Fair Maximal Independent Sets

Jeremy T. Fineman Calvin Newport Micah Sherr Tonghe Wang

Department of Computer Science
Georgetown University, Washington, DC, U.S.A.
{jfineman, cnewport, msherr}@cs.georgetown.edu and tw473@georgetown.edu

***Abstract*—Finding a maximal independent set (MIS) is a classic problem in graph theory that has been widely studied in the context of distributed algorithms. Standard distributed solutions to the MIS problem focus on time complexity. In this paper, we also consider *fairness*. For a given MIS algorithm $\mathcal{A}$ and graph G, we define the *inequality factor* for $\mathcal{A}$ on G to be the largest ratio between the probabilities of the nodes joining an MIS in the graph. We say an algorithm is *fair* with respect to a family of graphs if it achieves a constant inequality factor for all graphs in the family. In this paper, we seek efficient and fair algorithms for common graph families. We begin by describing an algorithm that is fair and runs in $O(\log^* n)$-time in rooted trees of size n. Moving to unrooted trees, we describe a fair algorithm that runs in $O(\log n)$ time. Generalizing further to bipartite graphs, we describe a third fair algorithm that requires $O(\log^2 n)$ rounds. We also show a fair algorithm for planar graphs that runs in $O(\log^2 n)$ rounds, and describe an algorithm that can be run in any graph, yielding good bounds on inequality in regions that can be efficiently colored with a small number of colors. We conclude our theoretical analysis with a lower bound that identifies a graph where all MIS algorithms achieve an inequality bound in $\Omega(n)$—eliminating the possibility of an MIS algorithm that is fair in all graphs. Finally, to motivate the need for provable fairness guarantees, we simulate both our tree algorithm and Luby's MIS algorithm [13] in a variety of different tree topologies—some synthetic and some derived from real world data. Whereas our algorithm always yield an inequality factor ≤ 3.25 in these simulations, Luby's algorithms yields factors as large as 168.**

***Keywords*-MIS; fairness;**

I. Introduction

In graph theory, an independent set of an undirected graph $G = (V, E)$ is a subset of nodes $I \subseteq V$ such that no two nodes in I neighbor each other in E. An independent set I is considered a *maximal independent set* (MIS) if all vertices in $V \setminus I$ neighbor a node in I. Distributed solutions to the MIS problem provide a key symmetry-breaking function. Accordingly, these algorithm are often used as subroutines in higher-level applications, including the construction of network backbones [6, 9], leader election [4], resource allocation [19], key exchange [7], and image processing [15].

Existing work on distributed MIS algorithms focuses on optimizing *time complexity*. This paper, by contrast, studies a novel attribute of such solutions: *fairness*. For a given randomized MIS algorithm $\mathcal{A}$ and graph G, we define the *inequality factor* for $\mathcal{A}$ on G to be the largest ratio between the join probabilities of the nodes in the graph. Consider, for example, Luby's classic MIS algorithm [13] executed in a star graph over nodes $V = \{u_1, u_2, ..., u_n\}$ centered on u_1. It is not hard to show that the nodes in $V \setminus \{u_1\}$ are roughly n times more likely to join the MIS than u_1. We would say, therefore, that Luby's algorithm in the star graph has an inequality factor of $\Theta(n)$, which, from a fairness perspective, is poor performance. (In Section IX, we simulate Luby's in a large variety of network topologies—both synthetic and real—and show that Luby's lack of fairness is not confined to contrived examples like the above star graph, but is in fact common in practice.)

The best possible inequality factor for an algorithm in a given setting is 1, indicating that all nodes join with exactly the same probability. In this paper, we consider any constant value to be sufficient to consider an algorithm *fair*. In more detail, we say that an MIS algorithm is *fair* with respect to a family of graphs if there exists some constant such that the algorithm's inequality is upper bounded by this constant for all graphs in that family. Though fairness is non-trivial in both the centralized and distributed setting, we focus here on distributed MIS solutions.

A. Motivation

Our study of fairness has both practical and theoretical motivations. From a practical perspective, we note that MIS algorithms are often used as subroutines by higher-level applications [6, 7, 9, 15, 19]. It follows that joining (or not joining) the MIS might have a consequence in terms of a node's expected work. In constructing a network backbone, for example, joining the MIS consigns a node to processing much more traffic than a non-MIS node in the same network. Similarly, in a network monitoring application that has MIS nodes log behavior of their neighbors, being in the MIS consigns a node to filling up its storage at a higher rate than its non-MIS neighbors. In both cases, using a fair MIS algorithm as a subroutine can ensure that the corresponding workload is more evenly distributed.

From a theoretical perspective, we are motivated by the fact that the problem is interesting. The fairness of an MIS algorithm captures something fundamental about the relationship between the underlying graph structure and the possible behavior of symmetry-breaking strategies.

B. Results

Our goal is to find efficient and fair randomized distributed MIS algorithms for common graph families. We begin, in Section IV, by describing a fair algorithm that runs in $O(\log^* n)$ time in rooted trees of size n. In Section V, we move to unrooted trees and describe a fair algorithm that runs in $O(\log n)$ time. Generalizing further to bipartite graphs, we describe, in Section VI, a third algorithm (leveraging the network decomposition strategy of Linial and Saks [12]) that maintains fairness at the cost of a slightly worse $O(\log^2 n)$ time complexity. Our final upper bound, described in Section VII, guarantees an inequality factor bounded by $O(k)$ in any k-colorable graph with a runtime of $O(\log^2 n + f(n, k))$ rounds, where $f(n, k)$ is the complexity of the relevant distributed k-coloring algorithm. Combining this result with known coloring algorithms for planar [1] graphs and more generally low-arboricity graphs, we obtain fair MIS algorithms for these graphs families that run in $O(\log^2 n)$ time. We emphasize that this algorithm can be executed in *any* graph without needing advance knowledge of the colorability, yielding good inequality factors in regions of the network that can efficiently be colored with a small number of colors. Section VIII concludes our theoretical results with a lower bound that identifies a graph in which all MIS algorithms have an inequality factor in $\Theta(n)$—proving the impossibility of an algorithm that is fair in all graphs.

We next turn our attention, in Section IX, to simulation-based evaluation. Though our algorithms are provably *fair* in the relevant graph families, these results are only useful if existing algorithms are *unfair* in these same graphs. To investigate this question we simulate Luby's $O(\log n)$-time MIS algorithm [13][1] in a variety of different tree topologies: some synthetic and some derived from real-world data. We then compare its performance—with respect to inequality factor—to our provably fair tree algorithm from Section V.

In regular trees, Luby's inequality factors were bounded from above by 6.42. For non-regular trees this inequality factor grew to as large as 36. For trees extracted from real-world traces, the factor jumped to a maximum of 168—indicating significant inequality. In all cases, by contrast, our fair algorithm had inequality factors of at most 3.25. These results motivate the need for MIS algorithms with explicit fairness guarantees for scenarios where this property matters.

C. Contributions

This paper offers the following contributions to both the theory and practice of distributed graph algorithms: (1) it introduces the notion of *fairness* as a practically-motivated, tractable but non-trivial property of MIS algorithms; (2) it establish by simulation that Luby's algorithm can be (perhaps surprisingly) unfair in practice—even in trees; and (3) it provides new MIS algorithms that are both efficient and provably fair in a variety of common graph families.

