Optimal Dynamic Distributed MIS

Keren Censor-Hillel[*] Elad Haramaty[†] Zohar Karnin[‡]

July 17, 2015

Abstract

Finding a maximal independent set (MIS) in a graph is a cornerstone task in distributed computing. The local nature of an MIS allows for fast solutions in a static distributed setting, which are *logarithmic* in the number of nodes or in their degrees [Luby 1986, Ghaffari 2015]. By running a (static) distributed MIS algorithm after a topology change occurs, one can easily obtain a solution with the same complexity also for the *dynamic* distributed model, in which edges or nodes may be inserted or deleted.

In this paper, we take a different approach which exploits locality to the extreme, and show how to update an MIS in a dynamic distributed setting, either *synchronous* or *asynchronous*, with only *a single adjustment*, meaning that a single node changes its output, and in a single round, in expectation. These strong guarantees hold for the *complete fully dynamic* setting: we handle all cases of *insertions* and *deletions*, of *edges* as well as *nodes*, *gracefully* and *abruptly*. This strongly separates the static and dynamic distributed models, as super-constant lower bounds exist for computing an MIS in the former.

We prove that for any deterministic algorithm, there is a topology change that requires n adjustments, thus we also strongly separate deterministic and randomized solutions.

Our results are obtained by a novel analysis of the surprisingly simple solution of carefully simulating the greedy *sequential* MIS algorithm with a random ordering of the nodes. As such, our algorithm has a direct application as a 3-approximation algorithm for correlation clustering. This adds to the important toolbox of distributed graph decompositions, which are widely used as crucial building blocks in distributed computing.

Finally, our algorithm enjoys a useful *history-independence* property, which means that the distribution of the output structure depends only on the current graph, and does not depend on the history of topology changes that constructed that graph. This means that the output cannot be chosen, or even biased, by the adversary, in case its goal is to prevent us from optimizing some objective function. Moreover, history independent algorithms compose nicely, which allows us to obtain history independent coloring and matching algorithms, using standard reductions.

[*]Department of Computer Science, Technion. Supported in part by ISF grant 1696/14. ckeren@cs.technion.ac.il.

[†]College of Computer and Information Science, Northeastern University. Supported in part by NSF grant CCF-1319206. Part of this work was done while the author was at Yahoo Labs. eladh@cs.technion.ac.il.

[‡]Yahoo Labs. zkarnin@ymail.com.

1 Introduction

Dynamic environments are very common in distributed settings, where nodes may occasionally join and leave the network, and communication links may fail and be restored. This makes solving tasks in a dynamic distributed setting of fundamental interest: Indeed, it is a widely studied area of research, especially in the context of asynchronous self-stabilization [17, 18, 26, 50], and also in the context of severe graph changes, whether arbitrary [36] or evolving randomly [3]. In this paper, we consider a dynamic distributed setting which is *synchronous* and assumes topology changes with sufficient time for recovery in between, as is typically assumed in the literature on *sequential dynamic algorithms* [15].

Solutions for problems from the static distributed setting translate nicely into our dynamic distributed setting, by running them in response to topology changes, in order to adjust the output [5, 6, 40]. This can be quite efficient especially for *local* problems, such as finding a maximal independent set (MIS) in the network graph. The cornerstone MIS problem admits fast distributed solutions whose complexities are logarithmic in the size of the graph or in the degrees of the nodes [2, 25, 31, 43].

In this paper, we exploit locality to the extreme, and present an MIS algorithm for the dynamic distributed setting, both *synchronous* or *asynchronous*, which requires only *a single adjustment*, where the adjustment measure of an algorithm is the number of nodes that need to change their output in response to the topology change, and a single round, in expectation. These strong guarantees hold for the *complete fully dynamic* setting, i.e., we handle all cases of *insertions* and *deletions*, of *edges* as well as *nodes*, *gracefully* and *abruptly*.[1] This is a strong separation between the static and dynamic distributed models, as super-constant lower bounds exist for the static setting [37, 42]. We further prove that for any deterministic algorithm, there is a topology change that requires n adjustments, with n being the number of nodes, thus we also strongly separate deterministic and randomized solutions. Below, we overview our technique and the applications of our result.

1.1 Our Contribution

Our approach is surprisingly simple: We simulate the greedy *sequential* algorithm for solving MIS. The greedy sequential algorithm orders the nodes and then inspects them by increasing order. A node is added to the MIS if and only if it does not have a lower-order neighbor already in the MIS. We consider *random greedy*, the variant of greedy in which the order is chosen uniformly at random. Consider simulating random greedy in a dynamic environment with the following template (ignoring the model of computation/communication for the moment). Each node needs to maintain the invariant that its state depends only on the states of its neighbors with lower order, such that it is in the MIS if and only if none of its lower order neighbors are in the MIS. When a change occurs in the graph, nodes may need to change their output, perhaps more than once, until they form a new MIS. Our key technical contribution is in proving:

Theorem 1 For any *arbitrary* change in the graph, the expectation over all random orders, of the number of nodes that change their output in the above random greedy template is at most 1.

