Two Edge Coloring Algorithms Using a Simple Matching Discovery Automata

J. Paul Daigle and Sushil K. Prasad
Department of Computer Science
Georgia State University
Atlanta, Georgia 30303, USA

***Abstract*—We here present two probabilistic edge coloring algorithms for a message passing model of distributed computing. The algorithms use a simple automata for finding a matching on a graph to produce the colorings. Our first algorithm for edge coloring finds an edge coloring of a graph which is guaranteed to use no more than $2\Delta - 1$ colors and completes in $O(\Delta)$ communication rounds using only one hop information, where Δ is the greatest degree of the graph. Our second algorithm finds a strong edge coloring of a symmetric digraph in $O(\Delta)$ communication rounds, using only one hop information.**

I. Introduction

In this paper, we use the matching based automata used in [3] and shown in Figure 1 as the basis for two different edge-coloring algorithms, edge coloring of an undirected graph and strong edge coloring of a directed graph. A practical distributed algorithm for the last problem is of some interest in networking, as it can be used as a model for channel or time-slot assignment in an adhoc network [2], [4].

Our algorithms are probabilistic and compete well with other probabilistic algorithms. For the edge coloring problem, our algorithm produces a coloring which is no greater than $2\Delta - 1$ in $O(\Delta)$ rounds in the typical case, where Δ is the maximum degree of the graph. Our algorithm for strong directed coloring produces a correct coloring in $O(\Delta)$ rounds.

Our algorithms assume a message based model of computing, so we can assume that compute nodes are synchronized and that each node can communicate with each of its neighbors in each round [8]. Each node is therefore assumed to move synchronously through each stage of the algorithm.

Our main contribution is extending the framework developed in [3] to a new set of problems. The algorithms based on the framework are competitive with know algorithms in time complexity and provide high quality solutions. We believe that this basic approach can be modified and extended to solve a variety of problems, providing a simple starting point for the development of new distributed, probabilistic algorithms that provide constant approximations for NP complete problems.

A. Problem Definitions

Definition 1 (Edge Coloring)**.** An edge coloring of a graph is an assignment of colors to the edges of a graph in such a way that no two adjacent edges are assigned the same color.

Formally, given a graph, $G(V, E)$, and set of colors C, an edge coloring of G is a mapping $f(e) = E \mapsto C \mid \forall\, e(u,v), e'(v,w) \in E, c \in C, f(e) = c \implies f(e') \neq c$.

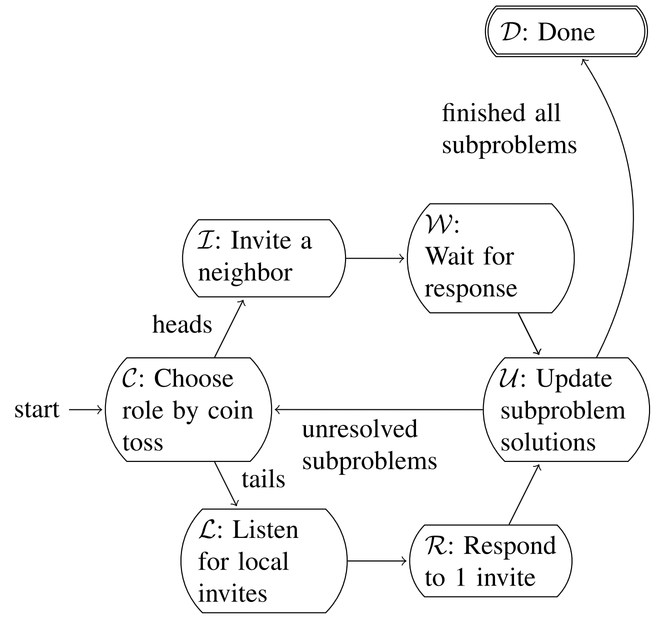

$\mathcal{D}$: Done
finished all subproblems
$\mathcal{I}$: Invite a neighbor
$\mathcal{W}$: Wait for response
heads
start
$\mathcal{C}$: Choose role by coin toss
unresolved subproblems
$\mathcal{U}$: Update subproblem solutions
tails
$\mathcal{L}$: Listen for local invites
$\mathcal{R}$: Respond to 1 invite

Fig. 1: Distributed Matching and Computation Automata

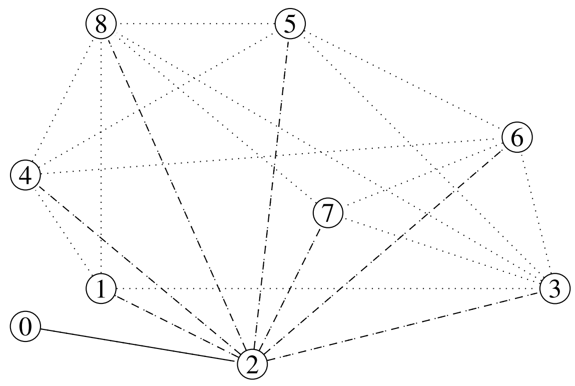
Definition 2 (Strong Directed Edge Coloring)**.** A strong edge coloring is an assignment of colors to the edges of the graph such that no two edges that can be connected by a common edge are assigned the same color.