[1]This is arguably the most commonly used distributed MIS solution as it is simple and offers a near-optimal time complexity. As of the writing of this introduction, for example, the journal version of this result has been cited over 875 times, according to Google Scholar.

II. Related Work

Luby's oft-cited distributed MIS algorithm [13] computes an MIS in $O(\log n)$ time in general graphs. There are faster known algorithms for restricted graph classes, including an $O(\sqrt{\log n \log\log n})$ algorithm for trees [11] (which nearly matches an $\Omega(\sqrt{\log n})$ lower bound [10]), an $O(\log^* n)$ algorithm for growth-bounded graphs [17], and multiple $o(\log n)$-time algorithms for low-degree graphs (e.g., [2]). In general graphs, the MIS problem can also be solved deterministically in $2^{O(\sqrt{\log n})}$ rounds [16]. For a fixed assignment of IDs and graph our definition of fairness is not useful for deterministic algorithms (any connected graph of size $n > 1$ will have an infinite inequality factor). If we assume, however, that the unique IDs used by the deterministic algorithm are assigned according to some probability distribution, its fairness becomes once again non-trivial.

Harris *et al.* [5] study the average degree of the MIS nodes in the graph. This complementary study supports our argument that distributed symmetry breaking is interesting beyond just the time complexity perspective. Métivier *et al.* [14] consider the correlation between nodes joining the MIS: they show that for bounded-degree graphs, the correlation degrades quickly as the distance increases, but also that this relationship does not hold in general. Uncorrelated join probabilities, however, is neither necessary nor sufficient to imply fairness.

III. Model

We assume the standard synchronous message-passing model defined with respect to some undirected graph $G = (V, E)$, where $n = |V|$ denotes the number of nodes. Without loss of generality, assume that each vertex is assigned a unique identifier. Each vertex knows its own ID and its immediate neighbors' IDs, as well as n. It has no other *a priori* topology information. As is standard for local graph algorithms, we assume that the maximum message size is bounded at $O(\log n)$ bits (i.e., enough for a constant number of IDs). Our lower bound (Section VIII), however, holds even for unbounded message size.

In the following, for $U \subseteq V$, we use $G(U)$ to refer to the subgraph of G induced by vertex set U, which comprises the vertices U and the edges of G whose endpoints are both in U. We use $N(u)$, for vertex u, to denote the neighbors of u in G. We also use $D(G) = \max_{u,v \in V} d_G(u, v)$ to denote the diameter of G, where $d_G(u, v)$, or $d(u, v)$ for simplicity, denotes the length of the shortest path from u to v in G.

A maximal independent set (MIS) algorithm, executed in $G = (V, E)$, must satisfy: *termination*, meaning that every node output a 1 (to indicate it joins the set) or a 0 (to indicate it does not), *independence*, meaning the subset $\mathcal{I} \subseteq V$ of nodes that output 1 is independent ($u, v \in \mathcal{I}, u \neq v \implies (u,v) \notin E$), and *maximality*, meaning every node in $V \backslash \mathcal{I}$ neighbors at least one node in $\mathcal{I}$. We consider randomized distributed algorithms and require that termination hold *with high probability* (i.e., probability at least $1 - 1/n^c$, for constant $c \geq 1$) while independence and maximality always hold. For a given MIS algorithm $\mathcal{A}$, graph $G = (V, E)$, and node $u \in V$, $P_{\mathcal{A},G}(u)$ denotes the probability that u joins the MIS when executing $\mathcal{A}$ in G. We use this property to formalize fairness as follows. (We define division-by-zero to evaluate to infinity.)

Definition 1. *The* inequality factor *of an MIS algorithm $\mathcal{A}$ w.r.t. a graph G is defined as*

$$F_{\mathcal{A}}(G) = \max_{u,v \in V} \left\{ \frac{P_{\mathcal{A},G}(u)}{P_{\mathcal{A},G}(v)} \right\}.$$

We also use $F_{\mathcal{A}}(\mathcal{G}) = \max_{G \in \mathcal{G}} F_{\mathcal{A}}(G)$, to denote the worst-case inequality factor of $\mathcal{A}$ w.r.t. a graph family $\mathcal{G}$.

Definition 2. *For a graph family $\mathcal{G}$, we say an MIS algorithm $\mathcal{A}$ is* fair *w.r.t. $\mathcal{G}$ if there exists a constant c s.t. $F_{\mathcal{A}}(\mathcal{G}) \leq c$.*

Informally, $F_{\mathcal{A}}(\mathcal{G})$ quantifies the worst-case ratio of join probabilities for any two nodes when running $\mathcal{A}$ in any graph drawn from $\mathcal{G}$. The lowest possible value of $F_{\mathcal{A}} = 1$ indicates that an algorithm is *perfectly fair* (i.e., all nodes have an equal chance of entering the MIS).

IV. Fair Algorithm for Rooted Trees

This section describes a fair MIS algorithm for rooted trees that runs in $O(\log^* n)$ time. This straightforward result provides a nice primer for our later results and highlights the general strategy we deploy throughout this paper: build an initial independent set with strong fairness guarantees, then refine it (using a potentially unfair MIS algorithm on the remaining uncovered nodes) to guarantee maximality.

A. Algorithm

The algorithm FairRooted runs in a rooted tree $T = (V, E)$, where each internal node is provided a pointer to its parent in the tree. (It is trivial to extend this algorithm to operate on a forest of rooted trees.) We will prove each node enters the MIS with probability $\geq 1/4$, implying fairness.

The algorithm proceeds in two stages, as shown by pseudocode in Figure 1. During the first stage, which consists of a single round of communication, each node tags itself with a single bit chosen uniformly at random. We associate with the root a virtual sentinel node v_0 to act as its parent, and have the root also choose the tag of its (virtual) parent. Each node then shares its tag with its children and compares its own tag to that of its parent. Any node with a tag of 0 whose parent has a tag of 1 enters the set $\mathcal{I}$.

By construction, the set $\mathcal{I}$ is an independent set, but it may not be maximal. It is not hard to show that the first stage alone guarantees that each vertex enters $\mathcal{I}$ with constant probability, which implies fairness. In the second stage, each node communicates once with all its neighbors, and any node that is in $\mathcal{I}$, or learns that it is covered by a node in $\mathcal{I}$, terminates the algorithm arriving at its final state. Those nodes that are not yet covered continue by participating in a generic MIS algorithm (possibly an unfair one) for rooted trees to ensure the final set is maximal; e.g., the $O(\log^* n)$-round algorithm of [3].

```
Stage 1: (∀v ∈ V)
  choose v.tag from {0,1} uniformly at random
  if v is the root then
      choose v_0.tag from {0,1} at random
  read u.tag from parent node u
  if v.tag = 0 and u.tag = 1 then v joins I

Stage 2: (∀v ∈ V)
  if v ∈ I then
      output "in MIS" and terminate
  else if N_T(v) ∩ I ≠ ∅ then
      output "not in MIS" and terminate
  else run an (unfair) MIS protocol for rooted trees
```

Figure 1. The two stages of the algorithm FairRooted.

B. Analysis

The following theorem proves the desired fairness and time complexity results for FairRooted.