The Challenge: We denote by π the random order of nodes, we denote by v^* the only node (if any) for which the above invariant does not hold after the topology change, and we denote by S the set of nodes that need to be changed in order for the invariant to hold again at all nodes. We look at S', the set of nodes that would have needed to be changed if the order was as in π, except for pushing v^* to be the first node in that order. The definition of S' does not depend on the real order of v^* in π. Therefore, we can prove that S can either be equal to S' if the order of v^* in π is minimal in S', and empty otherwise. Now the

[1] See definitions of topology changes in Section 2.

question is, what is the probability, given S', that v^* is indeed its minimal order node? The answer is that if S' were deterministic, i.e. independent of π, the probability would be $1/|S'|$. However, S' is a random set and having knowledge of its members restricts π to be non-uniform, which in turn requires a careful analysis of the required probability. To overcome this issue, we prove that the information that S' gives about π is either about the order between nodes not in S', or about the order within $S' \backslash \{v^*\}$, which both do not effect the probability that v^* is the minimal in S'.

Distributed Implementation: This powerful statement of $\mathbb{E}[|S|] \leq 1$ directly implies that a *single* adjustment is sufficient for maintaining an MIS in a dynamic setting. A direct distributed implementation of our template implies that in expectation also a single round is sufficient. This applies both to the synchronous and asynchronous models, where the number of rounds in the asynchronous model is defined as the longest path of communication.

Obtaining $O(1)$ Broadcasts and Bits: In fact, in the synchronous model, it is possible to obtain an expected number of $O(1)$ *broadcasts* and *bits*. Here the number of broadcasts is the total number of times, over all nodes, that any node sends a $O(\log n)$-bit broadcast message (to all of its neighbors)[2]. Moreover, since we only need a node to know the order between itself and its neighbors, using a similar technique to that of [45], we can obtain that in expectation, a node only needs to send a constant number of bits in each broadcast. The above holds for edge insertions and deletions, graceful node deletion, and node *unmuting*, while for an abrupt deletion of a node v^* we will need $O\left(\min\{\log(n), d(v^*)\}\right)$ broadcasts, and for an insertion of a node v^* we will need $O(d(v^*))$ broadcasts, in expectation.

This is done with a careful dynamic distributed implementation which guarantees that each node that changes its output does so at most $O(1)$ times, as opposed to the direct distributed implementation [3]. Hence, obtaining these broadcast and bit complexities comes at a cost of increasing the round complexity, but it remains constant (albeit not 1). In what follow, we focus on this result since we find it intriguing that we can get as little as $O(1)$ *total communication*, while paying $O(1)$ rounds instead of a single round is arguably not a big cost.

Matching Lower Bounds: We claim that any deterministic algorithm requires n adjustments, which can be seen through the following example. Let A be a dynamic deterministic MIS algorithm. Let G_0 be the complete bipartite graph over two sets of nodes of size k. We denote by L the side of G_0 that is chosen to be the MIS by A, and we denote the other side by R. For every $i \in [k]$ let G_i be the graph obtained after deleting i nodes from L, and consider executing A on $G_0, G_1, ..., G_k$. For every i, since G_i is a complete bipartite graph, one of the sides has to be the MIS. Since G_k contains only disconnected nodes of R then R is the only MIS of G_k. This implies that after some specific change along the sequence, the side of the MIS changes from L to R. In this topology change, *all* of the nodes change their output.

This gives a strong separation between our result and deterministic algorithms. Moreover, it shows that (1) the expected adjustment complexity of any algorithm must be at least 1, as we have a sequence of k topology changes that lead to at least k adjustments, and (2) it is impossible to achieve high probability bounds that improve upon a simple Markov bound. Specifically, this explains why we obtain our result in expectation, rather than with high probability. This is because the example can be inserted into any larger graph on n nodes, showing that *for every* value of k, there exists an instance for which at least $\Omega(k)$ adjustments are needed with probability at least $1/k$.

Approximate Correlation Clustering: In addition to the optimal complexity guarantees, the fact that our algorithm simulates the random greedy sequential algorithm has a significant application to correlation

[2]We emphasize that the term broadcast is used here to indicate the more restricted setting of not being able to send different messages to different neighbors in the same round. It does not refer to a wireless setting of communication.

