Low Memory Distributed Protocols for 2-Coloring

Amos Israeli[1], Mathew D. McCubbins[3], Ramamohan Paturi[2] *, and Andrea Vattani[2]

[1] Netanya Academic College.
amos.israeli@netanya.ac.il
[2] University of California, San Diego.
{paturi,avattani}@cs.ucsd.edu
[3] University of Southern California
mmccubbins@marshall.usc.edu

Abstract. In this paper we present new distributed protocols to color even rings and general bipartite graphs. Our motivation is to provide algorithmic explanation for human subject experiments that show human subjects can achieve distributed coordination in the form of 2-coloring over networks with a simple communication protocol. All our protocols use low (often constant) memory and reach a solution in feasible (polynomial rounds) and sometimes optimal time. All the protocols also have short message length and use a broadcast communication strategy. Our contributions include two simple protocols RingGuess and GraphCoalescing for rings and general bipartite graphs, which can be viewed as candidates for natural human strategies. We present two other protocols RingElect and GraphElect which are optimal or nearly optimal in terms of the number of rounds (proportional to the diameter of the graph) but require somewhat more complex strategies. The question of finding simple protocols in the style of RingGuess and GraphCoalescing that run in time proportional to diameter is open.

1 Introduction

In this paper we present new and simple distributed algorithms for 2-coloring in a basic model of distributed computing where each node has only local information about the network and requires very low memory. Our work is motivated by recent studies of social networks in which human subject experiments for achieving distributed coordination were conducted [14, 11, 12, 15, 8]. In particular, the experiments in [11, 15, 8] focus on the 2-coloring problem. In these experiments, each subject is a node of the underlying network, which is 2-colorable, and subjects are motivated (in the form of cash payment) to achieve a legal 2-coloring of

* This research is supported by NSF grant CCF-0905645 from the Division of Computing and Communication Foundations. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

the network. Each subject communicates with its neighbors using colored flags, and can change his/her color as many times as he/she wants. The network topologies considered are bipartite (2-colorable) graphs. Kearns, Suri and Montfort [11] show that subjects in an experimental setting can indeed solve a distributed version of the graph coloring problem using only local information when they are incentivized to work collectively. McCubbins, Paturi and Weller [15] extend the work of Kearns et. al. by considering asymmetric incentives: if a proper coloring is achieved, the participants ending with a specic color are paid more than the participants ending with the other color. The effect during the game is that participants are reluctant to leave the special color for the other one and therefore proper colorings are reached in longer times.
The fact that human subjects are able to achieve proper colorings is somewhat surprising given the limited resources they have at their disposal during the experiments: in terms of memory, they can only rely on their individual memorization skills; in terms of knowledge of the network and communication, they just have a local perspective and can only interact with their immediate neighbors in the network.
The goal of this paper is to provide an algorithmic interpertation for the success of the human agents in solving the 2-coloring problem over bipartite networks. We ask whether there exist protocols that are natural in the sense of closely representing the experimental conditions. In particular, we ask whether there exist protocols that rely solely on local information, use small amount of memory, have no knowledge of the size of the underlying graph, broadcast messages, and employ simple logic so they can be viewed as candidates for human strategies.

1.1 Model

We consider a distributed message-passing model which is synchronous, anonymous and uniform. That is, nodes in the network do not have distinct identifiers and do not have knowledge of the size of the network (or other parameters such as diameter). Moreover, they all run the same protocol.
Given our assumptions of anonymity and uniformity of the network, there exists no (even randomized) protocol that computes a coloring of the network and *terminates* (this impossibility result about termination is famous for the leader election problem, e.g. [3], and it extends easily to the coloring problem). However, we observe that the experiments in [11, 15] do not require the human subjects to terminate but only to obtain a proper coloring within the allocated time. The low memory protocols in [16] for the consensus problem are also not concerned with guaranteeing termination.
We recall that for synchronous networks the running time of a protocol is given by the number of communication rounds. In each round a processor receives messages from its neighbors, performs a local computation, and sends messages to its neighbors. Finally, we remark that all our protocols work in a simple broadcast model, where a node sends the same message to all its neighbors. The broadcast model is more suitable for capturing the setting of the coloring experiments in [11, 15, 8]. Indeed, in coloring

experiments, the subjects are not allowed to communicate with each other except that they are allowed to change their color and observe the changes in the color of a neighbor. Thus, a subject can only communicate with the neighbors by signaling or broadcasting a color change.

1.2 Results

Our results consist of constant/low memory protocols to solve the 2-coloring problem on (even) rings as well as on general bipartite graphs, and to compute optimal colorings of preferential attachment graphs. We remark that these graphs encompass most of the topologies considered in the experiments of [11, 15, 8]. For the ring, we present two constant-memory protocols. The first one RingGuess is extremely simple and converges in time quadratic in the size of the network. RingGuess can be viewed as a candidate for a natural human strategy. The second protocol RingElect achieves optimality in terms of time (linear number of rounds) while only requiring constant memory. In addition, this protocol can elect a leader in optimal time[4].

For general bipartite graphs with n nodes and m edges, we present a protocol GraphCoalescing which can be viewed as a generalization of RingGuess,that computes a 2-coloring in $O(nm^2 \log n)$ (or $O(\Delta n^2)$ for Δ-regular graphs) rounds with ea ch node u using $O(\log \delta_u)$ bits of memory[5]. GraphCoalescing is also very simple and is a candidate for a human strategy. We also present the protocol GraphElect which employs a leader election strategy. GraphElect requires up to $O(\log n)$ memory without requiring the knowledge of n. We show that GraphElect computes 2-coloring in $O(\log n + D)$ (essentially optimal) rounds where D is the diameter of the graph. Finally, for bipartite graphs, we discuss how asymmetric incentives can slow down the convergence of the protocols.

