vertex dynamics.

We model an *edge dynamics network* as a three tuple $N = (V, E, W)$ where

- V is the set of agents,
- $E \subseteq V \times V$ is the set of relationships between the agents,
- $W = \{w_e\}_{e \in E}$ and $w_e \in \mathbf{R}_{\geq 0}$ is a set of rates for each edge.

We can similarly define a *vertex dynamics network* by assigning rates to the vertices. We say that a network is *uniform* if all its rates are 1.

2.1 The Model

We model an *asynchronous edge dynamics* as a five tuple (N, Σ, S, A, T) where

- N is an edge dynamics network.
- $\Sigma = \Sigma_e \times \Sigma_i$ is the set of possible states of each vertex divided into external states and internal states.
- $S \subseteq \Sigma$ is set of possible start states.
- A is a piece of advice that each agent has access to which can be thought of as a noisy value of some network parameter (e.g. an approximate size of the network). Note that *we do not count this in the memory of the agent* because he cannot modify it, and all our dynamics are designed so that the advice need only be approximately correct. Finally, for ease of notation, we assume that all agents receive the same advice, this is not necessary, and all the results apply to the setting where each agent has his own noisy version of A.
- T is a possibly probabilistic update function of the following form: let Δ be the maximum degree in N, then $T : \Sigma \times \Sigma \times A \times [0,1] \to \Sigma \times \Sigma$ is of the form $T = T' \times T'$ where $T' : \Sigma \times \Sigma_e \times \{1, \ldots, \Delta\} \times A \times [0,1] \to \Sigma$. The map T' takes the current state, the neighbor external state, the neighbor's position, some random bits and produces a new state. When an edge rings we apply the map T' for each of the endpoints of the edge. We will sometimes omit the last two arguments of T for ease of notation.

A *configuration* $\mathbf{s}$ of a dynamics is an assignment of states for all the vertices of graph: $\mathbf{s} : V \to \Sigma$. We will usually denote $\mathbf{s} = \{s_v\}_{v \in V}$. A *starting configuration* $\mathbf{s}$ of a dynamics is a configuration where $s_v \in S$ for all $v \in V$.

Let $\mathbf{s}^*$ be some starting configuration, and let $\gamma = \{\gamma_v\}_{v \in V}$ where a $\gamma_v : \Gamma(v) \to \{1, \ldots, |\Gamma(v)|\}$ is a permutation which labels the neighbors of v.

Given such $\mathbf{s}^*$ and γ a *run* of an asynchronous edge dynamics is a random process (over the randomness of T and when the edges "ring") whose range is a set of mappings $\mathbf{X} = \{X_v\}_{v \in V}$ where $X_v : \mathbf{R}_{\geq 0} \to \Sigma$. $X_v(t)$ is the state of vertex v at time t. The process is defined by first letting $\mathbf{X}(0) = \mathbf{s}$. The value $\mathbf{X}(t)$ remains constant in time until the Poisson process at an edge e rings. The process at each edge "rings" at rate w_e. When the edge $e = (u, v)$ rings at time t, the state of the vertices incident to the edges are updated to be

$$\begin{aligned} T(s_u, s_v, a, r) = \\ (T'(s_u, (s_v)_e, \gamma_v(u), a, r), \\ T'(s_v, (s_u)_e, \gamma_u(v), p_u, r)), \end{aligned}$$

where

$$\mathbf{s} = \{s_v\}_{v \in V} = \{X_v(t-)\}_{v \in V}$$

is the state just prior to the ring, $a \in A$ is the advice, and $r \in [0,1]$ is the randomness.

We note that we can also define an *asynchronous vertex dynamics model* where the vertices ring. In this case N is a vertex dynamics network, and T is a set of functions $T = \{T_d\}_{d=0}^{\Delta}$ where Δ is the max degree of the graph. $T_d : \Sigma \times (\Sigma_e)^d \times P \times [0,1] \to \Sigma$ represents one player updating her state based on her neighbors' external states. Given a start state $\mathbf{s}^*$ and permutation $\gamma = \{\gamma_v\}_{v \in V}$ where a $\gamma_v : \{1, \ldots, |\Gamma(v)|\} \to \Gamma(v)$, we similarly define a run as a random variable $\mathbf{X}$ such that $\mathbf{X}(0) = \mathbf{s}^*$. This process is updated when any vertex v's clock rings according to a Poisson clock with rate w_v. Every time a vertex v's clock rings, s_v is changed to $T_{deg(v)}(X_v(t-), (X_{\gamma_v(1)}(t-))_e, \ldots, (X_{\gamma_v(deg(v))}(t-))_e, a, r)$, where $a \in A$ is the advice, and $r \in [0,1]$ is the randomness.

Finally, we can also define a *synchronous vertex dynamics model*. This is like the asynchronous vertex dynamics model however at each time step t, the function T is applied simultaneously to all vertices.

We remark that most of our results carry over to the vertex models, but leave detailed study of them to a future time. In this paper we focus on the asynchronous edge dynamics and remark about the other models when it is convenient.

Definition 1. For dynamics (N, Σ, S, A, T), let $\Sigma_b, \Sigma_r \subseteq \Sigma$ be two disjoint sets. We say that the dynamics *is symmetric with respect to* Σ_b and Σ_r if there exists a permutation $\pi : \Sigma \to \Sigma$ such that $\pi(\Sigma_b) = \Sigma_r$; π preserves internal, external, and start states ($\pi(\Sigma_i) = \Sigma_i$, $\pi(\Sigma_e) = \Sigma_e$, and $\pi(S) = S$); and such that $T(\pi(s_1), \pi(s_2), a, r) = (\pi \times \pi)(T(s_1, s_2, a, r))$ for all $s_1, s_2 \in \Sigma$, $a \in A$, and $r \in [0,1]$.

We note that our models are natural for social networks, and similar models have been proposed in the literature for example, see simulations in [18] and [13]. The synchronous model is similar to models in distributed systems, and many of our results will carry over to this setting.

Remark 2. *Note that our asynchronous models are quite different from the asynchronous models in distributed computing. The distributed computing models have broadcast time on edges which does not follow a distribution. Instead it is only guaranteed to be bounded. The performance of algorithms in distributed computing is measures by the total amount of time divided by the longest broadcast time of an edge, while our algorithms the time is measured without any normalization. Indeed for Poisson type models the longest broadcast time is of order* $\log n$ *and it is exactly this* $\log n$ *factor that some of our algorithms try to save.*

To understand the difference between the two different models, consider broadcast where there are many paths of equal length between two nodes. In the distributed computing setup no speed up is gained. This contrasts with our asynchronous models where if many paths exists between two nodes, the expected time it takes for a broadcast message to travel between those two nodes is greatly reduced. Additionally, our asynchronous models provides no worst case bound for any event.

2) Equivalence of Models

Any dynamics in the synchronous vertex dynamics model can be simulated by a uniform dynamics in either asynchronous dynamics model. The two propositions below are related to the "synchronizer" problem studied in distributed computing which studies how to simulate synchronous distributed algorithms on asynchronous distributed networks (see, for example, Chapter 6 in [19]).

Proposition 3. *Any synchronous vertex dynamics can be simulated by a uniform asynchronous dynamics with* $\log(n)$ *slow down (in expectation) and each agent's public memory growing to twice the size plus two bits.*

Proof. To simulate in the asynchronous vertex model each vertex simply keeps copy of the external state he was in during the previous step and an additional state of which step he is in modulo 4. To see how this works first imagine that each vertex keeps all his previous state and the step count his is on. An agent only performs a computation if his state is less than or equal to all his neighbors (in performing a step he updates his current state according to the synchronous rules and updates his history and step count in the natural way). Because each agent only performs an update when he is less than or equal to all his neighbors, and his neighbors save all their previous state, he will have all the information required to make this update, and this provides a faithful simulation. But notice that if each agent only updates when he is at a step less than or equal to all he neighbor states, then it will always be the case that his neighbor states are equal to, one more or one less then his state. Thus it is sufficient to keep around only the previous state and only the step count modulo 4.