[3]This bears some similarity to the method in [53], where the number of *moves* is reduced in an MIS self-stabilizing algorithm by adding a possible *wait* state to the standard *in MIS* and *not in MIS* states.

clustering. Correlation clustering requires the nodes to be partitioned into clusters in a way that minimizes the sum of the number of edges outside clusters and the number of non neighboring pairs of nodes within clusters (that is, missing edges within clusters). Ailon et al. [1] show that random greedy obtains a 3-approximation for correlation clustering[4], by having each MIS node inducing a cluster, and each node not in the MIS belonging to the cluster induced by the smallest random ID among its MIS neighbors. This directly translates to our model, by having the nodes know that random ID of their neighbors. Graph decompositions play a vital role in distributed computing (see, e.g., [47]), and hence the importance of obtaining a 3-approximation for correlation clustering.

History Independence: Finally, our algorithm has a useful property, which we call *history independence*, which means that the structure output by the algorithm (e.g., the MIS) depends only on the current graph, and does not depend on the history of topology changes. This means that the output cannot be chosen, or even biased, by the adversary, in case its goal is to prevent us from optimizing some objective function. Moreover, history independent algorithms compose nicely, which allows us to obtain history independent coloring and matching algorithms, using standard reductions.

1.2 Related Work

Distributed MIS: Finding an MIS is a central theme in the classical distributed setting. The classic algorithms [2, 31, 43] complete within $O(\log n)$ rounds, with high probability. More recently, a beautiful line of work reduced the round complexity to depend on Δ, the maximal degree in the graph. These include the $O(\Delta + \log^* n)$-round algorithm of [9], the $O(\log \Delta \sqrt{\log n})$-round algorithm of [10], and the very recent $O(\log \Delta) + 2^{O(\sqrt{\log \log n})}$-round algorithm of [25]. An excellent source for additional background and relations to coloring can be found in [8].

Distributed dynamic MIS: The problem of finding a fast dynamic distributed MIS algorithm appears as an open problem in [21], which studies the problem of maintaining a sparse spanner in this setting. Additional problems in this setting are also addressed in [11], and in slightly different settings in [4, 14, 32, 35]. However, we are unaware of any other work in this setting about maintaining an MIS. One standard approach for maintaining an MIS is running distributed algorithms that are designed for the static setting. This can be done for any distributed algorithm, sometimes using a corresponding *compiler*, e.g., when applied to an asynchronous dynamic setting [5, 6, 40]. One important exception is the solution in [34], which as in our algorithm, requires a constant number of rounds, but as opposed to our algorithm, makes the strong assumptions that (1) a node gracefully departs the network, and (2) messages may have unbounded size. An additional difference is that the number of broadcasts, as opposed to the number of rounds, may be large.

Additional distributed dynamic algorithms: A huge amount of literature is devoted to devising different algorithms in a self-stabilizing setting (see, e.g., [18, 26, 50] and references therein). This setting is inherently different from ours since it measures the time it takes an algorithm to reach a correct output starting from *any* arbitrary configuration. The setting is asynchronous, but considers a notion of time that is different than ours, where an asynchronous round requires that each node communicates with all of its neighbors. This inherently implies a lot of communication (broadcasts).

An MIS-based clustering algorithm for the asynchronous model that appeared in [20] also uses a random node order for recovering after a change. However, their self-stabilizing setting differs from ours in several aspects, such as assuming a bounded degree graph and discussing corrupted states of multiple nodes, and multiple topology changes. In addition, our techniques and analysis are completely different. In particu-

[4]In the same paper, they also provide a 2.5 approximation based on rounding a solution of a linear program. We do not elaborate on the details of this algorithm, nor the history of the correlation clustering problem as it is outside the scope of our paper.

lar, the clustering obtained there may not be an approximation to correlation clustering. Furthermore, the number of rounds required by [20] is $O(\log(n))$ as opposed to the single round algorithm (in expectation) presented here.

Related, but not identical, notions of error confinement, fault local and fault amendable algorithms have been studied in [7, 38, 39], where the internal memory of a node may change. Another property that self-stabilizing algorithms should aim for is super-stabilization [19], which means that they are self-stabilizing (eventually output the required structure) and also recover extremely fast from *a single* topology change. Super-stabilization requires also a small adjustment measure, which is the maximum number of nodes that have to change their output. Our MIS algorithm recovers from a single topology change in a single round, and has an adjustment measure of exactly 1, in expectation.

Simulating the sequential greedy algorithm: Simulating random greedy has been used before in order to obtain fast solutions for sequential local computation algorithms (LCA). In this setting, the algorithm does not have access to the entire graph, but rather an oracle access using queries about nodes or edges, and needs to provide an approximate solution for various problems, among which are the problems considered in this paper. We emphasize that the models are inherently different, and hence is our technical analysis. While we bound the size of the set of nodes that may change their output after a topology change, studies in the local computation literature [41, 44, 46, 54] bound the size of the set of nodes that need to be recursively queried in order to answer a *random* node query. In some sense, these sets are opposite: We begin with a single node v changing its state due to a topology change, and look at the set of nodes that change their state *due* to the change of v. Local computation algorithms begin with a single node v and look at the set of nodes whose states *determine* the state of v.