In the directed case, a strong edge coloring is a mapping $f(e) = C \mapsto E \mid \forall\, e(u,v), e'(v,u), e''(w,v), e'''(w,x) \in E, c \in C, f(e) = c \implies f(e') \neq c, f(e'') \neq c, f(e''') \neq c$.

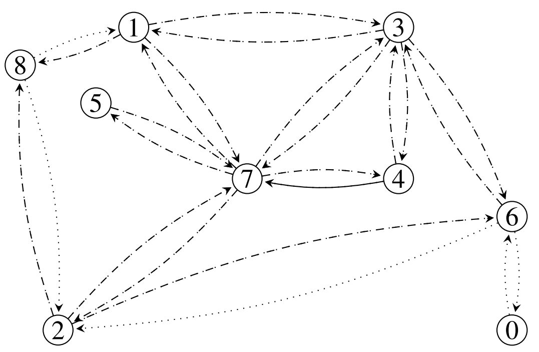
Figure 2 shows the influence of a color assignment to an edge of a graph on the possible color assignments to the rest of the graph–none of the dashed edges can be colored with the same color as the solid edge, but at least one of the dotted edges could be colored with the same color as the solid edge.

B. Prior Work

Distributed edge coloring is a well studied problem. Panconesi and collaborators have produced a number of papers tying edge coloring to channel assignment and presenting novel edge coloring algorithms with communication complexity of as low as $O(loglogn)$ [5],[11],[10],[9].



(a) Edge Coloring: Influence of (0,2)



(b) Strong Directed Edge Coloring: Influence of (4,7)

Fig. 2: Edge Colorings: assigning a color to the solid edge makes that color unavailable to the dashed edges, but not to the dotted edges

Gandham et al. present an deterministic algorithm which colors a graph using $\Delta + 1$ colors with a time complexity of $2\Delta + 1$ for acyclic graphs [4]. Barenboim et al. present a deterministic algorithm which extends beyond trees and provides an $O(\Delta)$ coloring in $O(\Delta^{1+\epsilon})$ time for an arbitrarily small constant $\epsilon \leq 1$ [1]. A limitation of this algorithm is that the constant factor for quality increases as ϵ decreases.

In strong edge coloring, Barret et al. present algorithms with running times dependent on the size of the graph n [2]. Kanj et al. show tight bounds for the quality and locality, but not the time bounds, of their algorithms [7].

C. The Message Passing Model

Our algorithms assume a message passing model of distributed computing. Therefore, we make the following two assumptions. First, communication rounds proceed synchronously. Second, each node can communicate with each of its neighbors once during any communication round. Structurally, our algorithms map each vertex of the graph to a compute node of the distributed computer.

We also use the term 'round' in two different senses. A computation round in our algorithm is composed of several communication rounds. Referring to the automata in Figure 1, the automata defines the possible states of a node during a single computation round. During that computation round, there are two distinct communication rounds where nodes in the $\mathcal{I}$ state send invitations, while nodes which are (synchronously) in the $\mathcal{L}$ state listen for invitations, and a second communication round in which nodes in the $\mathcal{R}$ state respond to any invitations which they have received while nodes in the $\mathcal{W}$ state wait to receive responses.

A key point is that nodes are assumed to move synchronously through the stages of the automata regardless of whether they receive invitations or responses, although they can only perform computations based on the messages which they receive.

The purpose of the our framework is to generate a matching[1] on the graph in each computation round. For a number of graph problems, including edge coloring and vertex cover, finding such a matching allows for local computations to be made simultaneously by the nodes in the matching without the possibility of conflict.

II. Edge Color Algorithm

A. Algorithm

In Figure 1, the states of the vertexes are labeled $\mathcal{C}$, $\mathcal{I}$, $\mathcal{L}$, $\mathcal{W}$, $\mathcal{R}$, $\mathcal{U}$, and $\mathcal{D}$. In our adaptation of the automata, shown in Algorithm 1, an additional state, $\mathcal{E}$, is used for the purposes of exchanging information at the end of each round.

Each node maintains a list of colors that each of its neighbors has used. This is updated at the end of each round.

All transitions are made synchronously, that is, we assume that if any node in the network is in a given state, all nodes in the network are either in that state or in a corresponding state. States that correspond are $\mathcal{C}$, $\mathcal{D}$, $\mathcal{I}$, $\mathcal{L}$, $\mathcal{R}$, $\mathcal{W}$.