Recently, Mossel and Schoenebeck [16] have presented low memory protocols for solving the consensus problem and a variation of it called the majority coordination problem. They analyze the consensus problem in social networks and parameterize social network computation by the amount of memory allocated to each node. Although their work is similar in spirit to ours in terms of the focus on low memory algorithms, their model diverges from ours in several aspects (see Section 1.1). Our focus on natural strategies calls for simplest possible protocols.

The work by Chaudhuri, Graham and Jamall [6] is also motivated by the coloring experiments in [11], but their setting is entirely different in that nodes never change their color. They show that a greedy strategy properly colors a network with high probability if the number of colors is at least $\Delta + 2$, where Δ is the maximum degree of the network.

[4] To the best of our knowledge, optimal leader election protocols are not known for our setting. (See section 6.3 of [18] for a summary of leader election protocols. Also see subsection **Model** for details about our setting.)

[5] Here δ_u denotes the degree of u.

2 Rings

Rings are among the most studied topologies in distributed coloring experiments as well as in distributed computing. In this section we analyze two constant-memory protocols for the ring topology. The first protocol, RingGuess, is natural and is a plausible candidate for subject strategies in rings. A slight variant of this protocol is also used in [11] as a comparative tool with respect to human subject performance[6]. We show that RingGuess converges to a 2-coloring in $\Theta(n^2)$ rounds (in expectation) in a ring with n nodes. The protocol does not involve any explicit message passing except that each node has access to the color of its neighbors. Its message complexity defined as the total number of color changes by all the nodes is bounded by $O(n^2 \log n)$.
Our analysis of RingGuess raises the question whether there exists a constant memory protocol that converges in linear number of rounds, which is clearly optimal for the ring. We present a new protocol, RingElect, to 2-color a ring which uses constant memory and converges in $O(n)$ rounds. RingElect employs a leader election strategy, and also elects a leader within the same resource bounds.
At the end of this section, we discuss how asymmetric incentives will slow down the protocols.

2.1 A natural protocol

Consider a situation that frequently occurs in the experiments of [11, 15, 8] when at some point during the game a subject sees a neighbor choosing the same color as his/hers. In this situation the subject may either change its color or wait for its neighbor to change its color. One could conceivably use timing strategies to make the decision. However it is not possible to implement timing strategies in bounded memory and without the knowledge of the size of the ring. As such, the most natural action is probably to change color with some (constant) probability.
With this motivation, we introduce protocol RingGuess: Initially, each node has an arbitrary color. Any node which has the same color as one of its neighbors repeatedly executes the following 2-round protocol:

1. Change your color with probability $p = \frac{1}{2}$, while memorizing your old color and the colors of your two neighbors.
2. If any of your neighbors changes its color during the first round, restore the previous color.

We now present the analysis of protocol RingGuess. Let a *conflict* be an edge with nodes of the same color, and the distance between two conflicts be the minimum number of edges that separates them. We observe that since a node with no conflicts does not change its color, and one with conflicts ends the 2-round protocol with a different color only if both

[6] In [11] the protocol bears the name *distributed heuristic* and works for any number c of colors. When restricted to the case $c = 2$ is essentially RingGuess but is used asynchronously. They simulate this algorithm on the networks used in human subject experiments to compare the steps required by the algorithm with the time to solve the problem by the subjects. No algorithmic analysis is provided in [11].

its neighbors did not change their color, the total number of conflicts on the ring never increases. The 2-round protocol RingGuess 'moves' the conflicts (clockwise or counterclockwise) with some probability. Also, when two conflicts have a node in common (i.e., 3 consecutive nodes have the same color), there is a probability of $p^3 = \frac{1}{8}$ that the two conflicts vanish – this happens when the middle node is the only one flipping its color.
The convergence proof of the protocol will make use of random walks. The following lemma bounds the number of steps for a random walk to terminate.

Lemma 1. *Let $\mathcal{W}_k = (X_0, X_1, \dots)$ be a unidimensional random walk on a path of nodes $1, 2, \dots, k, \dots$ starting on the the first node (i.e. $X_0 = 1$) and terminating when the k-th node is reached or surpassed. For $\delta \in \{1, 2\}$ and $j \geq 1$, consider the following transition probabilities:*

$$\begin{aligned}
\mathbf{P}_{j\to j-\delta} &= \begin{cases} q_\delta & \text{if } j-\delta > 0 \\ 0 & \text{if } j-\delta \leq 0 \end{cases} \\
\mathbf{P}_{j\to j+\delta} &= q_\delta \\
\mathbf{P}_{j\to j} &= 1 - \sum_{\delta\in\{1,2\}} (\mathbf{P}_{j\to j-\delta} + \mathbf{P}_{j\to j+\delta})
\end{aligned}$$

where $\mathbf{P}_{i\to j} = \Pr[X_{t+1} = j | X_t = i]$, $q_1, q_2 > 0$ and $2(q_1 + q_2) \leq 1$. Then the expected time for $\mathcal{W}_k$ to terminate is at most $\frac{k^2}{q_1+4q_2}$.