Theorem 3. *Let $\mathcal{R}$ denote the class of rooted trees. When run on any rooted tree $T \in \mathcal{R}$,* FairRooted *generates a correct MIS in $O(\log^* n)$ rounds. Moreover, it guarantees an inequality factor $F_{\text{FairRooted}}(\mathcal{R}) \leq 4$.*

Proof of the theorem follows directly from the three subsequent lemmas, which prove correctness, running time, and fairness, respectively.

Lemma 4. *When run in any rooted tree $T = (V, E)$,* FairRooted *generates a correct MIS.*

Proof of Lemma 4 is straightforward. The crux of the proof is that in Stage 1, no two neighbors may join $\mathcal{I}$ giving independence, and in Stage 2, maximality is restored.

Since Stage 1 and 2 use a constant number of rounds plus Cole and Vishkin's algorithm for rooted trees [3], it is straightforward to prove the following lemma. Moreover, Cole and Vishkin's algorithm is deterministic, so the bound holds in the worst case if nodes are given unique IDs in the range from 0 to $n^{\Theta(1)}$. If not, the nodes begin by choosing random IDs in this range, and the bound holds with high probability.

Lemma 5. FairRooted *completes in $O(\log^* n)$ rounds.* ■

Lemma 6. *Let $\mathcal{R}$ be the class of rooted trees.* FairRooted *has inequality factor $F_{\text{FairRooted}}(\mathcal{R}) \leq 4$.*

Proof: Consider any node $v \in V$, and let u be its parent. Then in the first stage, we have $Pr\{v \in \mathcal{I}\} = Pr\{u.tag =$

1 and $v.tag = 0\} = Pr\{u.tag = 1\} \cdot Pr\{v.tag = 0\} = (1/2)(1/2) = 1/4$.

In Stage 2, nodes are only *added* to the MIS, so we conclude that the probability that any node enters the MIS is at least $1/4$. Since the maximum probability is 1, the inequality bound follows. ■

V. Fair Algorithm for Unrooted Trees

In this section, we describe a fair MIS algorithm for unrooted trees that runs in $O(\log n)$ time. To explain this algorithm, we first note that it is not difficult to create a *centralized* algorithm $\mathcal{A}'$ that guarantees $P_{\mathcal{A}',G}(u) = P_{\mathcal{A}',G}(v)$ $\forall u, v \in V$, for any $G \in \mathcal{B}$, where $\mathcal{B}$ is the class of bipartite graphs. The real challenge is to find an efficient *distributed* algorithm that can approximate this guarantee. Here we tackle this challenge for $\mathcal{T} \subset \mathcal{B}$, where $\mathcal{T}$ is the class of unrooted trees, and we generalize to bipartite graphs in Section VI, albeit at the cost of a slower algorithm.

More precisely, we describe below a distributed MIS algorithm FairTree that when run on a graph $G \in \mathcal{T}$, guarantees a correct MIS such that $P_{\text{FairTree},G}(u) \geq (1-\epsilon)/4$, for every node u and an arbitrarily small ϵ. It terminates in $O(\log n)$ rounds, with high probability (matching the running time of Luby's algorithm [13]).

A. Algorithm

Figure 2 describes FairTree. This algorithm uses, as a subroutine, CntrlFairBipart, a distributed algorithm that can generate a perfectly fair MIS on unrooted trees (or, more generally, any bipartite graph) in $O(D(T))$ time, where T is the tree in which it is executed. At a high level these two algorithms work together as follows: FairTree starts by having nodes partition the original tree T, in a distributed manner, into components of smaller size. They then execute CntrlFairBipart in each of these resulting components, covering them with a fair MIS. Next, a careful *stitching* process (Stages 2–3) is used to resolve MIS conflicts between neighboring components in a local manner. It is straightforward to show that our initial partition breaks T into components with diameter $O(\log n)$, with high probability, allowing CntrlFairBipart to run in $O(\log n)$ time. It is more involved to then prove that the stitching step is also efficient *and* only increases the inequality of the algorithm by a small constant factor.

Stage 1: Cut ($\forall v \in V$)
- cooperate with each neighbor $u \in N_T(v)$,
  - and set $(u,v).cut = 1$ with probability $1/2$
- call CntrlFairBipart with $\widehat{D} = \gamma$,
  - but ignoring edges with $cut = 1$
- **if** v joined MIS **then** add v to $\mathcal{I}$

Stage 2: Resolve ($\forall v \in \mathcal{I}$)
- call CntrlFairBipart with $\widehat{D} = \gamma$
- **if** v joined MIS **then** keep v in $\mathcal{I}$
- **else** remove v from $\mathcal{I}$

Stage 3: Maximalize ($\forall v$ s.t. $(N_T(v) \cup \{v\}) \cap \mathcal{I} = \emptyset$)
- call CntrlFairBipart with $\widehat{D} = \gamma$
- **if** v joined MIS **then** add v to $\mathcal{I}$

Stage 4: Fix ($\forall v \in V$)
- **if** $v \in \mathcal{I}$ and $N_T(v) \cap \mathcal{I} \neq \emptyset$ **then**
  - remove v from $\mathcal{I}$
- **if** $v \in \mathcal{I}$ **then** output "in MIS" and terminate
- **else if** $N_G(v) \cap \mathcal{I} \neq \emptyset$ **then**
  - output "not in MIS" and terminate
- **else** call Luby's and mimic output

Figure 2. The four stages of FairTree algorithm. In the above, $\gamma = \Theta(\log n)$ and each stage runs for a fixed number of rounds (i.e., nodes not participating in a stage still wait the fixed number of rounds before proceeding to the next stage). The sets next to each stage name describe the nodes that participate in that step.

The CntrlFairBipart Algorithm. The algorithm takes a single parameter, $\widehat{D}$, which is an estimated upper bound on the diameter of the bipartite graph in which it is executed.[2] The algorithm starts by having each node run a basic flood-based leader election algorithm for $\widehat{D}$ rounds: each node in each round broadcasts the largest ID it has received so far; it accepts the largest ID its seen at the end of $\widehat{D}$ rounds as its leader. After this stage concludes, the leader u (or, potentially multiple leaders if $\widehat{D}$ is an underestimate), selects a bit b_u with uniform randomness. It then initiates a breadth-first search, beginning at itself, terminating after $\widehat{D}$ rounds, even if some nodes have not been reached. The search message includes the current depth of the search (u considers itself at level 0) and b_u. Each node (including u), that learns it is in some level i (i hops away from the leader), joins to the MIS if $i + b_u \equiv 0 \pmod 2$. As a special case, if the leader is alone (i.e., has degree 0), it always joins the MIS. It is straightforward to show:

Lemma 7. *Assume* CntrlFairBipart *is called by every node in unrooted tree $T = (V,E)$ during the same round with the same parameter $\widehat{D}$. Let $I(T)$ be the nodes that join the MIS. If $\widehat{D} \geq D(T)$, then: (a) $I(T)$ is a correct MIS for T; and (b) $\forall u \in V : P_{\text{CntrlFairBipart},T}(u) = 1/2$ if $|V| > 1$, and $P_{\text{CntrlFairBipart},T}(u) \geq 1/2$ in general.*