1) $\mathcal{C}$ (Choose) state: Each vertex chooses with equal probability to transition to the $\mathcal{I}$ state or the $\mathcal{L}$ state.

2) $\mathcal{I}$ (Invite) State: Each node in the $\mathcal{I}$ state chooses an available edge and available color. The edge is chosen randomly from among the nodes uncolored edges, and the color is the lowest indexed color available to color that edge (Algorithm 1, lines 1.11,1.36). The node then sends a message containing its own id, the id of the intended receiver, and the proposed color.

3) $\mathcal{L}$ (Listen) State: Each Node in the $\mathcal{L}$ state listens for invitations from its neighbors. From among the invitations received, the node will keep invitations containing its own id and transition to the $\mathcal{R}$ state.

[1]A matching for a graph $G(V, E)$ is a subset $E' \subset E \mid \forall\, e(u, v), e'(u, w), e''(v, x) \in E, e \in E' \implies e' \notin E', e'' \notin E'$. Less formally, no two contiguous edges are in the matching.

4) $\mathcal{R}$ (Respond) State: Each node in the $\mathcal{R}$ state chooses a random invitation from the invitations recieved and kept in the $\mathcal{L}$ state. The node sends a reply message containing its own id, the id of the original sender, and the proposed color from the invitation. This response message is a duplicate of the invitation message with the ids reversed. The node then transitions to the $\mathcal{U}$ state. Nodes that recieved no invitations simply transition to the $\mathcal{U}$ state.

5) $\mathcal{W}$ (Wait) State: Nodes in the $\mathcal{W}$ state collect all replies in their neighborhood and transition to the $\mathcal{U}$ state.

6) $\mathcal{U}$ (Update) State: Nodes in the $\mathcal{U}$ state update their own edges if they have received a response to an invitation or responded to an invitation. All nodes next transition to the $\mathcal{E}$ state.

7) $\mathcal{E}$ (Exchange) State: In this state, nodes that have used new colors broadcast those colors to their neighbors. All nodes use this information to update the colors that they have available to exchange with their neighbors. Nodes which have colored all of their edges transfer to the $\mathcal{D}$ (Done) State, and all other nodes transition back to $\mathcal{C}$.

B. Algorithm Analysis

We here address the termination, correctness, and solution quality of Algorithm 1. We will first show that Algorithm 1 is likely to terminate in $O(\Delta)$ rounds, then that it will produce a correct coloring if it does terminate, and finally that the coloring produced will use no more than $2\Delta - 1$ colors.

Proposition 1. *Algorithm 1 is likely to terminate in $O(\Delta)$ rounds.*

Discussion of Proposition 1:

Algorithm 1 terminates when all of the nodes have colored all of their edges. In order to color an edge, a node must form a pair in an a given round with each of its neighbors. The number of neighbors of each node is $\delta \leq \Delta$.

We note that there is no limit on the number of nodes that can participate in any given round. If a particular graph has a complete matching, than every compute node may participate in a round. Further, the fact that a node is participating guarantees that at least one of its neighbors is participating, but does not prevent any of its other neighbors from participating as well.

Therefore, if the probability that a given node w will participate in the computation for a given round can be shown to be constant, than the number of compute rounds is bounded by that constant times Δ.

We therefore will procede to find the probability that a random node w will color one of its edges in some round r.

In any given round w may choose to be an invitor or an invitee with equal probability. The probability that w will be an invitee in r is $1/2$.

Each neighbor of w will choose to be an invitor or an invitee with equal probability as well. In the average round, if the number of uncolored edges of w is δ, than the number of invitors incident to w will be $\delta/2$.