2 Dynamic Distributed Computations

The distributed setup is a standard message passing model. The network is modeled by an undirected graph $G = (V, E)$ where the node set is V, and E consists of the node pairs that have the ability to directly communicate. We assume a broadcast setting where a message sent by a node is heard by all of its neighbors. Also, we assume a synchronous communication model, where time is divided into rounds and in each round any willing node can broadcast a message to its neighbors. We restrict the size of each message to be $O(\log(n))$ bits, with $n = |V|$ being the size of the network[5]. The computational task is to maintain a graph structure, such as a maximal independent set (MIS) or a node clustering. That is, each node has an output, such that the set of outputs defines the required structure.

Our focus is on a dynamic network, where the graph changes over time. As a result, nodes may need to communicate in order to adjust their outputs. The system is *stable* is when the structure defined by the outputs satisfies the problem requirements.

A graph topology change can be with respect to either an edge or a node. In both cases we address both deletions and insertions, both of which are further split into two different types. For deletions we discuss both a *graceful deletion* and an *abrupt deletion*. In the former, the deleted node (edge) may be used for passing messages between its neighbors (endpoints), and retires completely only once the system is stable again. In the latter, the neighbors of the deleted node simply discover that the node (edge) has retired but it cannot be used for communication. For insertions, we distinguish between a *new node insertion* and an *unmuting* of a previously existing node. In the former, a new node is inserted to the graph, possibly with multiple edges. In the latter, a node that was previously invisible to its neighbors but heard their communication, becomes

[5]This is the standard assumption in a distributed setting. In our dynamic setting where the size of the graph may change we assume knowledge of some upper bound $N \geq n$, with $N = n^{O(1)}$, and restrict the message length to $O(\log(N)) = O(\log(n))$.

visible and enters the graph topology[6].

We assume that the changes are infrequent so that they occur in large enough time gaps, so that the system is always stable before a change occurs. We consider the performance of an algorithm according to three complexity measures. The first is the *adjustment-complexity*, measuring the number of nodes that change their output as a result of the recent topology change. The second is the *round-complexity*, which is the number of rounds required for the system to become stable. Finally, the third, more harsh, score is the *broadcast-complexity*, measuring the total number of broadcasts.

Our algorithms are randomized and thus our results apply to the expected values of the above measures, where the expectation is taken over the randomness of the nodes. We emphasize that this is the only randomness discussed; specifically, the result is not for a random node in the graph nor a random sequence of changes, but rather applies to any node and any sequence of changes. It holds for *every* change in the graph, not only in amortized over all changes. For this, we make the standard assumption of an *oblivious* non-adaptive adversary. This means that the topology changes do not depend on the randomness of the algorithm. This standard assumption in dynamic settings is natural also for our setting, as, for example, an adaptive adversary can always choose to delete MIS nodes and thereby force worst-case behavior in terms of the number of adjustments.

In what follows we discuss the problem of computing an *MIS*. Here, the outputs of the nodes define a set M, where any two nodes in M are not connected by an edge, and any node not in M has a neighbor in M. The second problem we discuss is that of *correlation clustering*. Here, the objective is to find a partitioning $\mathcal{C}$ of the node set V, where we favor partitions with a small number of "contradicting edges". That is, we aim to minimize the sum $\sum_{C\in\mathcal{C}}\sum_{u,v\in C}\mathbb{1}_{[(u,v)\notin E]}+\sum_{C_1\neq C_2\in\mathcal{C}}\sum_{u\in C_1,v\in C_2}\mathbb{1}_{[(u,v)\in E]}$.

3 A Template for Maintaining a Maximal Independent Set

In this section we describe a template for maintaining a maximal independent set (MIS). Initially, we are given a graph $G=(V,E)$ along with an MIS that satisfies certain properties, and after a topology change occurs in the graph, applying the template results in an MIS that satisfies the same properties. That is, the template describes what we do after a single topology change, and if one considers a long-lived process of topology changes, then this would correspond to having initially an empty graph and maintaining an MIS as it evolves. We emphasize that the template describes a process that is not in any particular model of computation, and later in Section 4 we show how to implement it efficiently in our dynamic distributed setting. This also means that there are only four topology changes we need to consider: edge-insertion, edge-deletion, node-insertion and node-deletion. For example, the notions of abrupt and graceful node deletions are defined with respect to the dynamic distributed setting because they affect communication, and therefore the implementation of the template will have to address this distinction, but the template itself is only concerned with a single type of node deletion, not in any particular computation model.