It is expected that in each $\log(n)$ steps each vertex rings at least once, and so after $k\log(n)$ steps, we expect each vertex to be at step count at least k.

Proposition 4. *Any dynamics in the synchronous vertex dynamics model can be simulated by a uniform dynamics in the asynchronous edge model with* $\log(|E|) \leq 2\log(n)$ *slow down and each agent* v*'s public memory growing to twice its size plus two bits, and each agent* v*'s private memory growing by a factor of* $|\Gamma(v)|$*, where* $\Gamma(v)$ *denotes the neighbors of* v*.*

Proof. To simulate in the asynchronous vertex model each vertex keeps track of which step he is on (modulo 4) and a copy of the external state he was in during the previous step as well as the external state for each neighbor agent for the current step he is on and a bit indicating if that state is current.

To see how this works first imagine that each vertex keeps all his previous state and the step count his is on. An agent only performs a computation if he has up to date information on the external states of all his neighbors (in performing a step he updates his current state according to the synchronous rules and updates his history and step count in the natural way and clears the history for his neighbors). Otherwise, he will simply update this information if possible.

Because each agent only performs an update when he is less than or equal to all his neighbors (this must be the case for his information to be up to date), and his neighbors save all their previous state, he will have all the information required to make this update, and this provides a faithful simulation. But notice that if each agent only updates when he is at a step less than or equal to all he neighbor states, then it will always be the case that his neighbor states are equal to, one more or one less then his state. Thus it is sufficient to keep around only the previous state and only the step count modulo 4.

It is expected that in each $\log(|E|) \leq 2\log(n)$ steps each edge rings at least once, and so after $k\log(|E|)$ steps, we expect each vertex to be at step count at least k.

It is unclear if the asynchronous models can simulate each other. The issue is that when you simulate in a manner like above, you destroy the independence

of the ring times. Also, it is unclear what it even means claim equivalence of asynchronous models that are not uniform. However, many dynamics that are not overly dependent on the order of rings will work in all models with no real modification.

2.2 Further Notation

We will always use n to denote $|V|$, d for the diameter of the underlying network, and $\Gamma(v)$ the neighbors of a vertex v. If $\mathcal{E}$ is some property of a configuration in the network configuration model, we will denote by $\tau(\mathcal{E})$ the first time an event occurs perpetually in the dynamics (so immediately before $\tau(\mathcal{E})$ the property does not hold, but after $\tau(\mathcal{E})$ the property always holds). In most of the considerations below $\mathcal{E}$ will be an event such that if it holds at time t then it holds at all later times. Note that for such events $\tau(\mathcal{E})$ is the first time the event holds.

For vertex v, let $B_v(t)$ be the set of vertices that, at time t, could have possibly received a message from v given the sequence of vertex/edge rings. Similarly, for vertex v, let $B'_v(t)$ be the set of vertices that, at time t, could have possibly sent a message to v given the sequence of vertex/edge rings.

We define the *broadcast time* of a dynamics to be $\omega = \max_{v \in V} \mathbf{E}[\tau(B_v(t)) = V)]$. Note that for every vertex v: $\omega/2 \leq \mathbf{E}[\tau(B_v(t)) = V)] \leq \omega$. For the synchronous model, the broadcast time will simply be the diameter of the graph.

Let $\mathcal{U}$ be the event that for every pair of vertices $u, v \in V$ $B_u(t) \cap B_v(t) \neq \emptyset$. Let $\mathcal{U}'$ be the event that for every pair of vertices $u, v \in V$ $B'_u(t) \cap B'_v(t) \neq \emptyset$.

We define the *collision time* of a dynamics to be $\eta = \mathbf{E}[\tau(\mathcal{U})] = \mathbf{E}[\tau(\mathcal{U}')]$

We define $\mathcal{C}$ to be the event of coordination (or majority coordination depending on the context).

2.3 Problems Considered

We consider two basic problems:

In the first, the *Coordination Problem*, given network N, we would like to design a dynamics (N, Σ, S, A, T) where Σ is partitioned into two special disjoint subsets Σ_r and Σ_b (which stand for red and blue), such that eventually, either all the agents' states are in Σ_r, or all the agents' states are in Σ_b. We will additionally require that 1) the dynamics are symmetric with respect to Σ_r and Σ_b (see Section 2.1 for precise definition) and 2) if all the states input to T are in Σ_r, then all the states output by T must be in Σ_r as well, (and similarly for Σ_b). Let $\mathcal{C}$ be the event that either all the agents' states are in Σ_r or all the agents' states are in Σ_b. For such a dynamics, we would like to study the expected time to consensus, which we define as $\max_{\mathbf{s},\gamma} \mathbf{E}[\tau(\mathcal{C})]$ where $\mathbf{s}$ is the initial state and γ are the set of permutations defined in Section 2.1. We would like to minimize this value, usually over some class of dynamics. Note that the agents need not be aware that they are in a consensus. All that is required is that they do not leave it.

The second, the *Majority Coordination Problem* is like the *Coordination Problem* but with one additional property. Given network N, we would like to design a dynamics (N, Σ, S, A, T) where Σ is partitioned into two special disjoint subsets Σ_r and Σ_b (which stand for red and blue), such that if the *majority* of the initial states are in Σ_r, then eventually, all the agents' states are in Σ_r; similarly for Σ_b; if there is a tie, then we just require a consensus. We will additionally require that 1) the dynamics are symmetric with respect to Σ_r and Σ_b (see Section 2.1 for precise definition) and 2) if all the states input to T are in Σ_r, then all the states output by T must be in Σ_r as well, (and similarly for Σ_b). Let $\mathcal{C}$ be this event. For such a dynamics, we would like to study the expected time to majority consensus, which we define as $\max_{\mathbf{s},\gamma} \mathbf{E}[\tau(\mathcal{C})]$ where $\mathbf{s}$ is the initial state and γ are the set of permutations defined in Section 2.1. We would like to minimize this value, usually over some class of dynamics. Note again that the agents need not be aware that they are in a consensus. All that is required is that they do not leave it.

We note that all of the dynamics considered in the paper will have the additional desired property: If the original network is in consensus then the dynamics will not change the value of the consensus.

3 The Consensus Problem

The problem of reaching a consensus was studied in statistical physics for a specific model called the *voter model*. In this each vertex is one of two states, when an edge "rings", using the common randomness one of the end points incident to the edge is chosen to copy the state of the other.

Definition 5. The *Voter Model asynchronous edge dynamics* (N, Σ, S, A, T) are defined so that $\Sigma = \Sigma_e = S = \{+1, -1\}$, $A = \emptyset$, $T(s_1, s_2) = (s_1, s_1)$ with probability $1/2$ and (s_2, s_2) with probability $1/2$.

The *Voter Model asynchronous vertex dynamics* (N, Σ, S, A, T) are defined so that $\Sigma = \Sigma_e = S = \{+1, -1\}$, $A = \emptyset$, $T(s_v, (s_{\gamma_v(1)}, \ldots, s_{\gamma_v(deg(d))} = s_{\gamma_v(i)})$ with probability $1/deg(d)$ for each $1 \leq i \leq deg(d)$.

Theorem 6. *For any connected network in the uniform asynchronous edge dynamics model, there ex-*

ists a memoryless dynamics independent of the graph which reaches consensus in expected time n^2.

The theorem is a result of the following propositions whose statement and proofs are variants of classical results on the voter model which proven using martingale arguments.