Proof: If $\widehat{D} \geq D(T)$, then the leader election correctly elects a single leader and the breadth-first search reaches all nodes. Because T is a tree, it is easy to see that (a) holds. To show (b), fix some node $v \in T$. Let u be the leader in T. If $|V| > 1$, then depending on the parity of $d_T(u,v)$, one value for b_u will put v in $I(T)$ and one will keep v out of $I(T)$. The fairness follows from the fact that b_u is chosen with uniform probability. If $|V| = 1$, then the node always enters the MIS. ■

The FairTree Algorithm. We now describe the main result of this section, the FairTree algorithm. This algo-

[2] Note that we do not assume advance knowledge of diameter information in our model. The CntrlFairBipart algorithm described here will be called as a subroutine by our FairTree algorithm, which will be responsible for specifying the $\widehat{D}$ parameter.

rithm consists of four *stages* described in Figure 2. Notice, not all nodes participate in each stage (the participants are specified by the set next to the stage name). The first three stages, however, each run for a fixed number of rounds (the $\Theta(\log n)$ time required by the call to CNTRLFAIRBIPART, plus the constant number of extra rounds needed for local communication, when relevant), so non-participants simply wait that number of rounds before proceeding to the next stage. This ensures all nodes start each of these stages during the same round.

Stage 1 divides the tree into components by cutting edges and then attempts to create a fair MIS $\mathcal{I}$ in each. If Stage 1 completes, $\mathcal{I}$ covers all nodes, but there may be MIS conflicts between neighbors in distinct components. Stage 2 resolves these conflicts by running CNTRLFAIRBIPART only on "MIS" nodes (those in $\mathcal{I}$), causing some nodes to drop out of $\mathcal{I}$. If both stages complete, then $\mathcal{I}$ is an independent set, albeit not necessarily maximal. Stage 3 restores maximality by running CNTRLFAIRBIPART on uncovered nodes. The resulting MIS stitches safely with the existing set $\mathcal{I}$.

We will prove that the components executing CNTRL-FAIRBIPART in each stage have sufficiently small diameters for this fixed-time algorithm to succeed, with high probability. In this successful case, $\mathcal{I}$ is a valid MIS, and we shall prove that $P_{\text{FAIRTREE},T}(u) \geq 1/4$ for every node u. With low probability, however, we might arrive at Stage 4 with an invalid MIS due to CNTRLFAIRBIPART failing to complete on a large diameter component in previous stages. To correct for this possibility, at the start of Stage 4 we remove all independence violations, then have uncovered nodes run a standard MIS algorithm (in this paper, we use the MIS algorithm of Luby [13], which we call LUBY'S). The resulting MIS will stitch together with the earlier independent set, but we make no guarantee about its inequality factor. In other words, LUBY'S is only ever called as a fallback mode in the low probability event that one of the previous three stages failed. This ensures that the MIS is always correct, but the join probability decreases by some $\epsilon \leq 1/n$.

B. Analysis

The following theorem proves the desired inequality factor and time complexity results for FAIRTREE.

Theorem 8. *For any unrooted tree $T = (V, E)$, the* FAIRTREE *algorithm, when executed in T, constructs a correct MIS such that $P_{\text{FAIRTREE},T}(u) \geq (1-\epsilon)/4$, for every $u \in V$ and some $\epsilon < 1/n$. It terminates in $O(\log n)$ rounds with high probability.*

To establish this theorem, we have three properties to prove: the time complexity, correctness, and inequality factor bound of FAIRTREE. We divide these efforts into the three lemmas below. To distinguish the value of our set $\mathcal{I}$ at the end of each stage, we use the notation $\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3$ to describe $\mathcal{I}$ at the end of Stages 1, 2 and 3, respectively.

Lemma 9. *Algorithm* FAIRTREE *terminates in $O(\log n)$ rounds with high probability.*

Proof: This time complexity follows directly from the fixed-length of Stages 1 to 3 and the $O(\log n)$ bound (*w.h.p.*) on LUBY'S proved in [13]. ■

Lemma 10. FAIRTREE *generates a correct MIS on T.*

Proof: Stage 4 starts by removing any independence violations, then ensures maximality by running LUBY'S on the remaining uncovered nodes. ■

As seen above, efficiency and correctness are straightforward to prove. Bounding the inequality factor, on the other hand, requires a more involved argument:

Lemma 11. *$P_{\text{FAIRTREE},T}(u) \geq (1-\epsilon)/4$, for all $u \in V$ and some $\epsilon \leq 1/n$.*

Proof: We consider Stages 1–3 *successful* if their calls to CNTRLFAIRBIPART are made with a sufficiently large value for $\widehat{D}$ (i.e., a value at least as large as the largest relevant component).

We will first show that if Stages 1–3 are successful, then no node will call LUBY'S in Stage 4. To see why this is true, notice that a node calls LUBY'S only if it ends Stage 3 uncovered by $\mathcal{I}$. Every node that is uncovered at the beginning of Stage 3, however, calls CNTRLFAIRBIPART, and by our assumption that this call is successful, each of these nodes ends the stage covered.

We have just established that if the first three stages are successful then no node will call LUBY'S. In other words, the MIS defined at the end of Stage 3 will be the final MIS. We next note that under this assumption of successful stages, this final MIS is fair. In more detail, the probability that a given u joins the MIS is at least

$$Pr\{u \in \mathcal{I}_2\} = Pr\{u \in \mathcal{I}_1\} \cdot Pr\{u \in \mathcal{I}_2 | u \in \mathcal{I}_1\}.$$

(Since Stage 3 only increases the join probability, it may be omitted when proving a lower bound.) Lemma 7 implies Stage 1 yields $Pr\{u \in \mathcal{I}_1\} \geq 1/2$. Stage 2 runs CNTRL-FAIRBIPART again on the nodes $\mathcal{I}_1$, giving $Pr\{u \in \mathcal{I}_2 | u \in \mathcal{I}_1\} \geq 1/2$. Combining these gives $Pr\{u \in \mathcal{I}_2\} \geq 1/4$.

Our final step is to incorporate the event that the first three stages are not successful. If this event occurs, for any u, we can trivially bound u's probability of joining at 0. Let ϵ be the probability that at least one of these stages fails is not successful. Combined with our result from above, we get that u joins the MIS with probability at least $(1-\epsilon)/4$.

We are left to bound ϵ. Our goal is to show that it is less than $1/n$, which will implies the inequality factor is upper bounded by a value that approaches 4 as n increases. To reach this goal, we argue that there exists a constant c for $\gamma = c \log n + \Theta(1)$ such that in all three calls to CNTRLFAIRBIPART, γ is a sufficiently large estimate of the maximum component diameter. We start by studying the calls in the first two stages. In these stages, a given path of length ℓ is included in a component with probability $2^{-\ell}$ (in Stage 1, the path must have $cut = 0$ for every edge, while in Stage 2, the path must have $cut = 1$ for every edge, where the cut decisions are all independent and uniform). Setting $\gamma = c \log n$ for a sufficiently large constant c, a union bound over the polynomially-many paths in T shows that for each of Stages 1 and 2, $Pr\{\text{length of any path} \geq \gamma\} \leq 1/(3n)$.