Throughout, we assume a uniformly random permutation π on the nodes $v\in V$. We define two *states* in which each node can be: M for an MIS node, and $\bar{M}$ for a non-MIS node. We abuse notations and also denote by M and $\bar{M}$ the sets of all MIS and non-MIS nodes, respectively. Our goal is to maintain the following *MIS invariant*: A node v is in M if and only if all of its neighbors $u\in N(v)$ which are ordered before it according to π, i.e., for which $\pi(u)<\pi(v)$, are not in M. It is easy to verify that whenever the MIS invariant is satisfied, it holds that the set M is a maximal independent set in G. Furthermore, it is easy to verify that this invariant simulates the greedy sequential algorithm, as defined in the introduction.

When any of the four topology changes occurs, there is at most a single node for which the MIS invariant

[6]The distinction is only relevant for nodes insertions, as there is no knowledge associated with an edge.

no longer holds. We denote this node by $v^* = v^*(G_{\rm old}, G_{\rm new}, \pi)$, where $G_{\rm old}$ and $G_{\rm new}$ are the graphs before and after the topology change. For an edge insertion or deletion, v^* is the endpoint with the larger order according to π. For a node insertion or deletion, v^* is the node. [7] In case the topology change is an edge change, we will need also to take into consideration its other endpoint. We denote it by $v^{**} = v^{**}(G_{\rm old}, G_{\rm new}, \pi)$, and notice that by our notation, it must be the case that $\pi(v^{**}) < \pi(v^*)$. In order to unify our proofs for all of the four possible topology changes, we talk about a node v^{**} also for node changes. In this case we define v^{**} to be v^* itself, and we have that $\pi(v^{**}) = \pi(v^*)$. Therefore, for any topology change, it holds that $\pi(v^{**}) \le \pi(v^*)$.

To describe our template, consider the case where a new edge is inserted and it connects two nodes $\pi(v^{**}) < \pi(v^*)$, where both nodes are in M. As a result, v^* must now be deleted from the MIS and hence we need to change its state. Notice that as a result of the change in the state of v^*, additional nodes may need their state to be changed, causing multiple state changes in the graph. An important observation is that it is possible that during this process of propagating local corrections of the MIS invariant, we change the state of a node more than once. As a simple example, consider the case in which v^* has two neighbors, u_1 and u_2, for which $\pi(v^*) < \pi(u_1), \pi(u_2)$, and that u_1 and u_2 are connected by a path (u_1, w_1, w_2, u_2), with $\pi(u_1) < \pi(w_1) < \pi(w_2) < \pi(u_2)$. Now, when we change the state of v^* to $\bar{M}$, both u_1 and u_2 need to be changed to M, for the MIS invariant to hold. This implies that w_1 needs to be changed to $\bar{M}$ and w_2 needs to be changed to M. In this case, since $\pi(w_2) < \pi(u_2)$, the node u_2 needs to be changed back to state $\bar{M}$.

The above observation leads us to define a set of *influenced* nodes, denoted by $S = S(G_{\rm old}, G_{\rm new}, \pi)$, containing v^* in the scenario where we need to change its state, and all other nodes whose state we must subsequently change as a result of the state change of v^*. To formally define the set S we introduce some notations. The notations rely on the graph structure of $G_{\rm new}$ unless the change is a node deletion in which case the rely on $G_{\rm old}$. For each node u, we define $I_\pi(u) = \{v \in N(u) \mid \pi(v) < \pi(u)\}$, the set of neighbors of u that are ordered before it according to π. These are the nodes that can potentially *influence* the state of u according to the MIS invariant. The definition of S is recursive, according to the ordering induced by π. If immediately after the topology change, in the new graph G with the order π it holds that the MIS invariant still holds for v^*, then we define $S = \emptyset$. (This is motivated by the fact that no node is influenced by this change.) Otherwise, we denote $S_0 = \{v^*\}$, and inductively define

$$S_i = \{u \mid u \in M, \text{ and } S_{i-1} \cap I_\pi(u) \neq \emptyset\} \cup \{u \mid u \in \bar{M}, \text{ and every } v \in I_\pi(u) \cap M \text{ is in } \cup_{j=0}^{i-1} S_j)\}. \quad (1)$$

The set S is then defined as $S = \bigcup_i S_i$. Notice that a node u can be in more than one set S_i, as is the case for u_2 in the example above, which is in both S_1 and S_4. The impact of a node u being in more than one S_i is that in order to maintain the MIS invariant, we need to make sure that we update the state of u after we update that of w, for any w such that $w \in I_\pi(u)$. Instead of updating the state of u twice, we can simply wait and update it only after the state of every such w is updated. For this, we denote by $i_u = \max\{i \mid u \in S_i\}$ the maximal index i for which u is in S_i.