Proposition 7. *Consider the edge model with rates 1 and the voter model on G. Let λ be the number of edges in the smallest cut in G. Then for any initial configuration the process will converge to the same color with probability at least $1/2$ by time at most*

$$T_1 = \frac{n^2}{2\lambda},$$

and with probability at least $1 - 2^{-k}$ by time kT_1.

The proof is very similar to previous voter model proofs.

Proof. Note that $X(t) = \sum_v X_v(t)$ is a bounded martingale. This follows from the fact that whenever an edge is chosen and the two end points are not identical, it is equally likely that the value of X will increase or decrease by 1.

Let $P(t)$ be the minimum over all initial configurations of the probability of convergence by time t.

By the orthogonality of martingale increments it follows that for all $t, h \geq 0$:

$$E[X^2(t+h)] = E[X^2(t)] + E[(X(t+h) - X(t))^2].$$

Therefore writing $\Psi(t) = E[X^2(t)]$ it follows that

$$\Psi'(t) = \lim_{h \to 0} h^{-1} E[(X(t+h) - X(t))^2].$$

We lower bound $\Psi'(t)$ by conditioning on the configuration at time t and noting that for small h, if at time t the process hasn't converged then we expect one edge between a vertex labeled by 1 and an edge labeled by -1 to ring with probability at least $h\lambda$ and therefore as $h \to 0$ the conditional expectation of $(X(t+h) - X(t))^2$ is at least $4h\lambda$. We thus conclude that

$$\Psi'(t) \geq 4\lambda(1 - P(t)).$$

So if $P(t) \leq 1/2$ we obtain:

$$n^2 \geq \Psi(t) \geq t2\lambda,$$

so

$$t \leq \frac{n^2}{2\lambda},$$

as needed.

In the asynchronous vertex model, when a vertex rings, it copies the state of a random neighbor.

Proposition 8. *For the vertex model the following holds. For any initial configuration the process will converge to the same color with probability at least $1/2$ by time at most*

$$T_2 = \frac{(\sum_v d(v))^2}{2 \min_{e=(u,v)} d(u) + d(v)}$$

and with probability at least $1 - 2^{-k}$ by time kT_2.

The proof is very similar to previous voter model proofs.

Proof. The proof for the vertex model is similar to that of Proposition 7. We only need to find the "right" martingale. We will look at $\sum_v d(v) X_v(t)$ where $d(v)$ is the degree of v. In order to show that this is a martingale consider an edge (u, v) where $X_u \neq X_v$. This edge can be chosen by choosing one of the two end points. At rate $d(v)^{-1}$ the value of $X_v(t)$ will be replaced by $X_u(t)$ and at rate $d(u)^{-1}$ the value of $X_u(t)$ will be replaced by $X_v(t)$. Summing the expected differences we get:

$$d(v)^{-1} d(v)(X_v(t) - X_u(t)) + d(u)^{-1} d(u)(X_u(t) - X_v(t)) = 0.$$

This established that $X(t)$ is a martingale. Define $\Psi(t) = \mathbf{E}[X^2(t)]$ as before. Consider $X(t+h) - X(t)$ for small h and an edge (u, v) that rings between time t and $t + h$ contributes

$$(d(v)^{-1} d(v)^2 + d(u)^{-1} d(u)^2)(X_v(t) - X_u(t))^2 = (d(v) + d(u))(X_v(t) - X_u(t))^2.$$

Therefore we have:

$$\Psi'(t) \geq 4 \min_{e=(u,v)} (d(u) + d(v)).$$

So if $P(t) \leq 1/2$ we obtain:

$$(\sum_v d(v))^2 \geq \Psi(t) \geq 2t \min_{e=(u,v)} (d(u) + d(v)).$$

so

$$t \leq \frac{(\sum_v d(v))^2}{2 \min_{e=(u,v)} d(u) + d(v)},$$

as needed.

In the next result we provide an optimal time algorithm, but the amount of memory needed is $O(\log(n))$. We include this theorem for its simplicity. Algorithms with best performance will be obtained

later. For the algorithm, we must *assume the players know the log of the size of the network up to a constant factor* to get optimal parameters. Note that these are very reasonable assumptions in many social network settings.

Definition 9. The Greatest-Element asynchronous edge dynamics (N, Σ, S, A, T) are defined as follows: Let $A = \mathbf{N}$, Let $a \in A$ be the advice such that $a = c\log(n)$. Let $\Sigma = \Sigma_e = \mathbf{Z} \setminus \{0\}$. $S = \{-1, 1\}$. We define $T(s_1, s_2, a, r)$. The outcome is (s_i, s_i) where $i = 1$ if $|s_1| > |s_2|$, $i = 2$ if $|s_2| > |s_1|$, and i is randomly chosen to be 1 or 2 if $|s_1| = |s_2|$. If s_1 and/or s_2 is $+1/-1$ then they first modify their states to be a choose a random non-zero number between $2s_i$ and $(2^{5a} + 1)s_i$, and then use T as defined above.

The story of what happens is that each agent generates reasons (of varying quality) to his sign. Each vertex relays the best reasoned message he has seen, or a random one if two are equally well reasoned. If there is honest confusion about the best reason, then it will take a while to converge.

Theorem 10. *The Greatest-Element asynchronous edge dynamics (N, Σ, S, A, T) will reach consensus in expected time $\omega + n^{4-5c}$ with memory $5c\log(2n + 2))$ where advice $a = c\log(n)$. In particular, if $c \in [1, 5]$, then it will converge in expected time $\omega + 1$ with memory $25\log(2n + 2))$.*

Proof. Assume for the analysis, that each vertex starts off with a random number between $-2^{5a} - 1$ and $+2^{5a} + 1$ that has absolute value greater than 1. We first condition on the fact that there is a unique such number of highest absolute value. In this case, it will take time ω to spread, because we are broadcasting from that vertex. Now condition on the fact that there is no unique highest initial internal state. In expected time ω, every agent will have some such message, but perhaps not associated with the same color. Now we are running the voter model, and so will converge in expected time n^2, by Proposition 7.

Let q be the probability there does not exists a unique number of highest absolute value. If we show that $q \leq n^{2-5c}$, then the running time is at most $\omega + qn^2 = \omega + n^{4-5c}$. However, the probability that all the numbers are unique is $1(1-\frac{1}{2^{5a}})\cdots(1-\frac{n-1}{2^{5a}}) \geq 1 - \frac{n^2}{2^{5a}}$ and so $q \leq n^{2-5c}$.

In the next result, we provide dynamics which are expected to converge only a constant factor slower than the broadcast time, and in addition only use an expected finite about of memory, *assuming the players know the diameter of the network and the sum of the rates on the entire network up to a constant factor.* Even if the player know nothing at all about the network, it uses only $O(log(n))$ memory and expects to converge with a $O(log(n))$ slow down.

Definition 11. The *Wait-and-See asynchronous edge dynamics* (N, Σ, S, A, T) are defined as follows: Let $A = \mathbf{R}_{\geq 0}$, $a = c\omega\sum_{(u,v)\in E} w_{(u,v)}$. Let $\Sigma = \Sigma_e = \mathbf{Z} \setminus \{0\}$. $S = \{1, -1\}$. We now define the transition function for vertices v_1 and v_2 in states s_1 and s_2 respectively. $T(s_1, s_2) =$:

- If $|s_1| > |s_2|$ then s_2 becomes s_1. Similarly, If $|s_2| > |s_1|$ then s_1 becomes s_2.
- If $|s_1| = |s_2|$ and they are odd, then with probability $1/2$, v_1 flips a coin and with probability $1/(2^{\lfloor s_1/2 \rfloor}a)$ and increases the magnitude of its state by 1, thereby moving to an even state and "electing" himself. With the remaining $1/2$ probability, v_2 does likewise.
- If $s_1 = s_2$ and they are both even, do nothing.
- If $|s_1| = |s_2|, s_1 \neq s_2$ and they are even, then each vertex increases the magnitude of his state by 1.