Proof (sketch). Let h_i be the expected value of the random variable representing the number of steps to reach (or surpass) state k from state i. Then the following system $\mathcal{S}^*$ of equations holds

$$\begin{aligned}
&h_1 = c_1 + q_1 h_2 + q_2 h_3 + (1 - q_1 - q_2) h_1 \\
&h_2 = c_2 + q_1(h_1 + h_3) + q_2 h_4 + (1 - 2q_1 - q_2) h_2 \\
&h_k = 0 \\
&h_{k+1} = 0 \\
&\text{For } 3 \leq j \leq k-1, \\
&h_j = c_j + q_1(h_{j-1} + h_{j+1}) + q_2(h_{j-2} + h_{j+2}) + (1 - 2q_1 - 2q_2) h_j
\end{aligned}$$

with $c_j = 1$, for $1 \leq j \leq k-1$.
Recall that our goal is to show that $h_1 \leq \frac{k^2}{q_1+4q_2}$. Let $h_1 = x_1^*, h_2 = x_2^*, \dots, h_{k+1} = x_{k+1}^*$ be the solution of this system S^*, and let $h_1 = \tilde{x}_1, h_2 = \tilde{x}_2, \dots, h_{k+1} = \tilde{x}_{k+1}$ be the solution of the system $\tilde{\mathcal{S}}$ obtained by setting $c_1 = 1$, $c_j = 2$, for $2 \leq j \leq k-1$, and replacing equation $h_k = 0$ with $h_k = \frac{k^2-(k-1)^2}{q_1+4q_2}$. We observe that it has to be $\tilde{x}_j \geq x_j^*$ for any $1 \leq j \leq k+1$. Using induction, we show $\tilde{x}_j = \frac{k^2-(j-1)^2}{q_1+4q_2}$ for $1 \leq j \leq k+1$. From this, we conclude that $h_1 = x_1^* \leq \tilde{x}_1 = \frac{k^2}{q_1+4q_2}$. □

We apply Lemma 1 to bound the number of rounds required for two conflicts in a ring to come close to each other. This will be the main ingredient to establish the main theorem.

Lemma 2. *Consider any 2-coloring with m conflicts such that no conflicts are at distance less than 2. Then, after at most $\frac{n^2}{2p^2m^2}$ expected number of rounds, there will be two conflicts at a distance less than 2.*

Proof. We observe that after an execution of the 2-round protocol, any conflict (independently of the others) will move one edge clockwise with probability p^2, one edge counterclockwise with probability p^2, and it will not change position with probability $1-2p^2$. Fix two consecutive conflicts and let D be a random variable representing the distance between them. After an execution of the 2-round protocol, D will (a) increase by 2, as well as decrease by 2, with probability $(p^2)^2 = p^4$; (b) increase by 1, as well as decrease by 1 with probability $2p^2(1-2p^2) = 2p^2 - 4p^4$; (c) not change with probability $1 - 2(p^4 + (2p^2 - 4p^4)) = 1 - (4p^2 - 6p^4)$.
We observe that the behavior of D can be interpreted as the random walk $\mathcal{W}_n$ in Lemma 1 using $q_1 = 2p^2 - 4p^4$ and $q_2 = p^4$ (We use $\mathcal{W}_n$ because the distance between two conflicts is always less than n.) Lemma 1 assures D will be less than 2 in at most $\frac{n^2}{q_1+4q_2} = \frac{n^2}{2p^2}$ expected number of rounds. Now, in order to prove the lemma, we will show that the expected time for two conflicts out of the total m conflicts to be at a distance less than 2 is no larger than the expected time for a random walk $\mathcal{W}_{\lfloor n/m \rfloor}$ to terminate. Consider the Markov chain $\bar{D}_0, \bar{D}_1, \ldots$, with $\bar{D}_t = (D_t^{(1)}, D_t^{(2)}, \ldots, D_t^{(m)})$, where $D_t^{(i)}$ is the random variable representing the distance between the i-th and $(i+1)$-st conflict on the ring at time t. $D_t^{(m)}$ represents the distance between the last and the first conflict at time t. We couple this Markov chain with another one, $M_t = \min_i D_t^{(i)}$, that keeps track of the distance between the closest pair of conflicts for $t \geq 0$. Now we observe that $M_t \leq \lfloor \frac{n}{m} \rfloor$, and that M_t approaches a value less than 2 at least as fast as the random walk $\mathcal{W}_{\lfloor n/m \rfloor}$ terminates. This observation along with Lemma 1 concludes the proof. $\square$

The main theorem of this section follows easily by Lemma 2.

Theorem 1. *Protocol* RingGuess *computes a* 2*-coloring of the ring in* $\Theta(n^2)$ *expected number of rounds and* $O(n^2 \log n)$ *bit complexity.*

Proof. By Lemma 2, starting with a configuration with m conflicts, it takes at most $\frac{n^2}{m^2}$ expected number of rounds for two conflicts to be at a distance less than 2 since $p = \frac{1}{2}$. We observe that when two conflicts are at a distance less than 2, there is a constant probability that the two conflicts vanish, decreasing the total number of conflicts by two. Therefore, the expected number of rounds for two conflicts to vanish is bounded by $O(\frac{n^2}{m^2})$. We conclude that the expected number of rounds to resolve all the conflicts is bounded by $\sum_{m=1}^{n} c\frac{n^2}{m^2} = O(n^2)$. Analogously the number of messages (or color changes) is bounded by $\sum_{m=1}^{n} 2cm\frac{n^2}{m^2} = O(n^2 \log n)$ since the expected number of color changes in a configuration with m conflicts is $2m$. The bound on the number of rounds is tight since if started with two conflicts at $\Omega(n)$ distance, it takes $\Theta(n^2)$ expected number of rounds for the conflicts to be at a distance less than 2. $\square$