Algorithm 1 Distributed Matching Based Edge-Coloring Algorithm

```
for all $v_u \in V$ in parrallel do
    $live_u \leftarrow C$                ▷ All colors are available
    $dead_u \leftarrow \{\}$             ▷ No colors are used
    $used_u \leftarrow [\,]$             ▷ No colors are assigned
    state $\leftarrow \mathcal{C}$
    repeat
        if state = $\mathcal{C}$ then
            State $\leftarrow (\mathcal{I} \vee \mathcal{L})$    ▷ Coin toss selects state
        else if state = $\mathcal{I}$ then
            Randomly select an uncolored edge, $e_{u,v}$
            $c \leftarrow (live_u \setminus used_v[1]$  ▷ assign first available color to $c$
            Broadcast $I_u^v, c$         ▷ $u$ Invites $v$ to color $e_{u,v}$ with $c$
            state $\leftarrow \mathcal{W}$
        else if state = $\mathcal{L}$ then
            Recieve $I_x^y, c$           ▷ all local invites
            if $y = u$ then              ▷ invite is targeted to $v_u$
                store $I_x^y, c$
            end if
            State $\leftarrow \mathcal{R}$
        else if state = $\mathcal{R}$ then
            Randomly Select $I_v^u, c$   ▷ from stored invites
            Broadcast $R_u^v, c$         ▷ $u$ accepts $v's$ invitation
            Assign $c \mapsto e_{u,v}$
            $used_u \hookleftarrow c$    ▷ Append c to assigned colors
            state $\leftarrow \mathcal{U}$
        else if state = $\mathcal{W}$ then
            Recieve $R_x^y, c$           ▷ all local responses
            if $y = u$ then              ▷ response is to $v_u$
                Assign $c \mapsto e_{u,v}$
                $used \hookleftarrow c$
            end if
            state $\leftarrow \mathcal{U}$
        else if state = $\mathcal{U}$ then
            Broadcast $used_u$           ▷ Broadcast all assigned edge colors
            Recieve $used_v$             ▷ Receive neighbors assigned colors
            $dead_u \hookleftarrow used_v$   ▷ the "dead" set contains the used colors from each neighbor, and is used when choosing colors in the invitation step
            state $\leftarrow \mathcal{E}$
        else if State = $\mathcal{E}$ then
            Subtract $used_u$ from $live_u$   ▷ update usable colors
            state $\leftarrow \mathcal{C}$
        end if
    until All edges are assigned a color
end for
```

Each neighbor of w also has $\delta \leq \Delta$ uncolored edges. Every inviting neighbor v of w will choose an uncolored edge, independently and at random, to send an invitation to. Therefore, there is a $1/\delta$ chance that v will send an invitation to w.

Therefore, we can calculate the odds that w will be an invitee which recieves an invitation by multiplying the odds that w is an invitee with the number of neighbors of w that are invitors by the odds that a given neighbor will send an invitation to w.

$$\frac{1}{2} \times \frac{\delta}{2} \times \frac{1}{\delta} = \frac{1}{4} \tag{1}$$

If w receives an invitation from v, it will respond to that invitation and color the edge $e(v, w)$. Therefore, for any given node in the network, the odds that it will color a single edge in a given round are bounded by a constant, and therefore the algorithm is likely to terminate in $O(\Delta)$ rounds. ∎

Proposition 2. *Algorithm 1 produces a correct coloring.*

Discussion of Proposition 2:
Assume that Algorithm 1 does not produce a correct coloring, There are two cases where this could occur: either there exists a node v that uses some color twice, or there exist nodes v, w that color the edge (v, w) with different colors.

A vertex colors an edge after negotiation with some neighbor. In order for an edge to be colored, a vertex v must send an invitation to some neighbor w to color (v, w) with a specific color. Because we assume a message passing model, we assume that w recieves this invitation. If w responds to the invitation, v assigns the color and w assigns the color. In the message passing model, it is safe to assume that v recieves the response from w. In order for v to choose a different color than w for (v, w), v would have to either color the edge without a response, which is contrary to the behavior of the vertex (line 1.26), or v must not receive the message, which is contrary to our model.

In the second case, a vertex could use the same color twice if it either issued or responded to an invitation to use a color twice. We know, however, that whenever an algorithm uses a color, that color is assigned to a list (lines 1.24, 1.36). These colors are further removed from the list in each round (line 1.39).

If a vertex responded to or issued more than one invitation in a single round, it is possible that this conflict could occur, but this also contradicts the behavior of the algorithm of building a message containing a single id in either case.

Algorithm 1 therefore produces a correct coloring. ∎

Proposition 3. *Algorithm 1 will use $2\Delta - 1$ colors in the worst case.*

Discussion of Proposition 3:
We begin by showing the worst case performance of Algorithm 1.

In each round that a node joins a pair, both nodes use the lowest common indexed color to color the edge between them. So in the first round, color 1 will be used for every edge in the matching, in the second round, color 1 or color 2, in the third 1,2, or 3, and so forth.

So to model our worst possible case, we propose a node w with the following characteristics. First, node w has a degree of Δ and all of the neighbors of w have a degree of Δ. Second, w does not participate in the matching in the first $\Delta - 1$ rounds, but every neighbor of w does. In this way we insure that w cannot form an edge with any neighbor with an index of less than $\Delta - 1$.

In this case, w will be forced to use an additional Δ colors to connect to each neighbor, and the total number of colors used will be $2\Delta - 1$. ∎

Propositions 2, 2, and 3 imply the following conjecture.

Conjecture 1. *Algorithm 1 will produce a 2-approximate coloring in $O(\Delta)$ rounds in the typical case.*

Conjecture 2. *Algorithm 1 uses $C \leq \Delta + 1$ colors in the typical run.*

Discussion of Conjecture 2:
A graph can certainly be colored with either Δ or $\Delta + 1$ colors. If a node v were to be forced to use $\Delta + 2$ colors to color a graph, that would mean that there are two colors of index $\leq \Delta$ which are being used by each neighbor of v but not by v itself.