The story behind these dynamics are that initially no one wants to venture an opinion (odd state). At some point, an agent gets tired of waiting and decides on his outcome (the sign represents the chosen outcome, the state being even represents the fact that he has chosen). He then broadcasts the outcome to his neighbors. As long as no one else decides to do the same (move to even) before hearing of this decision a consensus will be reached. Otherwise, if someone has decided on one outcome (even positive) and he talks with his neighbor who has decided on a different outcome (even negative), he will then broadcast to everyone to abandon their past decision (odd with magnitude +1). The amount of time that each agent waits depends on how fast he thinks his message will spread to the network, and on how many other people are making the same decision as him. If there is an error initially, each agent will be more patient the next time.

We note that even if the agents all receive advice 1 (which is essentially no information), then the dynamics still perform reasonably well.

Theorem 12. *The Wait-and-See asynchronous edge dynamics (N, Σ, S, A, T) will reach consensus in expected time $O(\omega(\log(\frac{1}{c}) + c))$ with memory $O(1 + \log(\frac{1}{c}))$ where c is such that $a = c\omega\sum_{(u,v)\in E} w_{(u,v)}$. In particular, if c is constant, then a consensus is reached in expected time $O(\omega)$ with expected memory $O(1)$, and if $a = 1$ (i.e. the agents have no information about the graph) then a consensus is reached in*

expected time $O(\omega(\log(\alpha)+\frac{1}{\alpha}))$ *with a expected memory* $O(1+\log(\alpha))$ *where* $\alpha = \omega(\sum_{(u,v)\in E} w_{(u,v)})$

Proof. Recall $\mathcal{C}$ is the event of consensus, and $\tau(\mathcal{E})$ is the first time an event $\mathcal{E}$ happens perpetually. Let $\mathcal{E}_k$ be the event of having a state of absolute value at least k. We call the *maximum state* of a configuration the absolute value of the state with the maximum absolute value. Let $2k^*+2$ be the maximum state when consensus is reached (note that it will be an even state with probability 1).

We begin with the following claim which bounds the expected time to reach consensus in terms of 1) the expected time starting with maximum state $2k+1$ to reach either consensus or maximum state $2k+3$, and the probability that maximum state $2k+2$ is ever reached.

Claim 13.

$$\boldsymbol{E}(\tau(\mathcal{C})) \leq \sum_{k=0}^{\infty} \boldsymbol{E}[\tau(\mathcal{C}\vee\mathcal{E}_{2k+3})|k^*\geq k]\Pr[k^*\geq k]$$

Proof.

$$\begin{aligned}
&\mathbf{E}(\tau(\mathcal{C}))\\
&=\sum_{k=0}^{\infty}\mathbf{E}[\tau(\mathcal{C})|k^*=k]\Pr[k^*=k]\\
&=\sum_{k=0}^{\infty}\Big(\mathbf{E}[\tau(\mathcal{C})-\tau(\mathcal{E}_{2k+1})|k^*=k]\\
&\quad+\sum_{\ell=0}^{k-1}\mathbf{E}[\tau(\mathcal{E}_{2\ell+3})-\tau(\mathcal{E}_{2\ell+1})|k^*=k]\Big)\cdot\\
&\quad\Pr[k^*=k]\\
&=\sum_{k=0}^{\infty}\mathbf{E}[\tau(\mathcal{C})-\tau(\mathcal{E}_{2k+1})|k^*=k]\Pr[k^*=k]\\
&\quad+\sum_{\ell=0}^{\infty}\sum_{k=\ell+1}^{\infty}(\mathbf{E}[\tau(\mathcal{E}_{2\ell+3})-\tau(\mathcal{E}_{2\ell+1})|k^*=k]\Pr[k^*=k]\\
&=\sum_{k=0}^{\infty}\mathbf{E}[\tau(\mathcal{C})-\tau(\mathcal{E}_{2k+1})|k^*=k]\Pr[k^*=k]\\
&\quad+\mathbf{E}[\tau(\mathcal{E}_{2k+3})-\tau(\mathcal{E}_{2k+1})|k^*\geq k+1]\Pr[k^*\geq k+1]\\
&=\sum_{k=0}^{\infty}\mathbf{E}[\tau(\mathcal{C}\vee\mathcal{E}_{2k+3})-\tau(\mathcal{E}_{2k+1})|k^*\geq k]\Pr[k^*\geq k]
\end{aligned}$$

It remains to bound $\mathbf{E}[\tau(\mathcal{C}\vee\mathcal{E}_{2k+3})|k^*\geq k]$ and $\Pr[k^*\geq k]$

Claim 14. $\boldsymbol{E}[\tau(\mathcal{C}\vee\mathcal{E}_{2k+3})|k^*\geq k]\leq\omega(c2^k+2)$

Proof. We define four types of configurations:

1 Max state is $2k+1$ is odd.

2 Max state is $2k+2$ is even and all agents in this state have the same sign.

3 Max state is $2k+2$ is even and all agents are in this state have the same sign (e.g. consensus is reached).

3' Max state is $2k+2$ is even and different agents in this max state have different signs.

Assume that we are in configuration type 1) with maximum state $2k+1$. Configuration type 1) must transition to configuration type 2), the only question is how long it will take. The conversion will take place as soon as a vertex incident to an edge between two state $2k+1$ vertices elects itself. The expected time for the state $2k+1$ to spread everywhere is ω. Once this has happened, the expected time until an vertex elects itself to maximum state $2k+2$ is $c\omega 2^{\lfloor 2k+1\rfloor}$. This is because each edge e has a poisson clock with rate w_e, however the rate at which either vertex incident on the edge elects itself is $\frac{w_e}{a2^{\lfloor 2k+1\rfloor}}$ because a vertex incident to the edge only appoints itself with probability $\frac{1}{a2^{\lfloor 2k+1\rfloor}}$. Thus all the clocks together have a combine effective rate of $\sum_{e\in E}\frac{w_e}{a2^{\lfloor 2k+1\rfloor}}=\frac{1}{c\omega 2^k}$, so the expected time until an vertex appoints itself is $c\omega 2^k$.

Once in configuration type 2) the expected time for the edges rings to be such that the state $2k+2$ spreads everywhere is ω. Once this happens, either, we have reached consensus, and are in configuration 3), or another vertex has elected himself, and we are in configuration 1) again.

Putting this together the expected time to transition is at most $\omega(c2^k+2)$.

Define c' and k_0 as follows: if $c\geq 64$ then $c'=c$ and $k_0=0$; if $c<64$ then fix k_0 so that $c2^{k_0}=c'\in[64,128)$, and thus $k_0\leq\max\{7-log(c),0\}$.

Claim 15. *For* $k\geq k_0$ $\Pr[k^*\geq k+1|k^*\geq k]\leq\frac{1}{4}$. *In particular,* $\Pr[k^*\geq k_0+k']\leq(\frac{1}{4})^{k'}$ *and* $\boldsymbol{E}[k^*]\leq k_0+2$.

Proof. Look at the time of the first maximum state $2k+2$. We will show that with probability $\frac{1}{4}$ no other vertex elects itself before this state spreads to the entire graph. Fix parameter $\Delta=\sqrt{c2^k}$. The probability that the initial $2k+2$ state spreads to the entire graph in time $\Delta\omega$ is $1-\Delta^{-1}$ by Markov.

Some edge between two $2k+1$ states will elect itself at rate at most $\frac{1}{\omega c2^{\lfloor 2k+1\rfloor}}$ (see calculation in Proof of Claim 14). Thus the probability of a second self election in time $\Delta\omega$ is at most $1-\exp(-\frac{1}{\omega c2^k}\Delta\omega)\leq\frac{\Delta\omega}{\omega c2^k}=\Delta^{-1}$.