2.2 An optimal protocol for rings

In this section we present an optimal protocol, RingElect, for ring networks: it uses constant memory and converges in $\Theta(n)$ expected number

```
Receive messages from both the neighbors and update variables $\texttt{msg}_i$
If $\texttt{status} = \textit{start}$ then
  For both $i \in \{l, r\}$ do $\textit{Election}(i, \star)$
  $\texttt{status} := \textit{leader}$
Elsif $\texttt{status} = \textit{leader}$ then
  For both $i \in \{l, r\}$ do
    If $\texttt{msg}_i$ is received in the current round then
      Set $\texttt{losing}_i$ to the value of the predicate:
      ($\texttt{msg}_i.\texttt{type} = \textit{contest}$ and $\texttt{msg}_i.\texttt{b} = 1$ and $\texttt{msg}_i.\texttt{h} = 0$ and $\texttt{b}_i = 0$)
  If $\texttt{losing}_i = \textit{true}$ for both $i \in \{l, r\}$ then
    Send concede messages to both neighbors, and $\texttt{status} := \textit{slave}$
  Else for both $i \in \{l, r\}$ do
    If $\texttt{msg}_i$ is received in the current round then
      If $\texttt{msg}_i.\texttt{type} = \textit{contest}$ then $\textit{Election}(i, \texttt{msg}_i.\texttt{b})$ else $\textit{Election}(i, \star)$
Elsif $\texttt{status} = \textit{slave}$ then
  Forward message $\texttt{msg}_l$ to neighbor $r$ unless $\texttt{msg}_l.\texttt{type} = \textit{contest}$
  and $r$ sent a concede message in the current or in the previous round
  Forward message $\texttt{msg}_r$ to neighbor $l$ unless $\texttt{msg}_r.\texttt{type} = \textit{contest}$
  and $l$ sent a concede message in the current or in the previous round
```

```
$\textit{Election}(i, h)$:
  $\texttt{b}_i :=$ random bit
  Send a contest message $\texttt{msg}$ with $\texttt{msg.b} := \texttt{b}_i$
  and with $\texttt{msg.h} = h$ to neighbor $i$
```

Fig. 1. RingElect protocol.

of rounds. The protocol elects a leader from which a 2-coloring of the ring follows. In its current form, RingElect is not self-stabilizing. We postpone the discussion on stabilizing RingElect to the final paper.

We describe RingElect in a more restrictive model, where local orientation is assumed. Specifically, a node is capable of sending a message only to a specific neighbor, and on reception of a message can distinguish which of its neighbors sent that message. This assumption is relaxed later.

The intuition behind the protocol RingElect is the following. We begin in a configuration where all nodes are leaders. Each leader plays two local leader elections on the segments of the ring connecting it to its clockwise-next and counterclockwise-next leaders. A leader losing in both its segments becomes slave and sends a message notifying that it is conceding the election to the leaders at the end of its segments. A concede message has the extra function of aborting all the election messages encountered whilst traversing the segment. A detailed description of the protocol follows. As mentioned before, we consider two types of messages, *concede* and *contest*. Each message `msg` has a field `msg.type` to denote its type. Concede messages are sent by nodes who became slaves. Contest messages are generated by leaders (a) during the first round, (b) on reception of a concede message, and (c) on reception of a contest message. A contest message `msg` carries a bit `msg.b` indicating the posi-

tion of the leader who sent it and a "history" $\texttt{msg.h} \in \{\star, 0, 1\}$ indicating what generated it: in cases (a) and (b) the history is set to $\star$; in case (c) the history is set to the election bit contained in the received contest message.
Each node has the following local variables: $\texttt{status} \in \{\textit{start,leader,slave}\}$ (initialized with *start*); for $i \in \{l, r\}$, $\texttt{msg}_i$ to remember the latest message received from direction i; $\texttt{b}_i$ to store the random bit used for the election on the segment in direction i; $\texttt{losing}_i$ to remember whether it is "losing" the election on the segment in direction i. Each node runs the protocol in figure 1 in each round.
A portion of the ring between nodes u and v form a *segment* at round t if at round t, u and v are leaders and all other nodes between u and v are slaves. At the beginning we have exactly n segments. As the protocol progresses, the number of segments goes down and the segments get larger. At the point when there is only one leader, all segments vanish.