In order for this to happen, there would need to be some round, or sequentially set of rounds, in which all of v's neighbors formed a matching, and v did not. We know from Proposition 1 that the odds of a node forming a match in a given round are greater than $1/4$, because the odds of a node being an invitor and recieving an invitation are approximately $1/4$. We can also easily calculate that the odds of a node being an invitor and sending a successful invitation are no greater than $1/4$, since there is a $1/2$ chance that a node w will choose to send invitations and a $1/2$ chance that the neighbor w sends an invitation to is an invitee.

So the odds of a node forming a pair at all in a given round are $1/x$, $4 \geq x \geq 2$.

For a given node to not form a pair while all of its neighbors do form pairs is therefore akin to the odds that in a fair coin toss, we first flip heads and then flip tails some arbitrary number of times in a row, or that in a simultaneous coin toss of some number of coins, one is heads while the rest are tails.

We therefore expect our algorithm to behave well in the following sense: we should get conistent results with similar graphs, the algorithm should color with Δ or $\Delta + 1$ colors most of the time, and in no experimental case should we ever see the maximum $2\Delta - 1$ colors used.

∎

III. DiMa2Ed Algorithm

A. Algorithm

Algorithm 2 (DiMa2Ed) proceeds in a manner similar to Algorithm 1, adding the additional $\mathcal{E}$ step to Fig 1 and adding subroutines at different steps. All compute nodes proceed

Algorithm 2 Distributed Matching Based Distance 2 Edge Coloring for Directed Graphs (DiMa2ED)

```
1: for all $v_u \in V$ in parrallel do
2:     repeat
3:         state ← nextstate
4:         case var  $state$
5:         when true $\mathcal{C}$
6:             reset variables
7:             $nextstate \leftarrow \{\mathcal{L} \vee \mathcal{I}\}$
8:         when true $\mathcal{I}$
9:             CHOOSEROUNDPARTNER
10:            $nextstate \leftarrow \mathcal{W}$
11:        when true $\mathcal{L}$
12:            Recieve and store messages
13:            $nextstate \leftarrow \mathcal{R}$
14:        when true $\mathcal{R}$
15:            EVALUATEINVITES
16:            $nextstate \leftarrow \mathcal{U}_i$
17:        when true $\mathcal{W}$
18:            Recieve and store messages
19:            $nextstate \leftarrow \mathcal{U}_o$
20:        when true $\mathcal{U}_i, \mathcal{U}_o$
21:            UPDATEEDGES
22:            $nextstate \leftarrow \mathcal{E}$
23:        when true $\mathcal{E}$
24:            UPDATECOLORS
25:            $nextstate \leftarrow \mathcal{C}$
26:        end case
27:        SEND $nextmessage$
28:    until $\forall v$ incident to $u, e(u,v)$ is colored
29: end for
```

Procedure 2-a ChooseRoundPartner

```
1: choose a random uncolored edge $e(u,v)$
2: $roundpartner \leftarrow v$
3: Choose an open channel $\phi$ for $v$
4: $nextmessage \leftarrow \phi, v, u$
```

Procedure 2-b EvaluateInvites

```
1: for $m$ in messages collected do
2:     if $m$ includes $u$ then
3:         $mine[] \leftarrow message$   ▷ Create an array of messages sent to $u$
4:     else
5:         $other[] \leftarrow message$   ▷ Create an array of messages not sent to $u$
6:     end if
7: end for
8: $mine[] \leftarrow mine[] \mid \phi \notin other$ ▷ Look for color collisions between $mine$ and $other$
9: Select a message $m$ from $mine$
10: $nextmessage \leftarrow \phi, u, v \mid m$ contains $\phi, v, u$   ▷ reply to the chosen message
```

Procedure 2-c UpdateEdges

```
1: select a message $m \mid m$ contains $roundparter$ ▷ find the reply–if any–to the last invite sent
2: if $m$ then
3:     case var  $state$
4:     when true $\mathcal{U}_i$
5:         color edge $e(roundpartner, u)$ with $\phi$ from $m$ ▷ color the incoming edge from the round partner
6:     when true $\mathcal{U}_o$
7:         color edge $e(u, roundpartner)$ with $\phi$ from $m$ ▷ color the outgoing edge to the round partner
8:     end case
9: end if
10: $nextmessage \leftarrow edges, u$
```

synchronously. We describe the behavior of each node in each state.

1) $\mathcal{C}$ (Choose State): Each active node (nodes that have not yet assigned colors to each edge) begin each round in the $\mathcal{C}$ state. In this state, the nodes choose, with equal probability, to transition to the $\mathcal{I}$ or $\mathcal{L}$ states for the next communication step.