So consensus is reached with probability $1-2/\Delta = 1-2/\sqrt{c2^k} \geq 1-2/\sqrt{c'} \geq 1-\frac{1}{4}$

Putting things together, we bound $\mathbf{E}[\tau(\mathcal{C})]$. Using Claims 13 and 14 we see that it is enough to bound $\sum_{k=0}^{\infty}(2+c2^k)\omega \Pr[k^* \geq k]$

$$\begin{aligned}
&\sum_{k=0}^{\infty}(2+c2^k)\omega \Pr[k^* \geq k] \\
= \;& 2\omega\mathbf{E}[k^*] + \omega\sum_{k=k_0}^{\infty}\Pr[k^* \geq k_0]\left(c2^k\right) \\
= \;& 2\omega\mathbf{E}[k^*] + \omega\sum_{k=0}^{k_0-1}c2^k + \sum_{k=k_0}^{\infty}\Pr[k^* \geq k_0]\left(c2^k\right) \\
\leq \;& \omega\left(2k_0+4+c2^{k_0}+\sum_{k=0}^{\infty}\left(\frac{1}{4}\right)^k\left(c2^{k_0+k}\right)\right) \\
\leq \;& \omega\left(2k_0+4+3c2^{k_0}\right) \leq \omega(406+\log(\frac{1}{c})+3c) \\
= \;& O(\omega(\log(1/c)+c)
\end{aligned}$$

We note that the amount of memory required is $\log(2k^*+1)$, and $\mathbf{E}[\log(2k^*+1)] \leq O(1+\log(1/c))$.

The dynamics are symmetric with respect to Σ_r being the positive states and Σ_b the negative. To see this, define the map $\pi(s) = -s$.

The next result shows that in the synchronous model, there is a simple processes with 3 bits of memory that allow the agents to coordinate in time related to the diameter (broadcast time) of the graph *assuming the players know the product of the diameter and the network and the size of the network up to a constant factor.*

Definition 16. We define the Wait-and-See synchronous dynamics (N, Σ, S, A, T) as follows: Let $A = \mathbf{N}$, let $a \in A$ such that $a = c(d+1)n$ (where d is the diameter of the network). Let $\Sigma = \Sigma_e = \{free, error, reset\} \times \{blue, red\}$. $S = \{(free, red), (free, blue)\}$. We will write $s_v \in \Sigma$ as $s_v = (p_v, c_v)$ where $p_v \in \{free, error, reset\}$ is the "phase" and $c_v \in \{blue, red\}$ is the "color". $T'(s_v, s_{\gamma_v(1)}, \ldots, s_{\gamma_v(deg(v))}, a, r) =$

- If $p_v = reset$, output $p_v = free$.
- Else, if $p_v = error$, output $p_v = reset$.
- Else, if $error \in \{p_{\gamma_v(1)}, \ldots, p_{\gamma_v(deg(v))}\}$, output $p_v = error$.
- Else, if $\{(set, red), (set, blue)\} \subseteq \{s_v, s_{\gamma_v(1)}, \ldots, s_{\gamma_v(deg(v))}\}$ output $p_v = error$.
- Else, if $p_v = free$ and $red \in \{c_{\gamma_v(1)}, \ldots, c_{\gamma_v(deg(v))}\}$, output $s_v = (set, red)$. Similarly if $p_v = free$ and $blue \in \{c_{\gamma_v(1)}, \ldots, c_{\gamma_v(deg(v))}\}$, output $s_v = (set, blue)$.
- Else, if $p_v = free$, with probability $1/a$ output $p_v = set$. In this case we say that agent v "elects" himself.

Theorem 17. *The Wait-and-See Synchronous dynamics will reach consensus in expected time*

$$c(d+1) + d + \frac{1}{c-1}(c(d+1)+2d)$$

where $a = c(d+1)n$. This is constant if c is also a constant greater than 1.

Proof. At any particular time we define k to be the maximum number of times that any vertex has been in the *error* phase. Now at any time t we classify the configuration into 3 regimes:

1. Some of the vertices have been in the *error* phase k times, and no vertex that has, has elected himself after being in the *error* phase k times. Additionally, not all vertices are in the $free$ phase.
2. All of the vertices have been in phase *error* k times and are now in the $free$ phase.
3. Some vertex has elected himself after being in the *error* phase k times.

Fix a run, and consider any two neighbors u and v. If at time t vertex u is in phase *error*, then it must be the case that v was also in phase *error* at time $t-1$, t, or $t+1$ (and vice versa). However each vertex can only be in the *error* phase every 3 steps (because after an agent is in phase *error* he transitions to phase *reset* and then $free$). This creates a bijection between time steps when v and u are in phase *error*. Thus, in any run, each vertex is in the *error* phase for the same number of steps. Moreover, if vertex v enters the *error* phase for the kth time at time t, then any vertex u of distance ℓ from v must enter the *error* phase for the kth time by time $t+\ell$

We first claim that it will take time at most time $d+2$ to move from the 1st regime to either the second or the third. If some vertex is in the *error* phase for the kth time at time t, by the above reasoning all vertices will have been in the *error* phase for the kth time at time $t+d$. Once each vertex has been in the *error* phase exactly k times, by the above reasoning, every vertex which neighbors an vertex in the *error* phase, is in either the *error* phase or the *reset* phase. Thus after 2 more steps (assuming no vertex that has been in the *error* phase k times elects itself) every vertex will be in the $free$ phase.

To move from regimes 2 to regimes 3 takes expected time $c(d+1)$ because whilst in phase $free$, each vertex elects himself with probability $\frac{1}{a} = \frac{1}{c(d+1)n}$ each round. There are n such vertices, so the expected time before a vertex elects itself is $c(d+1)$.

Once in regime 3 we claim that with probability $\geq 1 - 1/c$ we arrive at consensus before incrementing k. In this case, we claim that we arrive at consensus in at most d more steps. Say we go through and compute the updates for the vertices one at a time. Let v be the first vertex to elect himself and have been in the *error* phase k times, and say this happens at time t. Because v must be in the $free$ phase, he was in the *error* phase at least two units of time ago. Thus every vertex at distance ℓ from v has been in the *error* phase k times by time $t + \ell - 2$. Thus if no other vertex elects himself in the next d steps, then each such vertex at distance ℓ from v will be in phase $free$ at time $t + \ell$ and will receive the *red/blue* message from v. If however, another vertex u elects herself after having been in the *error* phase k times, then the messages will meet in time less than d, and if they are different colors, the configuration will return to regime 1. The probability that no other vertex elects herself between time t and $t + d$ is $(1 - 1/a)^{n(d+1)} \geq 1 - 1/c$. This means the expected value of k upon reaching consensus is $\frac{1}{c-1}$. Thus the total expected time is: $c(d+1) + d + \frac{1}{c-1}(c(d+1) + 2d)$

We will now show a lower bound. While the upper bounds work for the broadcast time, the lower bounds only apply to the collision time. In the synchronous model the collision time is just $1/2$ the broadcast time, which is the diameter. While we do not show that these are related in the asynchronous setting, we suspect they are, at least, in many natural settings.

Theorem 18. *For any dynamics $\boldsymbol{E}[\tau(\mathcal{C})] \geq \boldsymbol{E}[\tau(\mathcal{U})]/2$, even when each vertex starts assigned red or blue randomly.*

Theorems of this flavor are often attributed to folklore in distributed computing (see [19]).