Analysis of RingElect

The analysis of the protocol is not trivial. We proceed as follows: first, we observe some properties of the protocol and give useful definitions; then we give a careful characterization of the possible configurations of any segment at any round. The properties of these configurations will imply that the protocol is consistent, that is, there is always at least a leader at any round. Finally, we prove that the convergence of the protocol is optimal.
We begin by observing the following properties of the protocol RingElect. During any round if a node becomes a slave, it will remain a slave for all subsequent rounds. For every node and for all $i \in \{l, r\}$, $\texttt{b}_i$ is the same as the bit in the most recent contest message sent in the direction i. Also u receives contest messages only from v from the direction of v, and v only from u from the direction of u for the entire duration u and v are the leaders of a segment. When one of the two nodes in a segment becomes a slave a concede message will be sent by the slave toward the other leader of the segment. Any contest message received by a slave node which has received a concede message in the current or previous round going in the opposite direction will not be forwarded further.
We say that a message $\texttt{msg}$ is *crucial* if $\texttt{msg.type} = \textit{contest}$, $\texttt{msg.b} = 1$ and $\texttt{msg.h} = 0$. Similarly, a message variable $\texttt{msg}_i$ of a leader node is *crucial* at the end of round t if it holds a crucial message at the end of round t. For a segment with left-leader u and right-leader v, the variables of the segment are the variable $\texttt{msg}_r$ of u and the variable $\texttt{msg}_l$ of v.
Henceforth, we will say that a message m is on a segment at the end of round t, if during round t a node u of that segment sent m to another node v of that segment. The direction of a message is defined by who sent the message and who received it. We say that two messages m from u to v and m' from u' to v' on a segment are *converging* (resp. *diverging*) if a path $u, v, \ldots, v', u'$ (resp. $v, u, \ldots, u', v'$) is in the segment. Finally, for a segment of leaders u and v, and for a message m on the segment directed toward u, any message in between m and v is said to be *behind* m.

Definition 1. *A segment is in a safe configuration at the end of round t if the following properties hold at the end of round t.*

(i) There are one or two concede messages on the segment.

(ii) No variable of the segment is crucial.

(iii) Every crucial contest message on the segment is converging to a concede message on the segment.

(iv) If there are two concede messages on the segment then they are diverging and no other message is in between them. If there is only one concede message on the segment, then there can be only one message behind it. This message is non-crucial and is traveling in the same direction as the concede message.

Definition 2. *A segment is in a contest configuration at the end of round t if the following properties hold at the end of round t.*

(a) There are exactly two contest messages and no concede messages on the segment.

(b) At most one variable of the segment is crucial.

(c) Crucial messages on the segment travel in the same direction. Also if a variable of the segment is crucial, crucial messages travel toward the leader holding that variable.

Lemma 3. *At the end of any round $t \geq 1$, any segment is either in a safe configuration or in a contest configuration.*

Proof. We prove the lemma by induction on t. During round $t = 1$ every node sends a non-crucial contest message per segment, therefore at the end of the round there will be two non-crucial contest messages (history of messages is set to $\star$) per segment (also no leader has crucial variables). Therefore, at the end of round $t = 1$ each segment is in a contest configuration.
By induction suppose that the lemma holds at the end of round t, and consider any segment. First suppose that the segment is in a safe configuration at the end of round t. If no concede messages are received by any of the two leaders at the beginning of round $t+1$, it is easy to check that there will be a safe configuration at the end of round $t+1$. Otherwise a leader receiving a concede message at the beginning of round $t+1$, will send a non-crucial contest message on the segment (the history of the message is set to $\star$). Therefore at the end of round $t+1$ the segment will be either in a safe configuration (if at least a concede message is present) or in a contest configuration (if no concede messages are left).
Now suppose that the segment is in a contest configuration at the end of round t. Consider first the case when no leader of the segment becomes a slave during round $t+1$. We want to prove that at the end of round $t+1$ there will be a contest configuration. Note that property (a) holds at the end of round $t+1$ because a slave receiving a contest message will forward it, while a leader receiving a contest message will send another contest message on the segment (recall that we are assuming that leaders do not become slaves during round $t+1$). Property (b) holds at the end of round $t+1$ because property (c) guarantees that only one leader can receive a crucial message at the beginning of round $t+1$ (and always by (c) the other leader cannot have a crucial variable). Property (c) holds

at the end of round $t+1$ because a leader that receives a crucial message at the beginning of round $t+1$, will send a non-crucial contest message on the segment (the history of the message will be 1 since the received message is crucial). Now consider the case when a leader of the segment becomes a slave during round $t+1$. A leader becomes a slave only if both its variables are crucial. Therefore property (b) implies that only one leader of the segment can become a slave during round $t+1$. Let u be this node, and let v and w be the other leaders of the two segments of u. Note that since u becomes a slave during round $t+1$, it must be the case that both segments of u are in a contest configuration at the beginning of round $t+1$ (if not, one of u's variable would not be crucial). We want to prove that at the end of round $t+1$ the new segment defined by the leaders v and w will be in a safe configuration. Property (i) and (iv) are trivial since u will send two concede messages on its two sides. By property (b) both v and w have non-crucial variables at the end of round t, and since all crucial contest messages are traveling toward u by (c), properties (ii) and (iii) will hold at the end of round $t+1$. □

From Lemma 3, property (ii) of safe configurations and properties (b)-(c) of contest configurations, it follows that there is always at least one leader.

Corollary 1. *At any round t there exists at least one leader.*

We can now prove that the protocol converges in optimal time.