We formally describe our template in Algorithm 1. By construction, the updated states after executing Algorithm 1 satisfy the MIS invariant. In addition, the crucial property that is satisfied by the above template is that in expectation, the size of the set S is 1. The remainder of this section is devoted to proving the following, which is our main technical result.

Theorem 1. *For every two graphs $G_{\rm old}$ and $G_{\rm new}$ that differ only by a single edge or a single node, it holds that $\mathbb{E}_\pi\left[|S(G_{\rm old}, G_{\rm new}, \pi)|\right] \le 1$.*

[7]For a node deletion, we slightly abuse the definition of v^* in order to facilitate the presentation, and consider it to be the deleted node. This means that here we consider an intermediate stage of having v^* still belong to the graph w.r.t. the MIS invariant of all the other nodes, but for v^* the MIS invariant no longer holds. This is in order to unify the four cases, otherwise we would have to consider all of the neighbors of a deleted node as nodes for which the MIS invariant no longer holds after the topology change.

Algorithm 1 A Template for Dynamic Correlation Clustering.

```
Initially, $G = (V, E)$ satisfies the MIS invariant.
On topology change at node $v^*$ do:
1. Update state of $v^*$ if required for MIS to hold
2. For $i \leftarrow 1$, until $S_i = \emptyset$, do:
3.     For every $u \in S_i$ such that $i = i_u$:
4.         Update state of $u$
5.     $i \leftarrow i + 1$
```

Outline of the proof: In order to prove that $\mathbb{E}[|S|] \leq 1$, instead of analyzing the set S directly, we analyze the set $S' = S'(G_{\text{old}}, G_{\text{new}}, \pi, v^*)$, which is defined via recursion similarly to S with three modifications: (1) It is always the case that $S'_0 = \{v^*\}$ (2) The graph according to which S' is defined is G_{old} in the case of a node deletion or an edge insertion, and G_{new} otherwise. (3) The permutation according to S' is defined as π', that is identical to π other than its value for v^* that is forced to be the minimal among all other π values. Notice that S' does not depend on $\pi(v^*)$ and in particular, having knowledge about its elements does not give any information as to whether $\pi(v^*) < \pi(v^{**})$ or vice versa.

In Lemma 2, we prove that if $\pi(v^*) \neq \min\{\pi(u) \mid u \in S'\}$ then $S = \emptyset$, and otherwise $S = S'$ (in fact, it would be enough that $S \subseteq S'$). Then, in Lemma 3, we prove that for any set $P \subseteq V$, given the event that $P = S'$, the probability, over the random choice of π, that $\pi(v^*) = \min\{\pi(u) \mid u \in P\}$ is $1/|P|$. This leads to the required result of Theorem 1. Lemma 3 would be trivial if there was no correlation between π and S'. However, the trap we must avoid here is that S' is defined according to π, and therefore when analyzing its size we cannot treat π as a *uniformly* random permutation. To see why, suppose we know that inside $S' \setminus \{v^*\}$ we have nodes with large order in π. Then the probability that the order of v^* in π is smaller than all nodes in $S' \setminus \{v^*\}$, is much larger than $1/|S'|$, and can in fact be as large as $1 - o(1)$. In other words, S' gives some information over π. Nevertheless, we show that this information is either about the order between nodes outside of S', or about the order between nodes within $S' \setminus \{v^*\}$. Both types of restrictions on π do not affect the probability that v^* is the minimal of S'.

We now formally prove our result as outlined above. Throughout we use the notation $u \in M$ or $u \in \bar{M}$. This applies only to nodes u for which we are guaranteed that their states remain the same despite the topology change.

Lemma 2. *If $\pi(v^*) \neq \min\{\pi(u) \mid u \in S'\}$ then $S = \emptyset$. Otherwise, $S \subseteq S'$.*

Proof. First, assume that $\pi(v^*) \neq \min\{\pi(u) \mid u \in S'\}$. We show that the MIS invariant still holds after the topology change, and so $S = \emptyset$. Consider the node w, for which $\pi(w) = \min\{\pi(u) \mid u \in S'\}$. Notice that $w \notin S$, because $\pi(w) < \pi(v^*)$. We claim that $w \in M$. Assume, towards a contradiction, that $w \in \bar{M}$. This implies that w has a neighbor $u \in M$ such that $\pi(u) < \pi(w)$. For this node u we must have $u \notin S'$ due to the minimality of $\pi(w)$. It follows, according to the construction of S' that w cannot be an element of S', leading to a contradiction.

We have that $w \in M$ and due to the minimality of $\pi(w)$, it must be that $w \in S'_1$, which implies that w is a neighbor of v^*. But then, when considering S, v^* has a neighbor other than v^{**} which is ordered before it according to π which is in M. In the case of an edge insertion or deletion, this means that v^* remains in $\bar{M}$ despite the topology change meaning that $S = \emptyset$. In the case of a node deletion, v^* was not in M prior to the change hence $S = \emptyset$. In the case of a node insertion, v^* does not enter M hence again, $S = \emptyset$.