Proof. Let R_{ring} be the space of randomness for the edge rings. Let R be the space of randomness for the initial configuration and the transitions. Let $\tau(\mathcal{U}, r_{ring}) = \mathbf{E}_r[\tau(\mathcal{U})|r_{ring}]$, and let $\tau(\mathcal{U}, r_{ring}, r) = \mathbf{E}_r[\tau(\mathcal{U})|r_{ring}, r]$ neither of which contain any randomness. To prove the theorem, it is enough to show that for every $r_{ring} \in R_{ring} : \mathbf{E}_r[\tau(\mathcal{C})|r_{ring}] \geq \tau(\mathcal{U}, r_{ring})/2$, and this is what we will show. Recall that for any fixed r_{ring}, there exists $u, v \in V$ such that for any time $t < \tau(\mathcal{U}, r_{ring})$ we have $B'_u(t) \cap B'_v(t) = \emptyset$, where $B'_u(t)$ is the set of vertices that could have sent a message to agent u at time t. Define the bijection f on R that takes the vertices in $B'_u(t)$ and reverse their initial colors.

Let $A \subseteq R$ be such that if $r \in A$ then $\mathbf{E}_r[\tau(\mathcal{U})|r_{ring}, r] \leq \tau(\mathcal{U}, r_{ring})$. Note that if $r \in A$ then $f(r) \notin A$ because if the colors of i and j match at time $\tau(\mathcal{U}, r_{ring})$ with randomness r, then they are opposites with randomness $f(r)$ because the color of u has flipped, and the color of v has remained the same.

Because f is a bijection, this means that at most half of r is in A, and thus for every $r_{ring} \in R_{ring} : \mathbf{E}_r[\tau(\mathcal{C})|r_{ring}] \geq \tau(\mathcal{U}, r_{ring})/2$.

4 Majority Coordination Problem

Unlike the coordination problem, the majority coordination problem is impossible to do without additional any memory [14]. Intuitively, if there were some dynamics, then it would have to have some transition from red to blue with non-zero probability. But this transition could only operate on local information, so you could embed many of these states inside a larger graph with one more red then blue. With some probability the dynamics would change a red to blue, and then the graph would have a majority blue, so from this point it would have to go to all blue.

However, as the next section shows, you can do this with just one additional bit of memory.

4.1 Achieving Majority via Weak and Strong Votes

In this subsection we analyze a simple algorithm that reaches a majority consensus. As was communicated to us by David Xiao, our algorithm is very similar to the one previously suggested in [2]. However, we strengthen and extend the work in [2] by providing an algorithm which converges even in cases where the number of red and blues are equal. Further our analysis provides explicit convergence time bounds.

Definition 19. The *Strong Weak Voter asynchronous edge dynamics* (N, Σ, S, A, T) is defined as follows. Let $\Sigma = \{-2, -1, +1, +2\}$, $S = \{-2, +2\}$, $A = \emptyset$, and $T(s_u, s_v) =$

- If $s_u = s_v$, the two states remain the same.
- If $|s_u| > |s_v|$ output $(s_u/2, s_u)$; similarly if $|s_v)| > |s_u|$ output $(s_v, s_v/2)$. (weak voters follow the sign of strong voters and then they switch places.
- If $s_u = -s_v$ output $(1, 1)$ and $(-1, -1)$ with equal probability. (Two strong voters cancel each other out and become weak, and weak voters run the voting model).

We prove the following

Theorem 20. *The Strong Weak edge dynamics $N = (N, \Sigma, S, A, T)$ will reach majority consensus in expected time $O(n^3)$ with each vertex v having 2-bits of memory.*

Proof. The analysis of the algorithm proceeds in phases. The goal of the first phase is to eliminate either all 2 or all -2. Note that since the only way to eliminate a 2 is by interaction with a -2 which makes them both disappear, by the end of this phase we have the following:

- If there were equal number of 2 and -2's in the original signal then there are no 2 and -2 remaining.
- Otherwise, by the same reasoning we will have 2 remaining if they were the majority in the original signal and -2 if they were the majority.

Suppose that vertex v is assigned 2 at time 0 and vertex u is assigned -2. Note that unless the 2 or -2 become 1 or -1, both of them preform random walks on the graph G. Furthermore, these walks are independent except when u, v are adjacent and the edge connecting them is ringing. By the cat and mouse game analyzed in Aldous-Fill book section 6.4.3 it follows that the two random walks will meet in time $O(n^2)$ in expectation.

Since the process above can be repeated for any pair of $2, -2$ it follows that by expected time $O(n^3)$ phase 1 has terminated.

We now proceed to phase 2. There are two cases to consider. In the first case there are no $2, -2$ present. In this case we just perform the voter model which is expected to converge by time $O(n^2)$ (see Proposition 7).

In the second case, there are some 2's present and all other vertices are labeled by $1, -1$. We claim that here the process will converge to all 1's and 2s in time $O(n^3 \log n)$. Let $X(t)$ denote the number of positive vertices at time t, where time 0 denotes the first time where there are no -2's. The analysis is done via comparison of $\mathbf{X}(t)$ (the normal run) to an auxiliary process denoted $\mathbf{Y}(t) = \{\mathbf{Y}_v(t)\}_{v \in V}$. $Y_v(t)$ takes the values $1, -1$ only. At time 0 we set $Y_v(0) = sign X_v(0)$. Moreover, $\mathbf{Y}$ follows the same choice of edges as $\mathbf{X}$. When an edge rings $\mathbf{X}$ does the following: it performs a standard voter model update. However, if the update leads to all -1 configuration then it is canceled. Let $Y(t)$ denote the number of positive vertices at time t, where time 0 denotes the first time where there are no -2's is $\mathbf{X}$.

It is easy to see that $X(t) \geq Y(t)$ for all t. $\mathbf{Y}(t)$ performs a random walk reflected at the point where there is exactly 1 positive elements. Therefore standard random walk estimates imply that $Z(t)$ converges to n in expected time $O(n^2)$.

These dynamics are also symmetric because if we assign the positive states to one partition (think red), and the negative states to the other (think blue), then the map $\pi(s) = -s$ preserves symmetry.

4.2 Majority Coordination in Time Related to the Diameter

Unlike in the coordination problem, here we only obtain algorithms with running time that depends on the diameter (not the broadcast time). It turns out our solution is cleaner in the synchronous case, and so we present that first. The synchronous algorithm can then be adapted to the asynchronous edge version via a slight modification to the generic reduction given in Theorem 4. (A common technique in distributed computing).

The idea is to run the Wait-And-See consensus dynamics but instead of passing a color, point to the neighbor that you first saw colored (which we now call the *passingup* phase), and thus form a tree of depth at most d with the elected vertex as the root. The tree then sums up the number of each color, least significant to most significant bit. The root looks at which has the greatest most significant bit and passes the decision down the tree.

One problem is that the Wait-And-See consensus dynamics may fail to elect a single leader, and two trees may form. In this case, if the trees produce different colors, then we can restart.

What makes the problem tricky in the asynchronous case is that a spanning tree formed by broadcasting may have depth much greater than the diameter of the tree (if there are many long paths between two vertices, but only a few short paths). Such a tree is very inefficient to route messages across.

The intuitive idea for the algorithm is simple and borrows heavily from related literature; the formal exposition, however, is somewhat cumbersome.

Definition 21. The *Synchronous Wait-And-See Majority dynamics* (N, Σ, S, A, T) are defined as follows:

Let $A = \mathbf{N}$ and $a \in A$ be such that $a = cn(d + 1)$

for some c. Let

$$\Sigma = \Sigma_e =$$
$$color \in \{blue, red\}$$
$$colorsaved \in \{blue, red\}$$
$$state \in \left\{ \begin{array}{l} free, passingup, outofbits \\ passingdown, error, reset \end{array} \right\}$$
$$parent \in \Gamma(v) \cup \emptyset$$
$$digitmod4 \in \{0,1,2,3\}$$
$$carryred \in \mathbf{Z}$$
$$carryblue \in \mathbf{Z}$$
$$bitred \in \{0,1\}$$
$$bitblue \in \{0,1\}$$

$$S = \{(red, red, free, \emptyset, 0, 0, 0, 0, 0), (blue, blue, free, \emptyset, 0, 0, 0, 0, 0)\}$$

For some agent v, we denote by K_v the set of other agents with v as their *parent*.