2) $\mathcal{I}$ (Invitor) State: If node v is in the $\mathcal{I}$ state it executes Procedure 2-a, choosing a potential round partner. Round partners are chosen at random from among the uncolored edges v, u of v, v chooses a random u and broadcasts u a message containing the id of v, the id of u, and a proposed color. v then transitions to the $\mathcal{W}$ state.

3) $\mathcal{L}$ (Listen) State: If a node v is in the $\mathcal{L}$ state it collects all messages from its neighbors and transitions to the $\mathcal{R}$ state.

4) $\mathcal{R}$ (Respond) State: A node v in the $\mathcal{R}$ State must evaluate all of its invites to look for one that it can respond to. First, invites can be divided into two categories, those which contain the id of v and those which do not. Call the former group group a and the latter group group b. v searches group a for a message which proposes a useable color that is not in any proposal from group b. v chooses a single message that meets that qualification and rebroadcasts it. v deletes all other messages from memory and transitions to $\mathcal{U}$.

5) $\mathcal{W}$ (Wait) State: In the $\mathcal{W}$ state, a node checks all messages from its neighbors looking for the message that it sent in the $\mathcal{I}$ state. If it finds such a message, it keeps it, otherwise, it deletes all messages from memory, and transitions to the $\mathcal{U}$ state.

6) $\mathcal{U}$ (Update) State: Nodes in the update state either retain 1 or 0 messages in memory. If they contain a message in memory, they color the edge described in the message with the color contained in the message, then eliminate that color from their list of usable colors. Nodes then transition to the $\mathcal{E}$ state.

7) $\mathcal{E}$ (Exchange) State: Every node exchanges the changes to their color lists with their neighbors and updates their own color lists based on what their neighbors have communicated. Nodes which have uncolored edges now transition back to the $\mathcal{C}$state.

B. Algorithm Analysis

We here address the termination and correctness of Algorithm 2.

Proposition 4. *Algorithm 2 will terminate in $O(\Delta)$ rounds in the typical case.*

The discussion of Proposition 1 applies to Proposition 4 as well.

Proposition 5. *Algorithm 2 produces a correct coloring.*

Discussion of Proposition 5: We proceed by direct contradiction. Suppose that Algorithm 2 terminates, but the coloring is not correct. There are two cases where this can happen.

a) Case 1: A node uses the same color twice.

Recall that each node colors exactly one outgoing or incoming edge at a time. In order for a single node to use one color for two edges, that node would have to either issue or accept an invite to use a color that it is already using. Since each node checks its legal color list before it issues or accepts an invite, this could only happen if some node used a color but did not update its color list. This contradicts the steps of the algorithm, in particular line 2.24.

b) Case 2: $\exists\ u, v, w, x \in V, (u, v), (v, w), (w, x) \in E \mid (u, v)$ is using color ϕ and (w, x) is using color ϕ.

For (uv) and (wx) to be colored simultaneously, node v would have to accept the invitation from node u. However, our graph is bidirectional, the existence of (vw) implies the existence of (wv). Therefore, in the round that v accepts the invitation from u, it would also collect the invitation from w to x. Since v will not accept an invitation if it detects a conflict, v will not accept the invitation from u and the edges cannot be colored simultaneously.

For (uv) and (wx) to be colored in two different rounds, either v must accept the invitation from u after (wx) is colored or w must send an invitation to x after (uv) has been colored. Either case assumes that v or w did not update their legal color lists at the end of some previous round. This contradicts the steps of the algorithm. If this is true, Proposition 5 is correct. ■

If Proposition 4 is correct and Proposition 5 is correct, Conjecture 3 is true.

Conjecture 3. *Algorithm 2 will produce a correct, distance-2 directed coloring in $O(\Delta)$ rounds where Δ is the maximum degree of the graph.*

IV. Experiments and Results

A. Algorithm 1 on Erdos-Renyi Graphs

Erdos-Renyi graphs were generated using the iGraph ruby bindings [6]. Graphs were generated with 200 or 400 nodes, and an average degree of either 4, 8, or 16. 50 graphs were generated for each size.

Consistent with Conjecture 2, $\Delta + 2$ colors were used in only 2 of the 300 runs, and in no run was the number of colors in excess of $\Delta + 2$. In the typical run, our algorithm found a coloring with either Δ or $\Delta + 1$ colors.

In keeping with our hypothesis, the number of rounds required to terminate the algorithm increased linearly with Δ, and was not affected by the number of nodes in the network. Figure 3 shows this relationship.