For Stage 3, we note that for a path $u_1, u_2, ..., u_\ell$ of length ℓ consisting of nodes uncovered by $\mathcal{I}_2$, each u_i was never previously in $\mathcal{I}$ (if it was, it would be covered in this stage), and, therefore, each u_i has a neighbor v_i that was in $\mathcal{I}_1$ but not $\mathcal{I}_2$. Furthermore, because T is a tree, for v_i, v_j, $i \neq j$, v_i and v_j are in different components in Stage 2 (they are separated by the path $u_i, \ldots, u_j$) and therefore their outcomes in Stage 2 are independent. Lemma 7 thus implies $Pr\{v_i \notin \mathcal{I}_2\} \leq 1/2$ independently for each v_i. The same argument as for Stages 1 and 2 shows that for sufficiently large γ, the probability that Stage 3 contains a large-diameter component is at most $1/(3n)$. A final union bound over stages gives us the desired $\epsilon < 1/n$. ■

VI. Algorithm for Bipartite Graphs

This section describes a fair distributed MIS algorithm for the class of bipartite graphs that runs in $O(\log^2 n)$ time. This algorithm generalizes the algorithm of Section V, but does not subsume it because it is slower by a log-factor. As with Section V, the main idea of the algorithm is to partition the graph into low-diameter subgraphs, find an MIS on the subgraphs using CNTRLFAIRBIPART, and then correct for maximality by running LUBY'S on the uncovered nodes. Unfortunately, the Section V's partitioning algorithm does not generalize to bipartite graphs, so a more pliable approach is necessary. Fortunately, such an approach exists in a classic network decomposition result due to Linial and Saks [12], which we leverage to quickly produce a useful low weak-diameter subgraph in $O(\log^2 n)$ rounds.

Below, we describe and analyze the useful subroutine we adopt from [12], then describe and analyze our fair MIS algorithm that makes use of it.

A. Construct_Block Routine

Linial and Saks' network-decomposition algorithm [12] uses a key subroutine called "Construct_Block." In this routine, each node v chooses a communication range r_v according to the distribution π, specified as:

$$\pi : Pr\{r_v = k\} = \begin{cases} p^k(1-p) & \forall k \in 0, 1, \cdots, \gamma - 1 \\ p^\gamma & k = \gamma \end{cases}$$

where the parameters p and γ represent a probability and maximum range, respectively. For our purposes, $p = 1/2$ and $\gamma = \Theta(\log n)$ suffice.

Next, each node v broadcasts its ID to all other nodes within radius r_v. After the broadcast completes, node v identifies the node u with the largest ID among messages it has received. It considers u its leader. If the distance between v and u is strictly less than r_u, then v joins u's ***block***. In other words, u's block is the set of nodes that select u as a leader and join a block. Otherwise, the distance between v and its leader is exactly r_u, in which case v is called a ***boundary node***. Boundary nodes do not join any block.

As stated in the following lemma, proved in [12], the Construct_Block routine provides some nice structural guarantees. Notably for correctness, if a node v joins a block, then each of its neighbors is either a boundary node or in the same block.

Lemma 12. *Construct_Block has the following properties (i) each vertex belongs to a block with the probability of at least $p(1-p^\gamma)^n$; (ii) all connected non-boundary nodes have the same leader.* ■

The implementation of Construct_Block described in the next section requires $O(\log^2 n)$ rounds. Though this is slower by a $O(\log n)$ factor than our FAIRTREE algorithm for unrooted trees, the structural properties on the blocks (as described by Lemma 12) are stronger than what we get from our FAIRTREE partitioning. In particular, the blocks we obtain here are separated by non-block boundary nodes (follows from ii), allowing us to avoid independence conflicts between neighboring nodes in different blocks.

B. Algorithm

Figure 3 contains pseudocode describing FAIRBIPART, our fair distributed MIS algorithm for bipartite graphs. Roughly speaking, the goal of Stage 1 is to run Construct_Block to create blocks, and then run CNTRLFAIRBIPART on each block. However, each block has only

Stage 1: block construction ($\forall v \in V$)
- pick r_v according to distribution π
- choose b_v from $\{0, 1\}$ uniformly at random
- create tables $v.L[0..\gamma]$ and $v.B[0..\gamma]$
- initialize $v.L[r_v] = v$ and $v.B[r_v] = b_v$.
- **repeat** γ times
  - broadcast leader table $v.L$ and $v.B$
  - **on receiving** leader tables L^*, B^*:
    - **for all** $i \in \{0, 1, 2, \ldots, \gamma - 1\}$ **do**
      - **if** $v.L[i] > L^*[i+1]$ **then**
        - $v.L[i] = L^*[i+1]$
        - $v.B[i] = \neg B^*[i+1]$
- let $leader\_ID = \max_i v.L[i]$ and $j = \text{argmax}_i\, v.L[i]$
- **if** $leader\_ID \neq v.L[i]$ for any $i > 0$ **then**
  - v is a boundary node and does not join a block
- **else** v joins $leader\_ID$'s block
  - **if** $v.B[j] = 1$ **then** v joins $\mathcal{I}$

Stage 2: MIS ($\forall v \in V$)
- **if** $v \in \mathcal{I}$ **then**
  - output "in MIS" and terminate
- **else if** $N(v) \cap \mathcal{I} \neq \emptyset$ **then**
  - output "not in MIS" and terminate"
- **else** call LUBY'S and mimic output

Figure 3. Pseudocode of FAIRBIPART

weak diameter guarantees in the sense that the distance in G between any two nodes in the same block is small (by construction), but the distance between those nodes in the subgraph induced by the block nodes may be arbitrarily large. To compensate for this reality, Stage 1 simulates CNTRLFAIRBIPART along with the execution of Construct_Block by sending the extra random bit selected by the leader along with each message.

For concreteness, Figure 3 provides full pseudocode for Stage 1. There are several variants for Construct_Block described in [12]. Here we adopt one with bounded message sizes, allowing $O(\log n)$ bits per edge per round of communication (as required by our model). More precisely, Stage 1 operates as follows. Each node v initially selects a range or $r_v \leq \gamma = \Theta(\log n)$ according to the distribution π described above. To simulate CNTRLFAIRBIPART, each node also selects a random bit b_v. Each node v constructs "leader tables" $v.L[0\,..\,\gamma]$ and $v.B[0\,..\,\gamma]$, where

$$v.L[i] = \text{maximum ID } v \text{ has seen with } i \text{ range remaining},$$

$$v.B[i] = \begin{cases} b_u & d(u,v) \text{ is even, where } u = v.L[i] \\ \neg b_u & d(u,v) \text{ is odd, where } u = v.L[i] \end{cases}$$

This table is initialized with $v.L[r_v] = v$ and $v.B[r_v] = b_v$, and all other entries initialized to null values.

The execution of Stage 1 is then divided into γ superrounds. In each superround, a node v sends its full leader tables to its neighbors—since a leader table has γ entries, this can be simulated in $O(\gamma)$ rounds. On receiving a leader table, v decrements the range of all received entries by 1 and updates the appropriate entries in its own leader tables to reflect the maximum ID seen.

After completing the γ superrounds of communication, v selects a leader by looking at the maximum ID stored in its leader table $v.L$; i.e., $\max_i v.L[i]$. If that leader's ID occurs only in $v.L[0]$ then v becomes a boundary vertex. Otherwise, it joins a block. To finish the simulation of CNTRLFAIRBIPART, if v joins a block, and if the corresponding bit in $v.B$ is 1, then v joins the MIS $\mathcal{I}$.