Next, assume that $\pi(v^*) = \min\{\pi(u) \mid u \in S'\}$. We show that either $S = \emptyset$ or $S = S'$. If there is no need to change the state of v^* as a result of the topological change then $S_0 = \emptyset$, and so $S = \emptyset$ and the claim holds. It remains to analyze the case where $S_0 = S'_0 = \{v^*\}$. If $u \in S'_1$ then $\pi(v^*) < \pi(u)$ hence according to its definition $u \in S_1$. If $u \notin S'_1$ then u must have a neighbor $w \in M$ with $\pi(w) < \pi(u)$ meaning that $u \notin S_1$. We have that $S_1 = S'_1$ and similarly $S_i = S'_i$ for all $i > 1$. We conclude that $S' = S$ as required. $\square$

The following lemma shows that the probability of having $S = S'$ is $1/|S'|$, which immediately lead to Theorem 1 as the only other alternative is $S = \emptyset$.

Lemma 3. *For any set of nodes $P \subseteq V$, it holds that*

$$\Pr\left[\pi(v^*) = \min\{\pi(u) \mid u \in P\} \mid S' = P \textit{ and } \pi(v^{**}) \leq \pi(v^*)\right] = \frac{1}{|P|}.$$

To prove this lemma we focus on S'. Notice that the events we considered in the previous lemma depend only on the ordering implied by π and hold for any configuration of states for the nodes that satisfy the MIS invariant. Roughly speaking, the lemma will follow from the fact the the event $S' = P$ does not give any information about the order implied by π between nodes in P and nodes in $V \backslash P$. To this end, for every permutation τ on V, we define $S'(\tau) = S'(G_{\text{old}}, G_{\text{new}}, \tau, v^*)$ as the set corresponding to S' under the ordering induced by τ. We denote by Π_P the set of all permutations τ for which it holds that $S'(\tau) = P$. We first need to establish the following about permutations in Π_P: If π and σ are two permutations on V such that $\pi|_P = \sigma|_P$ and $\pi|_{V\backslash P} = \sigma|_{V\backslash P}$, then $\sigma \in \Pi_P$ if and only if $\pi \in \Pi_P$.

Claim 4. *Let $P \subseteq V$ be a set of nodes, and let π and σ be two permutations such that $\pi|_P = \sigma|_P$ and $\pi|_{V\backslash P} = \sigma|_{V\backslash P}$. Assume $\pi \in \Pi_P$. We have that $V\backslash P \subseteq V \setminus S'(\sigma)$ and every $u \in V\backslash P$ has the same state according to π and σ.*

Proof. Let $u \in V\backslash P$. We prove that $u \in V \setminus S'(\sigma)$ and that its state under σ is the same as it is under π by induction on the order of nodes in $V\backslash P$ according to π (which is equal to their order according to σ).

For the base case, assume that u has the minimal order in $V\backslash P$. We claim that u cannot have a neighbor in P. Assume, towards a contradiction, that u has a neighbor $w \in P$. Since $w \in P$ then it is possible that after the a w will be in M. Since two nodes in M cannot be neighbors and $u \notin P$, then u must be in $\bar{M}$ according to π. In this case there is a node $z \in I_\pi(u) \cap V\backslash P$ that is in M according to π. But this cannot occur due to the minimality of $\pi(u)$ in $V\backslash P$. Therefore, u has no neighbors in P as required.

We have that all of the neighbors of u are in $V\backslash P$ and that u is the minimal among its neighbors according to π. Since $\pi|_{V\backslash P} = \sigma|_{V\backslash P}$ we have that u has the minimal order among its neighbors according to σ. This translates into u having a state of M under σ and in particular, u is not an element of $S'(\sigma)$, thus proving our base case.

For the induction step, consider a node $u \in V\backslash P$, and assume the claim holds for every $w \in V\backslash P \cap I_\pi(u)$. We consider two cases, depending on whether u has a neighbor in P or not.

Case 1: u does not have any neighbor in P. If $u \in \bar{M}$, then there is a node $z \in I_\pi(u) \cap V\backslash P$ that is in M according to π. By the induction hypothesis, $z \in V \setminus S'(\sigma)$ and $z \in M$ also according to σ. Since $\pi|_{V\backslash P} = \sigma|_{V\backslash P}$, we have that u is in $\bar{M}$ according to σ too. Otherwise, if $u \in M$, then every $w \in I_\pi(u)$ (which is also $V\backslash P$) is in $\bar{M}$ according to π. Any node $w \in I_\sigma(u)$ is also in $w \in I_\pi(u)$, since it is not in P and $\pi|_{V\backslash P} = \sigma|_{V\backslash P}$. The induction hypothesis on w gives that it is also in $V \setminus S'(\sigma)$ (otherwise it would be in $S'_\pi = P$ in contradiction to the assumption of case 1), and its state according to σ is $\bar{M}$. Hence, u must be in $V \setminus S'(\sigma)$ as well, and in state M according to σ.