Theorem 2. *Protocol* RingElect *elects a leader and computes a* 2*-coloring in $O(n)$ expected number of rounds and $O(n \log n)$ bit-complexity.*

Proof. Fix a round t. For a leader u, let the *scope* of u be the union of the two segments of u (notice that scopes are not disjoint and each scope contains exactly 3 leaders.) Let $l^{\langle t\rangle}$ be the numbers of leaders at round t. Let T be a maximal set of node-disjoint scopes. Then it has to be $|T| \geq \lfloor \frac{l^{\langle t\rangle}}{3} \rfloor$. We observe that at least $(\lfloor \frac{T}{2} \rfloor - 1)$ scopes of T have length at most $\frac{6n}{l^{\langle t\rangle}}$ (otherwise the remaining scopes of T would contain at least $(\frac{|T|}{2}+1)\frac{6n}{l^{\langle t\rangle}} > n$ distinct nodes.) Consider any of these "short" scopes. Observe that at most every $\frac{12n}{l^{\langle t\rangle}}$ rounds either at least one of the three leaders has become a slave or all the nodes have drawn new bits for the elections on the two segments of the scope. In each of these phases there is a constant probability that the central leader of the scope becomes a slave (note that the two segments are a in contest configuration assuming the two non-central leaders do not become slaves). Therefore, we can conclude that in expectation after $O(\frac{n}{l^{\langle t\rangle}})$ rounds all the short scopes will have lost at least one leader. Thus, in expectation $l^{\langle t+O(n/l^{\langle t\rangle})\rangle} \leq l^{\langle t\rangle} - (\frac{|T|}{2} - 1) \leq \frac{l^{\langle t\rangle}}{c}$ for some constant $c > 1$. That is, the number $l^{\langle t\rangle}$ of leaders at some round t decreases by a constant factor of c after a phase of $c'\frac{n}{l^{\langle t\rangle}}$ rounds (for some constant c'). Note that given $c' \geq 12$ we can assume that the election bits in a phase are independent from the bits of the previous phase because the leaders we consider are playing in short scopes, and therefore draw new bits every $\frac{12n}{l^{\langle t\rangle}}$ rounds. By iterating this argument, the expected number of rounds in order to reduce the

number of leaders from l to 1 is proportional to $\frac{n}{l}\sum_{i=0}^{\log_c(l)} c^i = O(n)$. With a similar analysis it is possible to show that the expected number of messages is $O(n \log l) = O(n \log n)$. The existence of at least one leader is established in Corollary 1.
We will now explain how a 2-coloring can be achieved. In the first round every node chooses a color for itself. Every time that a leader receives a concede message, it will start to propagate its coloring. A slave who receives two non-compatible colorings by its two neighbors will not propagate any of the two. When only one leader is left, the coloring of this leader will be propagated through the entire network (in linear number of rounds). $\square$

Relaxation of the model. We will briefly describe how to modify the protocol RingElect so that only broadcast is used. We will still assume that when a node receives a message, it can distinguish which neighbor broadcast it[7]. The key property that will use is the fact that a slave never receives two messages coming from the same neighbor in two consecutive rounds. (This property can be shown to hold inductively.) Using this property we can modify the protocol as follows. A slave node u will accept (and therefore broadcast) a message m broadcast from a slave neighbor v iff at least one of these conditions holds: (i) in the previous round u did not accept m from the other neighbor; (ii) v is broadcasting two messages (this happens only if in the previous round v accepted the message m from u and a message m' from its other neighbor; in this case, if $m \neq m'$, u knows what message to ignore, otherwise u will accept any of the two). A slave node u will accept a message m broadcast from a leader neighbor w iff in the previous round u broadcast a message received by the other neighbor. Similar rules can be used for leader nodes. The only major modification is the following: when a leader accepts two messages in the same round (coming from the two different segments) and does not become a slave, it will draw only one bit and use (broadcast) it for both segments. This modification of the election process does not affect the performance of the protocol which will still converge in linear time.

Asymmetric incentives. As mentioned in the introduction, the experiments in [15] introduce asymmetric incentives: if a proper coloring is achieved, the participants ending with a specific color are paid more than the participants ending with the other color. Longer convergence time has been observed in this setting.
We wish to quantify the influence of these asymmetric incentives on our protocols. We model "selfish" participants in the following way. We say that a node is Byzantine if, when supposed to give up the special color or become a slave, it does so with probability q strictly less than one. Now consider a ring with (at least) two Byzantine nodes at odd distance $\Omega(n)$. (Note that if we place two Byzantine nodes at random on the ring, this situation will happen with constant probability.) Then, with proofs

[7] Note that this is a natural assumption. For example, this assumption holds for the coloring experiments in [11, 15, 8].

similar to the ones presented, it is possible to show that the convergence time of the protocols gets slower in the following way: RingGuess will converge in $\Theta(\frac{n^2}{q^2} + \frac{1}{q^3})$ time; and RingElect will converge in time $O(\frac{n}{q})$. An interesting aspect caused by the requirement on constant memory, is that detection of Byzantine nodes is impossible for the other nodes.

3 General bipartite graphs

We now turn our attention to general bipartite graphs. First we present a simple protocol that computes a 2-coloring of any bipartite graph in poly(n) time. Each node v uses an amount of memory proportional to the logarithm of its degree. Secondly, we show that using a little more memory per node, namely $O(\log n)$ bits, we develop a protocol whose convergence time is essentially optimal.

3.1 A coalescing particles protocol

Without loss of generality let the color palette be $\{0, 1\}$. Consider the following simple protocol that we call GraphCoalescing.
In the first round every node chooses a random color b and sends a "suggestion" $\bar{b}$, the complement of b, to a random neighbor. In every round $t \geq 2$, each node u receiving at least one suggestion:
(a) randomly chooses a suggestion b among the received ones;
(b) colors itself with b; and
(c) randomly chooses a node w in the set composed of its neighbors and itself; if w is a neighbor sends suggestion $\bar{b}$ to w, otherwise re-suggests b to itself for the next round.
Observe that each node u uses $O(\log \delta_u)$ bits of memory to select a random neighbor.
The idea of this protocol is that every node proposes a coloring. Each proposal takes a random walk on the graph, and when two (or more) proposals meet, only one of them will survive (the proposals will *coalesce* into one) and continue its random walk. Now suppose that at some point only one proposal is left: then it will walk randomly through the network, and will lead to a proper coloring once all nodes have been visited. Viewing proposals as tokens, it follows that the protocol can also be used to provide a token management scheme (See [10] for a similar approach). The following theorem borrows heavily from the literature.