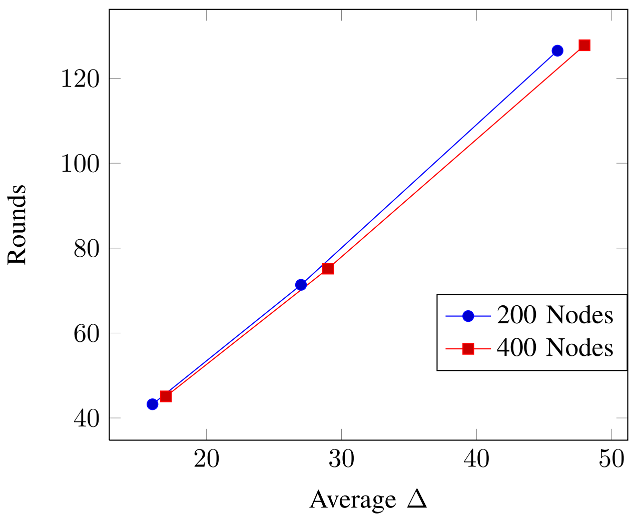


Fig. 3: Edge Coloring of Erdos-Renyi Graph

B. Algorithm 1 on Scale-Free Graphs

300 scale-free graphs were generated with either 100 or 400 nodes, with alterations in weighting to create increasingly disparate graphs. Algorithm 1 was used to generate an edge coloring for each graph.

As expected, rounds increased with Delta at what appears to be a constant rate. Interestingly, in contrast to our results on random graphs, on scale-free graphs we did not use more than Δ colors to color any of the generated graphs.

Figure 4 shows our results for Experiment IV-B.

C. Algorith 1 on Small World Graphs

300 small world graphs were generated, 100 each with 16, 64, and 256 nodes, 50 sparse and 50 dense graphs per set. Algorithm 1 was used to generate an edge coloring for each graph.

Consistent with Conjecture 1, the number of rounds required to find a coloring increased linearly with Δ and was not affected by the number of nodes in the graph. Further, the number of colors required to color the graph was less than $2\Delta - 1$ in all cases. Figure 5 shows the relationship between Δ, the number of rounds, and the number of nodes.

Conjecture 2 was not supported for this set of graphs. In particular, dense graphs with more nodes tended to use more than $\Delta + 1$ colors in most runs. The most colors used in any run was $\Delta + 5$ for a dense graph with 256 nodes. The average Δ for this group was 44.4. Further analysis of these results is required.

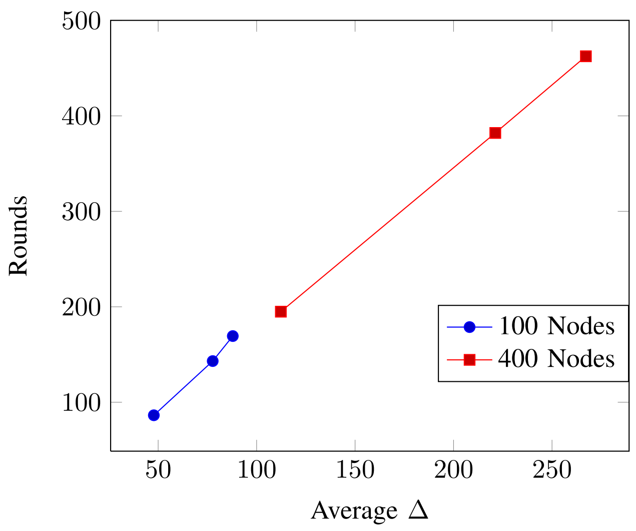


Fig. 4: Edge Coloring of Scale-Free Graphs

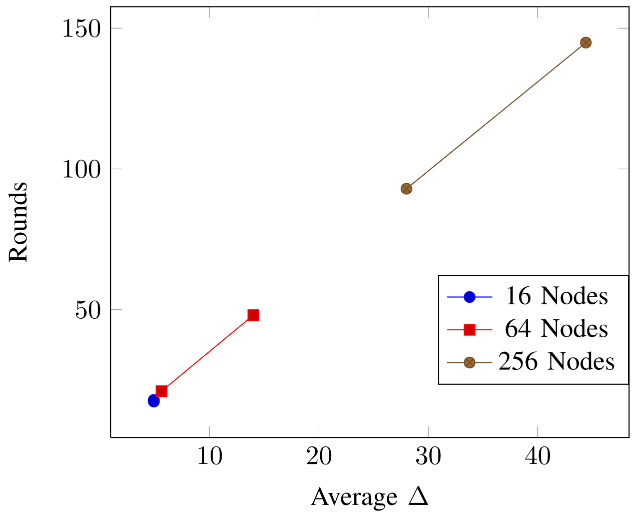


Fig. 5: Edge Coloring of Small World Graphs

D. Algorithm 2 on Erdos-Renyi Graphs

50 Erdos-Renyi graphs of 200 and 400 nodes were constructed with an average degree of 4 and 8. The averaged results showed that the graphs with 400 nodes were solved in almost identical time, with any variance easily attributable to a slightly higher average Δ. The degree of increase in the number of rounds required to create a correct distance two coloring was also consistent with the increase in Δ.