At the start of Stage 2, $\mathcal{I}$ is an independent set but may not be maximal. All nodes that are already covered terminate. Any remaining nodes execute LUBY'S to restore maximality.

C. Analysis

The following theorem proves the desired fairness and time complexity results for FAIRBIPART. In the following, assume we fix $\gamma = 2\lg n$ and $p = 1/2$. Proof of the theorem follows directly from the subsequent three lemmas.

Theorem 13. *Let $\mathcal{B}$ denote the class of bipartite graphs. When run on any bipartite graph $G \in \mathcal{B}$,* FAIRBIPART *generates a correct MIS in $O(\log^2 n)$ rounds, with high probability. Moreover, it guarantees inequality factor $F_{\text{FAIRBIPART},\mathcal{B}} \leq 8$.*

Lemma 14. FAIRBIPART *outputs a correct MIS.*

Proof: This proof amounts to showing that $\mathcal{I}$ produced at the end of Stage 1 is an independent set. If $\mathcal{I}$ is an independent set, then the fact that Stage 2 yields an MIS follows from the same style of argument as in Lemma 4.

To prove that $\mathcal{I}$ is an independent set, we begin by applying Lemma 12. Fix some v with leader u at the end of the Construct_Block logic. All of v's neighbors are either in u's block, or they are boundary nodes. Boundary nodes do not join $\mathcal{I}$, so it is sufficient to show that nodes in the same block that join $\mathcal{I}$ are independent. Below we prove that this follows from the correctness of the CNTRLFAIRBIPART logic integrated in this routine.

We begin by stating an important property of bipartite graphs, which we then use to argue that no neighbors in the same block read the same $v.B[j]$ value at the end of Stage 1. Consider any nodes $u, v \in G$. We say that these nodes have *even distance* if any path between them has even length, and *odd distance* if any path between them has odd length. Since G is bipartite, the notion of even/odd distance is well defined—all paths between a particular pair of vertices have the same parity.

Consider any node v, and let $u = v.L[i]$ be the ith entry in its leader table. A simple inductive argument over rounds of Stage 1 shows that $v.B[i] = b_u$ if and only if u and v have even distance (recall that the algorithm negates this value with each hop). It follows that all of v's neighbors with u in their leader table observe the value $\neg b_u$. Hence v joins $\mathcal{I}$ only if its neighbors do not. ■

Lemma 15. *Algorithm* FAIRBIPART *terminates in $O(\log^2 n)$ rounds.*

Proof: The communication in Stage 1 consists of γ superrounds, which each includes γ rounds. Because we fixed $\gamma = \Theta(\log n)$, we get $O(\log^2 n)$ total rounds. In addition, Stage 2 comprises one round of communication to determine if $N(v) \cap \mathcal{I} = \emptyset$, and we then require $O(\log n)$ rounds (with high probability) for LUBY'S. ■

Lemma 16. *If $G \in \mathcal{B}$ is a bipartite graph, then $P_{\text{FAIRBIPART},G}(u) \geq 1/8$ for all $u \in V$.*

Proof: A node v joins $\mathcal{I}$ at the end of Stage 1 if 1) it joins a block, and 2) the bit corresponding to its leader in $v.B$ is 1. Notice, these two events are independent. It is easy to see that 2 occurs with probability $1/2$.

We now turn our attention to the probability of 1. By Lemma 12, we know that each node joins a block with probability at least $p(1 - p^{\gamma})^n$. As specified above, we fixed $\gamma = 2\lg n$ and $p = 1/2$, which yield a block join probability of $(1/2)(1 - 1/n^2)^n$. Notice that this function is monotonically increasing (approaching $\sqrt{1/e}$ as $n \to \infty$). By assuming $n \geq 2$, we lower bound the probability as: $(1/2)(1 - 1/4)^2 > 1/4$. (The $n = 1$ case can be handled separately.)

Multiplying the probabilities of events 1 and 2 give us a result $\geq (1/4)(1/2) = (1/8)$, completing the proof. ■

Notice, that in the above calculation we can drive the inequality bound arbitrarily close to 4 by replacing $2 \lg n$ with $c \lg n$ for increasing values of c (which pushes the probability joining a block in the above proof toward $1/2$ in the limit as c grows). This increased fairness, however, comes at the expense of time complexity, where γ appears as a multiplicative factor.

VII. A k-Fair Algorithm for k-Colorable Graphs

This section describes a distributed MIS algorithm that guarantees an inequality factor bounded by $O(k)$ (what we can call, *k-fairness*) in $O(f(n,k) + \log^2 n)$, rounds for any graph for which there exists a distributed k-coloring algorithm that runs in $f(n,k)$ time. Combining this algorithm with a known $O(\log n)$-time distributed $O(1)$-coloring algorithm for planar graphs [1], yields a fair distributed MIS algorithm for planar graphs that runs in $O(\log^2 n)$ rounds. More generally, this algorithm can be combined with a general distributed coloring algorithm and executed in arbitrary graphs: it is straightforward to show that it will yield good (i.e., small) inequality factors in regions of the graph that can be colored with a small number of colors.

A. Algorithm

Our ColorMIS distributed MIS algorithm must be combined with a distributed k-coloring algorithm $\mathcal{A}$. It begins by having nodes execute $\mathcal{A}$.[3] It then has nodes execute a variant of our augmented Construct_Block subroutine described and analyzed in Section VI. In more detail, the only change is that we replace the randomly generated bit b_u generated by each node u, with a color c_u, selected from the k colors with uniform randomness. (If nodes do not know k in advance, then we can add an extra step where the leader in each block counts the colors before randomly choosing one. For concision, in the following we assume knowledge of k.)

Unlike with b_u, the selected value c_u remains unchanged as it propagates through the network. A node v joins the MIS at this point only if it joined a block with leader u and v's color, from executing the distributed coloring algorithm, matches the color c_u randomly chosen by its leader. As usual, we conclude by having all uncovered nodes execute Luby's to fix any remaining maximality mistakes.

B. Analysis

The following theorem bounds the fairness and time complexity of ColorMIS:

Theorem 17. *Fix some distributed k-coloring algorithm $\mathcal{A}$ that terminates in $f(n,k)$ rounds in all graphs of size n, with probability at least $1 - 1/n^2$. Let $\mathcal{C}_{\mathcal{A}}$ be the class of graphs that can be k-colored by $\mathcal{A}$. When run on any $G \in \mathcal{C}_{\mathcal{A}}$,* ColorMIS *using $\mathcal{A}$ generates a correct MIS in $O(f(n,k)+\log^2 n)$ rounds, with probability at least $1-1/n$. Moreover, it guarantees an inequality factor $O(k)$.*

Proof: The time complexity follows from the running time of $\mathcal{A}$ and the running time of Construct_Block, as established in Section VI.

To show that the resulting MIS is correct, it is enough to show that there are no independence violations in the set *before* the remaining uncovered nodes run Luby's algorithm. Notice two nodes in the same block join the set only if they share the block leader's color. And by the correctness of $\mathcal{A}$, no two neighbors share the same color.