Theorem 3. *Protocol* GraphCoalescing *2-colors any bipartite graph with m edges in $O(m^2 n \log n)$ expected number of rounds. If the graph is Δ-regular the expected number of rounds is $O(\Delta n^2)$.*

Proof. In the proof we refer to each proposal as a particle. Let G be a (bipartite) graph. We observe that part (c) of the protocol implies that each particle is performing a random walk on the graph G' that is obtained from G by adding self-loops to each node. Therefore, since G' is not bipartite, the random walk of each particle is aperiodic. The expected number of rounds required for coloring the graph is bounded by the expected number T_{coalesce} of rounds for the particles to coalesce to a single particle, plus the cover time T_{cover} (that is, the expected number of rounds for a single particle to visit all the nodes). A classic result in [1] shows that the cover time of a graph is $T_{\text{cover}} = O(mn)$. By [2, Section 14.3], we have that $T_{\text{coalesce}} = O(\Delta n^2)$ for Δ-regular graphs, and $T_{\text{coalesce}} = O(T_{\text{cat\&mouse}} \log n)$ for general graphs, where $T_{\text{cat\&mouse}}$ is the time required for two random walks to meet. For non-bipartite graphs it is well-known that $T_{\text{cat\&mouse}} = O(m^2 n)$. The theorem follows. □

We observe that the protocol (as it is) is not suitable for a broadcast model because nodes must be able to send messages to a specific neighbor. We now argue that this issue can be addressed if we observe the isomorphism between the coalescing particles process and the voter model. In the voter model each node starts with an opinion (a proposal in our case). As time passes, nodes modify their opinions in the following way. At each step, each node changes its opinion to the opinion of a random neighbor or stick to its opinion where all the options are equally probable. It is known that the expected time for only one opinion to survive (the number of opinions can only decrease with time) is the same as the expected time for all the particles to coalesce (e.g. see [7]). This observation easily leads to a broadcast protocol with the same guarantees.

3.2 A time-optimal protocol

In this section we present GraphElect, a protocol that uses $O(\log n)$ memory in expectation and computes a 2-coloring of any bipartite graph in $O(D + \log n)$ expected number of rounds, where n and D are size and diameter of the graph respectively. Any distributed protocol that 2-colors general bipartite graphs requires $\Omega(D)$ rounds: therefore GraphElect is time-optimal in graphs of diameter at least $\Omega(\log n)$.

We now describe the protocol GraphElect. At any given stage, a processor can be either a leader or a slave. At the beginning of the protocol all processors are leaders. Each processor presents to its neighbors a variable `Leading-Rank` (initial value 0), and its color (initial value either 0 or 1). In addition, each processor keeps locally a variable `Rank` (initial value 0). At the beginning of each round, a processor reads the state variables of all its neighbors and computes the maximal leading rank among all its neighbors. The processor holding that maximal leading rank is the processor's leading neighbor. If there is more than a single processor holding the maximal leading rank, the leading neighbor is elected arbitrarily from among the leading processors. If the maximal leading rank is larger than the processor's own rank, the processor adjusts its color to be the opposite color of its leading neighbor, and becomes a slave (if it was a leader). A slave keeps doing this simple routine forever and never gets a chance to become a leader again.

In addition to all aforementioned variables, a leader also holds a timer whose initial value is zero. The nodes counts down from timer value to zero. When the count goes to 0, if the processor is still a leader, it increments its rank and leading rank by 1. Then it updates its timer value to be twice the old timer value, plus a random value in $\{0, 1\}$.

The following lemma is pivotal for the analysis of the protocol.

Lemma 4. *Let u and v two nodes in the graph. If, at the beginning of a certain round, u is still a leader and its rank is greater than the rank of v, then for the rest of the computation there will be some node (possibly u) with rank greater than the rank of v.*

Proof. Let $W_k^{\langle u \rangle}$ be the value of the timer of u right after the k-th update. $W_k^{\langle u \rangle} = 2 \cdot W_{k-1}^{\langle u \rangle} + B_k^{\langle u \rangle}$ (as long as u is a leader), where the $B_k^{\langle u \rangle}$'s are i.i.d. random variables taking values from $\{0, 1\}$. We observe that $W_k^{\langle u \rangle} = \sum_{i=0}^{k} 2^{k-i} B_k^{\langle u \rangle}$. Now consider the first round t^* when the rank of u is greater than the rank of v. During round t^*, u must have updated its timer, and let this one be its k-th update. Note that it must be the case that $W_j^{\langle u \rangle} = W_j^{\langle v \rangle}$ (and therefore $B_j^{\langle u \rangle} = B_j^{\langle v \rangle}$) for all $j < k - 1$, and $W_{k-1}^{\langle u \rangle} < W_{k-1}^{\langle v \rangle}$ (and therefore $B_{k-1}^{\langle u \rangle} < B_{k-1}^{\langle v \rangle}$). Now consider the k-th update for u and v. We have that $W_k^{\langle u \rangle} < W_k^{\langle v \rangle}$ since

$$2W_{k-1}^{\langle u \rangle} + B_k^{\langle u \rangle} \leq 2W_{k-1}^{\langle u \rangle} + 1 \leq 2(W_{k-1}^{\langle v \rangle} - 1) + 1 < 2W_{k-1}^{\langle v \rangle} \leq 2W_{k-1}^{\langle v \rangle} + B_k^{\langle v \rangle}.$$