Let $\mathbf{s} = \{s_v\}_{v \in V}$ be the current state. $T(s_v, s_{\gamma_v(1)}, \ldots, s_{\gamma_v(deg(d))})$ is defined as follows:

If $state_v = reset$ then
  $color_v = colorsaved_v$;
  $parent_v = \emptyset$;
  $state_v = free$;
  $bitred_v = bitblue = 0$;
  if $colorsaved_v = blue$ then
    $carryblue_v = 1$;
    $carryred_v = 0$;
  if $colorsaved_v = red$ then
    $carryred_v = 1$;
    $carryblue_v = 0$;
Else if $state_v = error$ then
  set $state_v = reset$;
Else if ($state_u = error$ for any $u \in \Gamma_v$) then:
  set $state_v = error$;
Else if ($state_v = state_{\gamma_v(1)} = \cdots$
    $= state_{\gamma_v(deg(v))} = free$) then
  with probability $\frac{1}{a}$ set $state_v = passingup$;
Else if ($state_v = free$ and
    $state_u = passingup$ for any $u \in \Gamma_v$) then
  $state_v = passingup$;
  $parent_v = u$; (break ties arbitrarily)
Else if ($state_v = passingup$ and
    $state_u \neq free$ for any $u \in \Gamma_v$) then
  if ($state_u = outofbits$ for all $u \in K_v$ and
    $bitblue_v = bitred_v = 0$ and
    $carryblue_v = carryred_v = 0$) then
    $state_v = outofbits$;
    else if (($digitmod4_v = digitmod4_{p_v}$ or
      $p_v = \emptyset$)
      and for all $u \in K_v$
        $digitmod4_u = digitmod4_v + 1 (mod 4)$) then
    $redbit_v = (redcarry_v + \sum_{u \in K_v} redbits_u) \mod 2$;
    $redcarry_v = \lfloor (redcarry_v + \sum_{u \in K_v} redbits_u)/2 \rfloor$;
    update $bluebit_v$ and $bluecarry_v$ similarly.
    $digitmod4_v = digitmod4_v + 1 \mod 4$
    if$redbit_v > bluebit_v$ then
      $color_v = red$;
    if$bluebit_v > redbit_v$ then
      $color_v = blue$
Else if $state_v = outofbits$ then
  if $parent_v = \emptyset$
    $state_v = passingdown$;
  else if $state_{parent_v} = passingdown$ then
    $state_v = passingdown$;
    $color_v = color_{parent_v}$;
Else if ($state_v = passingdown$ and
    for some $u \in \Gamma_v$:
      $state_u = passingdown$ and
      $color_v \neq color_u$) then
  $state_v = error$.

Theorem 22. *The Wait-and-See Majority synchronous dynamics $N = (N, \Sigma, S, A, T)$ will reach consensus in expected time $\frac{c}{1-c}(c(d+1)+4d+2\log(n))$ with each vertex v requiring memory $O(\log(|\Gamma(v)|))$ where $a = cn(d+1)$. Note that if $c > 1$ is a constant then this is $O(d + \log(n))$.*

Proof. After expected time $c(d+1)$ a vertex will elect himself. If no other vertices elect themselves, then after t additional steps the nodes exactly t away from the self-elected vertex will be joined to a tree with the self-elected vertex node at the root. If more vertices elect themselves, there will be more than one tree formed. Upon entering this tree, each node will enter the *passingup* state.

We claim that eventually the reds and the blues are summed up the tree. We will show the following:

1 Each node is always computing the same digit or one greater then its parent and the same digit or one less then its children.
2 Each node correctly computes what it reports. That is when it enters into a particular $digitmod4 = \ell$ for the hth time, then the *redbit* (or *bluebit*) is correctly computing the $4(h-1)+\ell$th bit of the number of red bits (or blue bits) in his subtree.
3 Each node at depth k computes the rth bit no later than $2d-k+2r$ bits after the vertex's election.

1) Is by contradiction. This is the way that things

begin because all nodes start computing 0 mod 4. Assume this ever fails to be the case. Then either a parent is a 2 digits behind a child, or a child is a digit behind his parent. But the rules explicitly forbid a child computing more than 1 ahead of his parent or parent computing further than his children.

3) Then follows in part from 2). Because parent and child never compute bits which are further distant than 1, the *digitmod*4 will always keep them aligned with each other.

4) To see this, image that the tree has depth d along every path. Now this tree computes slower then the actual tree. However, on this tree it is easy to see by induction on t, where t is the number of time steps after the election of the original vertex, that: at time $t = d$ all the vertices will be in the *passingup* state (and computing bit 0); and after time $t \geq d$, a vertex at depth k is computing bit $\lceil (t - 2d + k)/2 \rceil$ if this is positive, and bit 0 otherwise. Each node at depth k computes the rth bit no later than $2d - k + 2r - 1$ steps after the vertexes election. Thus all the bits are computed by the root in time $t = 2d + 2\log(n) - 1$ after the election.

It follows that the root will enter $state = outofbits$ at time at most $2d + 2\log(n)$ because after $\log(n) + 1$ digits, the correct answer is 0. The last color that the root has will be the last bit when either *blue* or *red* was greater, and thus will indicated whether there are more blues or reds in the tree. Thus eventually the root will pass down this element.

If there is no unique tree, then the above will happen, but perhaps on several trees. If the trees all come to the same answer (red or blue), then they will remain. Otherwise, they will enter the *error* state.

Once an *error* state occurs, then in the next $d + 1$ steps, each agent will be in *error* for exactly 1 step unless an agent that has been in *error* elects himself in the future. For the sake of contradiction, let v be the first node to be in *error* twice. Let t_1 be the first time he was in *error*, and t_2 the second time. It is clear that the first time that his neighbors were in *error* were $t_1 - 1$, t_1, or $t_1 + 1$. However, $t_2 > t_1 + 3$ because agent will be in state *reset* at time $t_1 + 1$ and thus unable to be in error in state $t + 2$. But then agent v must have had a neighbor in error in time $t_2 - 1$ or greater, and so that neighbor must have been the first to be in error twice.

Thus, after an agent enters *error* he will again be in state $free$ until another agent elects himself. This will happen in expected time at most $c(d + 1)$ after there are no *error* states left. Notice that because each agent is in error exactly once, the newly elected *passingup* cannot propagate to reach the previous error. Thus we are in the situation before, where with probability that another vertex elects himself before entering the *passingup* state is at most $\frac{1}{c}$.

The probability that two nodes elected themselves is at most $1/c$, thus the number of times this must be repeated until there is no error is $\frac{c}{1-c}$

Thus the total expected time is: $\frac{c}{1-c}(c(d+1)+4d+2\log(n))$

Note that the only things that need to be stored that incur super constant memory are *parent* which require memory $\log(|\Gamma(v)| + 1)$, and the *carryred* and *carryblue* states, which also require at most $\log(|\Gamma(v)|+1)$ (because adding i, j-bit numbers results in a value of at most $i2^j$ which has at most $\log(i) + j$ bits, and so $\log(i)$ carry bits.

The dynamics are symmetric between Σ_r where $color = red$ and Σ_b where $color = blue$, as can be seen by the permutation π which maps $color = r$ to $color = b$ and swaps states *bitblue* and *bitred*, as well as swapping *carryblue* and *carryred*.

Theorem 23. *The Wait-and-See Majority asynchronous edge dynamics* (N, Σ, S, A, T) *will reach consensus in expected time* $\frac{c}{1-c}(c(d + 1) + 4d + 2\log(n))\log(n)$ *with each vertex* v *requiring memory* $O(\log(|\Gamma(v)|))$ *where* $a = c(d + 1)n$.