Finally, we consider fairness. The analysis of Construct_Block in Section VI established that a node joins a block with constant probability. If a node joins a block, the probability that its leader choose its color is $1/k$. Therefore, all nodes join with probability at least $\Omega(1/k)$ yielding the needed $O(k)$ bound on the inequality factor. ■

We obtain the following corollary by combining this theorem with coloring algorithms for low-arboricity graphs [1] that produce an $(\lfloor(2+\epsilon)\cdot a(G)\rfloor + 1)$-coloring in $O(a(G)\log n)$ time, where $a(G)$ is the "arboricity" of G. and $\epsilon > 0$ is an arbitrarily small parameter. Coupled with the fact that planar graphs have arboricity at most 3 yields the following corollary. Note that this bound actually holds for any constant-arboricity graph.

Corollary 18. *There exists a fair distributed MIS algorithm for planar graphs that runs in $O(\log^2 n)$ time, with high probability.* ■

VIII. Lower Bound

In this section we prove limits on fairness. Consider, for example, the *cone* graph $C = (V,E)$, where $V = \{u_0, u_1, ..., u_{2k}\}$, for some $k \geq 1$, and $E = \{(u_i,u_j) \mid i,j > 0\} \cup \{(u_0,u_i) \mid 0 < i \leq k\}$. That is, C consists of a clique among the nodes u_1 to u_{2k}, as well as an edge from u_0 to every node from u_1 to u_k. We prove below that every MIS algorithm has inequality factor $\Omega(n)$ when run in C. This result eliminates the possibility of any *universally fair* MIS algorithm (i.e., an algorithm that can guarantee bounded inequality in all graphs).

An interesting property of C is that the ratio between the largest and smallest degree is constant. This implies that inequality is not just caused by disparities in density in different regions of a graph (e.g., nodes in sparse regions joining with higher probability than nodes in dense regions), but can also have deeper topological roots. A better classification of exactly which properties unavoidably yield inequality remains an intriguing open question.

Theorem 19. *For every MIS algorithm $\mathcal{A}$, $F_{\mathcal{A}}(C) = \Omega(n)$.*

[3] Nodes can execute $\mathcal{A}$ for a fixed number of rounds that describe its high probability termination bound. In the low probability event that a node remains uncolored, it just proceeds to next step uncolored.

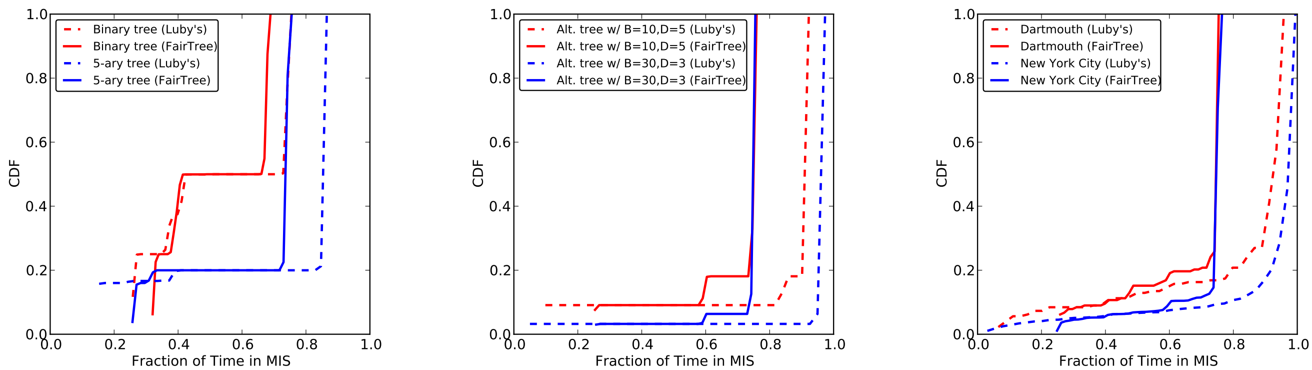

Figure 4. Cumulative distribution of the percentage of runs of Luby's and FAIRTREE in which nodes were in the MIS for *(left)* complete trees, *(center)* alternating trees, and (*right*) real-world trees.

Proof: For conciseness, let P be a shorthand for the function $P_{\mathcal{A},C}$. We first note that if $P(u_i) = 0$ for any $u_i \in V$, then the inequality factor is trivially in $\Omega(n)$ (it would, in fact, be infinite). Assume moving forward, therefore, that all nodes have a join probability strictly greater than 0. We turn our attention to the relationship between u_0 and the k nodes in $S = \{u_{k+1}, ..., u_{2k}\}$. Let $p_S = \sum_{u_i \in S} P(u_i)$. Notice, if a node in S joins the MIS then u_0 must also join the MIS, as the MIS node in S would cover $\{u_1, ..., u_k\}$, requiring u_0 to join to preserve maximality. The reverse argument also holds: if u_0 joins then a node in S must also join. It follows that $P(u_0) = p_S$. Because p_S is the sum of $|S| = k$ elements, there must be some $u^* \in S$ such that $P(u^*) \leq p_S/k$. Pulling together these pieces, it follows that $F_{\mathcal{A}}(C) \geq \frac{P(u_0)}{P(u^*)} \geq \frac{p_S}{(p_S/k)} = k = \Omega(|V|) = \Omega(n)$. ∎

Tree	Tree Size	Algorithm	Ineq. Factor
Complete trees			
Binary tree (Branch=2, Depth=10)	$\lvert V\rvert = 2047$	Luby's	3.07
	$\lvert E\rvert = 2046$	FAIRTREE	2.22
5-ary tree (Branch=5, Depth=5)	$\lvert V\rvert = 3906$	Luby's	6.42
	$\lvert E\rvert = 3905$	FAIRTREE	3.09
Alternating trees			
Branch=10 (for even depths) Depth=5	$\lvert V\rvert = 1221$	Luby's	11.92
	$\lvert E\rvert = 1220$	FAIRTREE	3.15
Branch=30 (for even depths) Depth=3	$\lvert V\rvert = 961$	Luby's	36.59
	$\lvert E\rvert = 960$	FAIRTREE	3.09
Real-world traces			
Dartmouth	$\lvert V\rvert = 178$	Luby's	22.75
	$\lvert E\rvert = 177$	FAIRTREE	3.07
New York City	$\lvert V\rvert = 17834$	Luby's	168.49
	$\lvert E\rvert = 17833$	FAIRTREE	3.25

Table I
INEQUALITY FACTORS FOR REGULAR AND ALTERNATING TREES, AND TREES FORMED FROM REAL-WORLD TRACE DATA.

IX. EVALUATION

This section presents a simulation-based study showing that Luby's algorithm is not fair, and hence a better algorithm such as in this paper is necessary to achieve fairness. This study examines the inequality that results when applying MIS algorithms both to synthetic and real-world topologies.

To conduct our evaluation, we constructed a discrete network simulator that takes as input a tree-structured network topology. Our simulator runs Luby's MIS algorithm and FAIRTREE over the network in synchronous rounds and measures, for each algorithm, (1) the fraction of runs for which each node enters the MIS and (2) the inequality factor; both are computed over 10,000 runs of the randomized protocols. The source code of the simulator and all tested network topology data (described next) are available for download at http://www.cs.georgetown.edu/mis-simulator.