Figure 6 shows the results of experiment IV-D.

V. Conclusion

We presented distributed algorithms for two different graph problems. Based on our prior work on vertex cover and the current work on directed edge coloring, we believe that our basic approach of a matching based automata may be applicable to a variety of graph algorithms with good results.

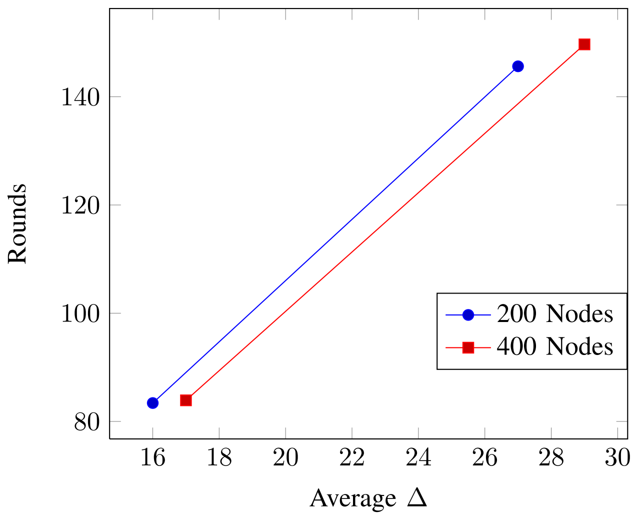


Fig. 6: Strong Edge Coloring of Directed Erdos-Renyi Graph

Our experiments supported our conjecture that we would get good results from these algorithms on a variety of graph structures. In particular, we were able to color scale free graphs with the minimum number of possible colors in every case.

Our experiments also supported our conjecture that the number of computation rounds required to solve the edge coloring problem would scale with Δ rather than with n, tending to be around 2Δ for edge coloring and 4Δ for strong directed edge coloring.

In future work we intend to improve on the experimental and theoretical results presented here and define the properties of problems that can be solved using our state machine.

References

[1] L. Barenboim and M. Elkin, “Distributed deterministic edge coloring using bounded neighborhood independence,” in *Proceedings of the 30th annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, ser. PODC ’11. New York, NY, USA: ACM, 2011, pp. 129–138. [Online]. Available: http://doi.acm.org/10.1145/1993806.1993825

[2] C. Barrett, G. Istrate, V. Kumar, M. Marathe, S. Thite, and S. Thulasidasan, “Strong edge coloring for channel assignment in wireless radio networks,” in *Pervasive Computing and Communications Workshops, 2006. PerCom Workshops 2006. Fourth Annual IEEE International Conference on*, 2006, pp. 5 pp. –110.

[3] J. P. Daigle and S. K. Prasad, “A matching based automata for distributed graph algorithms,” in *Proceedings of the 25th IEEE International Parallel & Distributed Processing Symposium, Workshops and Phd Forum*. IEEE Computer Society, May 2011.

[4] S. Gandham, M. Dawande, and R. Prakash, “Link scheduling in sensor networks: distributed edge coloring revisited,” in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 4, March 2005, pp. 2492 – 2501 vol. 4.

[5] D. A. Grable and A. Panconesi, “Nearly optimal distributed edge colouring in o(log log n) rounds,” in *Proceedings of the eighth annual ACM-SIAM symposium on Discrete algorithms*, ser. SODA ’97. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1997, pp. 278–285. [Online]. Available: http://dl.acm.org/citation.cfm?id=314161.314266

[6] A. Gutteridge, “igraph’s igraph-0.3.3 documentation,” November 2007. [Online]. Available: http://igraph.rubyforge.org/igraph/

[7] I. A. Kanj, A. Wiese, and F. Zhang, *Local Algorithms for Edge Colorings in UDGs*, ser. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer-Verlag, 2010, vol. 5911, ch. Local Algorithms for Edge Colorings in UDGs, pp. 202–213.

[8] F. Kuhn and R. Wattenhofer, "On the complexity of distributed graph coloring," in *PODC '06: Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*. New York, NY, USA: ACM, 2006, pp. 7–15.

[9] V. S. A. Kumar, M. V. Marathe, S. Parthasarathy, and A. Srinivasan, "End-to-end packet-scheduling in wireless ad-hoc networks," in *SODA '04: Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2004, pp. 1021–1030.

[10] M. V. Marathe, A. Panconesi, and j. Larry D. Risinger, "An experimental study of a simple, distributed edge-coloring algorithm," *J. Exp. Algorithmics*, vol. 9, p. 1.3, 2004.

[11] A. Panconesi and A. Srinivasan, "Randomized distributed edge coloring via an extension of the chernoff–hoeffding bounds," *SIAM J. Comput.*, vol. 26, pp. 350–368, April 1997. [Online]. Available: http://dl.acm.org/citation.cfm?id=249364.249368