Proof. It remains to extend this to the case of asynchronous edges. By simulating in the naive way (see Proposition 4), we see that we can do it with $\log(n)$ slow down and by increasing our memory by a factor or at most $|\Gamma(v)|$. However, we can do better, because the only external information that in edge needs to compute its next state is: if a neighbor state is in *error*, if all neighbor states are $free$, if a neighbor state is *passingup*, if no neighbors are free, if all children are $outofbits$, if parent is on same *digitmod*4 and all children are on one greater *digitmod*4, if a parents state is *passingdown*, if there is neighbor of state *passingdown* with a different color, the sum of the red and blue bits the children. All this information can be stored in $\Gamma(|v|) + 10$ bits of private information.

5 Conclusion and Future Research

We mention some of the future research directions. These conceptual problems are interdisciplinary. All have an important computational component but important aspects of these problems come from sociology, and statistical physics.

The main challenge is to find models that accurately simulate or predict human behavior. Put very coarsely, in what situations do people behave like particles and/or small state automata? The empirical results of [5, 9, 10, 15] agreed well with computer simula-

tions that simulated models related statistical physics ([15], [5], [9]).

In particle systems and automaton models, the time to converge is much slower than in our memory-bounded dynamics models. It would be interesting to see what happens in more realistic situations. In particular is the time to converge polynomial in the diameter of the graph or in the number of nodes? Would the time to convergence be predicted better by the diameter, or some sort of "mixing time". Of course, it may be that none of these models predicts reality well.

In part, our study begins to provide a theoretical foundation and understanding of the empirical results of [5, 9, 10, 15]. To this end, we simplify the model in several ways. This leads to several dynamics, some of which seem more robust/realistic than others. It would be interesting to capture this in a more rigorous fashion. Our model has several simplifications that may gloss over important concerns. Future work could relax some of these assumptions:

- We assume that everyone uses and agrees on the same original dynamics. It would be very interesting to model (perhaps through some kind of notion of evolvability (see [20]) how agents could come to agree on a particular protocol.
- We use bounded memory models to simulate humans. Of course, humans can compute much more, and at the same time, perhaps, less than these models. Perhaps using different computational bounds, or introducing a notion of error into the calculations or communications would better model reality.
- We attempt to minimize the expected time. In all the aforementioned experiments there was instead a deadline. Perhaps the optimal dynamics change when the deadline is very small or very large.
- We do not have any game theory or selfish incentives in our models. While the problems that we study do not require it, there are many problems that do.

5.1 The Coloring Model and Random and Planted Sat

The coloring model has attracted a lot of attention in the social network literature. In this section we discuss some features of these experiments and potential explanations. One of the basic question which was looked at was how does the problem change as the structure of the network changes?

As the network for the coloring game changes, two things happens. First, each agents has access to different information. Secondly, the underlying coloring problem changes, perhaps getting easier or harder. It is hard to differential one effect from the other. Did the network become more powerful or did the problem just become easier? From the sociology point of view there are very good reasons to study this, because these models capture the real-world situations of anti-correlation (see [10]). However, in this study we simply things to just study the effect of the network graph.

In [5] they note that there are "good" edges, and "bad" edges. The former "good edges" make the color game more easily solvable by giving the network more information while not actually restricting the set of solutions. These edges are already implied by the constraints of the graph. The latter "bad edges" restrict the solutions space of colorings. They then confirm in experiments that adding containing edges make the solutions harder (it is solved less often or takes more time) and that adding redundant edges make the problem easier. There is a large middle range of edges that are neither entirely easy nor hard and the effect of these is not studied

Viewed from complexity stand point the same phenomenon had already been observed for many problems, including many satisfiability problems. For example the work in [6] implies that message passing algorithms for planted SAT problems are more rapidly converging for higher densities of formulas. In fact such sat problems go from easy (at low densities) to hard, and back to easy again. Intuitively we expect that the same behavior will be displayed in real social network, esp. since this behavior is expected even for very simple local algorithms such as MCMC and message passing algorithms.

Already there are natural dynamics with no memory for the coloring model from statistical physics which was used for the empirical coloring results [4]. This coincides with the natural MCMC for coloring, i.e., the Glauber dynamics for low temperature antiferromagnatic Potts model. It would be interesting to try to employ related analytical tools on social network graphs with different topologies.

Acknowledgements

EM acknowledge the support by DMS 0548249 (CAREER) award, DOD ONR grant N0014-07-1-05-06, ISF grant 1300/08 grant PIRG04-GA-2008-239137. GS was supported by a National Science Foundation Graduate Fellowship and an internship at Microsoft Research New England (NERD). The authors would want to acknowledge the hospitality and interdisciplinary atmosphere at Microsoft NERD where this this paper was born.

References

[1] F. Bťenťezit, P. Denantes, A. G. Dimakis, P. Thiran, and M. Vetterli. Reaching consensus about gossip: convergence times and costs. In *Information Theory and Applications*, January 2008.

[2] F. Benezit, P. Thiran, and M. Vetterli. Interval consensus: from quantized gossip to voting. In *ICASSP 2009*, pages 3661 – 3664, 2009.

[3] P. Cliford and A. Sudury. A model for spatial conflict. *Biometrika*, 60(3):581–588, 1973.

[4] D. Enemark, July 2009. Personal and email coorespondence.

[5] D. Enemark, M. McCubbins, R. Paturi, and N. Weller. Good edge, bad edge: How network structure affects a group's ability to coordinate. In *ESORICS*, March 2009.

[6] U. Feige, E. Mossel, and D. Vilenchik. Complete convergence of message passing algorithms for some satisfiability problems. In *Proceedings of Random 2006*, pages 339–350. Springer, 2006.

[7] A. Holley and T. M. Liggett. Ergodic theorems for weakly interacting infinite systems and the voter model. *Ann. Probab.*, 3:643–663, 1975.

[8] M. O. Jackson. *Social and Economic Networks*. Princeton University Press, 2008.

[9] M. Kearns, S. Judd, J. Tan, and J. Wortman. Behavioral experiments on biased voting in networks. *Proceedings of the National Academy of Science*, January 2009.

[10] M. Kearns, S. Suri, and N. Montfort. An experimental study of the coloring problem on human subject networks. *Science*, 313:824–827, August 2006.

[11] M. Kearns and J. Tan. Biased voting and the democratic primary problem. In *Proceedings of the 4th International Workshop on Internet and Network Economics (WINE'08)*, pages 639–652, 2008.

[12] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. *Foundations of Computer Science, Annual IEEE Symposium on*, 0:482, 2003.

[13] G. Kossinets, J. Kleinberg, and D. Watts. The structure of information pathways in a social communication network. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 435–443, New York, NY, USA, 2008. ACM.

[14] M. Land and R. K. Belew. No perfect two-state cellular automata for density classification exists. *Phys. Rev. Lett.*, 74(25):5148–5150, Jun 1995.

[15] B. Latané and T. L'Herrou. Spatial clustering in the conformity game: Dynamic social impact in electronic groups. *Journal of Personality and Social Psychology*, 70(6):1218–1230, 1996.

[16] T. M. Liggett. *Interacting particle systems*, volume 276 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, New York, 1985.

[17] R. Olfati-saber, J. A. Fax, and R. M. Murray. Consensus and cooperation in networked multi-agent systems. In *Proceedings of the IEEE*, page 2007, 2007.

[18] J. P. Onnela, J. Saramäki, J. Hyvönen, G. Szabó, D. Lazer, K. Kaski, J. Kertész, and A. L. Barabási. Structure and tie strengths in mobile communication networks. *Proceedings of the National Academy of Sciences*, 104(18):7332–7336, May 2007.

[19] D. Peleg. *Distributed Computing: A Locally-Sensitive Approach*. SIAM Monographs, Philadelphia, USA, 2000.

[20] L. G. Valiant. Evolvability. *J. ACM*, 56(1):1–21, 2009.