Network topologies. Our simulation study considers three categories of trees: *complete trees* (specifically, binary and 5-ary trees); *alternating trees* where even-depth internal nodes have $c > 1$ children and odd-depth nodes have 1 child; and trees based on *real-world* datasets. The alternating trees are intended to isolate the impact of local degree variations on the performance of Luby's algorithm. The two "real-world" trees, Dartmouth and New York City (NYC), represent simulated networks respectively comprising 700 wireless access points (WAPs) located on Dartmouth College's campus [8] and approximately 18,000 WAPs located in NYC (obtained by querying the Wigle.NET wardriving collection service [18]). The Dartmouth and NYC datasets contain the physical locations (latitudes and longitudes) of WAPs in Hanover, NH and NYC, respectively. We build trees from these datasets by first imposing a maximum physical distance that may be represented by an edge, and then forming a minimum spanning tree over the graph. The parameters of the tested trees are listed in Table I.

Simulation results. Table I summarizes the experimental results, indicating the inequality factor for each algorithm on each network. Luby's algorithm has low inequality factor on low-degree regular trees, which is not surprising as achieving fairness in constant-degree graphs is easy. When turning to the synthetic alternating trees and real-world datasets, however, Luby's inequality factor rises to as high as 168 in a real-world graph, indicating that Luby's is not fair in practice. In contrast, FAIRTREE has an inequality factor of at most 3.25 and is hence fair for all of the experiments; this fact is consistent with Theorem 8.

To break apart the degree of unfairness further, Figure 4 plots the cumulative distribution function (CDF) of the fraction of the time that each node is in the MIS over all 10,000 simulation runs (that is, the distributions of $P_{\mathcal{A},G}(v)$ for all nodes $v \in V$, given an MIS algorithm $\mathcal{A}$ and a tree G). In all plots, the distribution exhibited by FAIRTREE is more compact with no tail extending to low or high probabilities. In contrast, Luby's unfairness is exhibited by a more diffuse distribution. Interestingly, the general shape of the curves is similar for Luby's and FAIRTREE, with the latter being more condensed and hence more fair.

Figure 4 *(center)* plots data for the algorithms on alternating trees, highlighting a case in which Luby's produces high inequality factors. For example, for the alternating tree with branching factor of $B = 10$, Luby's results in approximately 80% of the nodes being in the MIS 90% of the time. Moreover, nearly 10% of the nodes enters the MIS only 10% of the time, and hence the "unfairness" is not isolated to just a few nodes. Perhaps surprisingly, the trees based around real-world datasets, shown in Figure 4 *(right)*, exhibit even more diffuse distributions for Luby's algorithm.

X. CONCLUSION

Distributed MIS algorithms are well-studied from a time complexity perspective. In this paper, we introduce and explore a novel property motivated by both practical and theoretical interests: *fairness*. To improve our understanding of this property, we produce efficient and fair MIS algorithms for many classes of graphs, and show that no MIS algorithm can offer bounded inequality (i.e., $o(n)$) on all graphs. We help motivate the need for these provably fair algorithms by showing, via simulation, that the standard algorithm due to Luby [13] is not fair in many common topologies.

This work opens up many interesting (and likely tractable) questions. Perhaps foremost is a better understanding of when fairness is possible and impossible. It would also be important to understand the fundamental relationship between fairness and time complexity. Does a fair solution, for example, require more rounds than a non-fair solution; i.e., does the relevant MIS lower bound increase from $\Omega(\sqrt{\log n})$ to $\Omega(\log n)$ when imposing a fairness constraint?

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their insightful comments. This work is partially supported by the National Science Foundation through grants CNS-1064986, CNS-1149832, CNS-1204347, CCF-1314633, CCF-1218188 and CCF-1320279, and the Ford Motor Company University Research Program.

REFERENCES

[1] L. Barenboim and M. Elkin. Sublogarithmic distributed mis algorithm for sparse graphs using nash-williams decomposition. In *Proceedings of the ACM Symposium on the Principles of Distributed Computing*, 2008.

[2] L. Barenboim, M. Elkin, S. Pettie, and J. Schneider. The locality of distributed symmetry breaking. In *Proceedings of the IEEE Annual Symposium on Foundations of Computer Science*, 2012.

[3] R. Cole and U. Vishkin. Deterministic coin tossing with applications to optimal parallel list ranking. *Information and Control*, 70(1):32–53, 1986.

[4] S. Daum, S. Gilbert, F. Kuhn, and C. Newport. Leader election in shared spectrum radio network. In *Proceedings of the ACM Symposium on the Principles of Distributed Computing*, 2012.

[5] D. G. Harris, E. Morsy, G. Pandurang, P. Robinson, and A. Srinivasan. Efficient computation of balanced structures. In *Proceedings of the International Colloquium on Automata, Languages and Programming*, 2013.

[6] T. Jurdzinski and D. R. Kowalski. Distributed backbone structure for algorithms in the sinr model of wireless networks. In *Proceeding of International Symposium on Distributed Computing*, 2012.

[7] A. Kayem, S. Akl, and P. Martin. An independent set approach to solving the collaborative attack problem. In *Proceeding Parallel and Distributed Computing and Systems*, 2005.

[8] M. Kim, J. J. Fielding, and D. Kotz. CRAWDAD trace. Downloaded from http://crawdad.cs.dartmouth.edu/dartmouth/wardriving/placelab/aplocations, June 2006.

[9] F. Kuhn and T. Moscibroda. Initializing newly deployed ad hoc and sensor network. In *Proceeding of the ACM SIGMOBILE Annual International Conference on Mobile Computing and Networking*, 2004.

[10] F. Kuhn, T. Moscibroda, and R. Wattenhofer. Local computation: Lower and upper bounds. *CoRR*, abs/1011.5470, 2010.

[11] C. Lenzen and R. Wattenhofer. MIS on trees. In *Proceedings of the ACM Symposium on the Principles of Distributed Computing*, 2011.

[12] N. Linial and M. Saks. Low diameter graph decompositions. *Combinatorica*, 13:441–454, 1993.

[13] M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM Journal on Computing*, 15(4):1036–1055, 1986.

[14] Y. Métivier, J. Robson, N. Saheb-Djahromi, and A. Zemmari. An optimal bit complexity randomized distributed mis algorithm. *Distributed Computing*, 23(5-6):331–340, 2011.

[15] A. Montanvert, P. Meer, and A. Rosenfeld. Hierarchical image analysis using irregular tessellations. *IEEE Transactions on Pattern Analsis and Machine Intelligence*, 13(4):307–316, 1991.

[16] A. Panconesi and A. Srinivasan. On the complexity of distributed network decomposition. *Journal of Algorithms*, 20(2):356–374, 1996.

[17] J. Schneider and R. Wattenhofer. A log-star distributed maximal independent set algorithm for growth-bounded graphs. In *Proceedings of the ACM Symposium on the Principles of Distributed Computing*, 2008.

[18] Wigle.Net wireless geographic logging engine. http://wigle.net/.

[19] D. Yu, Y. Wang, Q.-S. Hua, and F. C. M. Lau. Distributed $(\Delta+1)$-coloring in the physical model. *Algorithms for Sensor Systems*, 7111:146–160, 2012.