Therefore, $W_k^{\langle u \rangle}$ rounds after t^*, u will increase its rank to $k + 1$, while the rank of v can only increase it to $k+1$ after at least $1 + W_k^{\langle v \rangle} > W_k^{\langle u \rangle}$ many rounds from t^*.
By induction, as long as u is a leader, u will have a rank greater than the rank of v. If at some point u loses its leadership status, it must be that another node u' has a rank greater than the rank of u, and thus greater than the rank of v. □

The main theorem of this section is the following.

Theorem 4. *Protocol* GraphElect *elects a leader and computes a* 2*-coloring of the graph in* $O(D + \log n)$ *expected number of rounds, where* D *is the diameter of the graph.*

Proof. We will compute the expected time to have only one leader remaining in the network. Once this event happens, the coloring propagated by the leader will be adapted by each node eventually thus producing a 2-coloring. Let $t(k)$ be the minimum round such that there exists a node that is updating its timer for the k-th time. In other words $t(k)$ is the round during which one or more nodes achieve the rank k for the first time and k is the largest rank at round $t(k)$. Also let L_k be the set of nodes with the largest rank k at round $t(k)$. By Lemma 4, we have that $L_{k+1} \subseteq L_k$ for any k. We want to compute the expected k^* such that $|L_{k^*}| = 1$. At the beginning L_0 contains all the nodes in the graph, so $|L_0| = n$. At round $t(k)$ we have L_k nodes which will select i.i.d. random numbers in $\{0, 1\}$: we observe that only the nodes that select 0 will be in L_{k+1}. Therefore, in expectation, $|L_{k+1}| = \frac{1}{2}|L_k|$. We conclude that in $O(\log n)$ expected rounds there will be only one node with the largest rank.
At this point we have only one node of largest rank, call it u^*. However we can still have multiple leaders: for example u^* might be very far from some other leader w, and by the time the leading rank is propagated from u^* to w, w might have increased its own rank. Note that this cannot happen when the timer length of u^* (and therefore of all the other

leaders) is at least D. Since, after $O(\log n)$ rounds the u^*'s timer value will be more than 0 with high probability, and the timer value doubles at each update, we have that after at most $O(\log n + D)$ rounds from round $t(k^*)$ the u^*'s timer value will be at least D. Thus, after $O(\log n + D)$ expected number of rounds there will be only one leader. $\square$

References

1. R. Aleliunas, R.M. Karp, R.J. Lipton, L. Lovász and C. Rackoff. Random Walks, Universal Traversal Sequences, and the Complexity of Maze Problems. *FOCS'79*, 218-223.
2. D. Aldous and J. Fill. *Reversible Markov Chains and Random Walks on Graphs.* http://www.stat.berkeley.edu/~aldous/RWG/book.html.
3. H. Attiya, and J. Welch. *Distributed Computing; Fundamentals, Simulations and Advanced Topics*, second ed., John Wiley&Sons, 2004.
4. A. L. Barabási, R. Albert. Emergence of Scaling in Random Networks, *Science* 1999, Vol. 286. no. 5439, pp. 509-512.
5. B. Bollobás, O. Riordan, J. Spencer, G. Tusnády. The degree sequence of a scale-free random graph process. *Random Structures & Algorithms*, Volume 18, Issue 3, May 2001.
6. K. Chaudhuri, F. Chung Graham, M. S. Jamall. A network coloring game. *WINE'08*, pp. 522–530, 2008.
7. C. Cooper, A. Frieze, and T. Radzik. Multiple random walks in random regular graphs. *SIAM Journal on Discrete Mathematics*, 23(4):1738-1761, 2009.
8. D. Enemark, M. McCubbins, R. Paturi, and N. Weller. Good edge, bad edge: How network structure affects a group's ability to coordinate. *ESORICS*, March 2009.
9. M. R. Garey and D.S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman, 1979.
10. A. Israeli, M. Jalfon. Token Management Schemes and Random Walks Yield Self-Stabilizing Mutual Exclusion. *PODC'90*, 119-131.
11. M. Kearns, S. Suri, and N. Montfort. An experimental study of the coloring problem on human subject networks. *Science*, 313(5788):824–827, August 2006.
12. M. Kearns, S. Judd, J. Tan, and J. Wortman. Behavioral experiments on biased voting in networks. *National Academy of Science*, Jan.'09.
13. S. Khot. Improved inapproximability results for maxclique, chromatic number and approximate graph coloring. *FOCS'01*, 600-609.
14. 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):12181230, 1996.
15. M. D. McCubbins, R. Paturi, and N. Weller. Connected Coordination: Network Structure and Group Coordination. *American Politics Research*, 37:899-920, 2009.
16. E. Mossel and G. Schoenebeck. Reaching Consensus on Social Networks. In *Innovations in Computer Science (ICS)*, 2009.
17. D. Peleg. *Distributed Computing: A Locally-Sensitive Approach.* SIAM Monographs, Philadelphia, 2000.
18. N. Santoro. *Design and Analysis of Distributed Algorithms.* John Wiley & Sons, Inc., 2007.