Case 2: Assume that u has a neighbor $w \in P$. Since $w \in P$ then it is possible that after the algorithm w will be in M. Since two nodes in M cannot be neighbors and $u \notin P$, then u must be in $\bar{M}$ according to π. In this case there is a node $z \in I_\pi(u)\backslash P$ that is in M according to π. By the induction hypothesis, $z \in V \setminus S'(\sigma)$ and $z \in M$ also according to σ. Since $\pi|_{V\backslash P} = \sigma|_{V\backslash P}$, we have that u is in $\bar{M}$ and in $V \setminus S'(\sigma)$ according to σ too. $\square$

Claim 5. *Let $P \subseteq V$ be a set of nodes, and let π and σ be two permutations such that $\pi|_P = \sigma|_P$ and $\pi|_{V\setminus P} = \sigma|_{V\setminus P}$. Assume $\pi \in \Pi_P$. We have that $P \subseteq S'(\sigma)$.*

Proof. We prove that every node $u \in P$ is also in $S'(\sigma)$ by induction on the order of nodes in P according to π (which is equal to their order according to σ), with the modification forcing v^* to be the first among the nodes of P. The base case is for v^*, which is clearly in both sets $S'(\pi)$ and $S'(\sigma)$. Consider a node $u \in P$ and assume that the claim holds for every node in P which is ordered before u according to π. Since $u \in P$ and $u \neq v^*$ there must be some $w \in I_\pi(u) \cap P$ since $\pi|_P = \sigma|_P$ and $u \in P$ we have according to our induction hypothesis that $w \in S'(\sigma)$, meaning that $I_\sigma(u) \cap S'(\sigma)$ is non-empty.

Consider now an arbitrary $w \in I_\sigma(u)$. If $w \in P$ then since $\pi|_P = \sigma|_P$ and $u \in P$ we have according to our induction hypothesis that $w \in S'(\sigma)$. If $w \notin P$ then it must be the case that $w \in \bar{M}$ according to π, otherwise u cannot be in P. We thus have according to Claim 4 that (1) $w \in V \setminus S'(\sigma)$ and (2) $w \in \bar{M}$ according to σ. It follows that all neighbors of u in $I_\sigma(u)$ are either in $S'(\sigma)$ or in $\bar{M}$ according to σ, hence since $I_\sigma(u) \cap S'(\sigma) \neq \emptyset$ it must be the case that $u \in S'(\sigma)$. $\square$

Claims 4 and 5 combined imply that if $\pi|_P = \sigma|_P$ and $\pi|_{V\setminus P} = \sigma|_{V\setminus P}$ then $\sigma \in \Pi_P$ if and only if $\pi \in \Pi_P$. We are now ready for the proof of Lemma 3.

Proof. (of Lemma 3) Given two permutations σ^+ and σ^- on $P\setminus\{v^*\}$ and $V\setminus P$, respectively, we define ρ_{σ^+,σ^-} as $\rho_{\sigma^+,\sigma^-} = \Pr\left[\forall u \in P, \pi(v^*) \leq \pi(u) \mid \pi|_{P\setminus\{v^*\}} = \sigma^+ \text{ and } \pi|_{V\setminus P} = \sigma^-\right]$.

First, we observe that for two pairs of permutations σ_1^+, σ_1^- and σ_2^+, σ_2^- as above, it holds that $\rho_{\sigma_1^+,\sigma_1^-} = \rho_{\sigma_2^+,\sigma_2^-}$. This is because given the condition for σ_1^+, σ_1^-, applying the permutation $(\sigma_1^+)^{-1}\sigma_2^+$ to nodes in $P\setminus\{v^*\}$ and applying the permutation $(\sigma_1^-)^{-1}\sigma_2^-$ to nodes in $V\setminus P$ has no affect on whether the event $\forall u \in P, \pi(v^*) \leq \pi(u)$ holds. Next, since $\Pr\left[\forall u \in P, \pi(v^*) \leq \pi(u)\right] = \frac{1}{|P|}$, we have that for any pair of permutations σ^+, σ^- on $P\setminus\{v^*\}$ and $V\setminus P$, respectively:

$$\begin{aligned}\frac{1}{|P|} &= \Pr\left[\forall u \in P, \pi(v^*) \leq \pi(u)\right] = \sum_{\tau^+,\tau^-} \rho_{\tau^+,\tau^-} \Pr\left[\pi|_{P\setminus\{v^*\}} = \tau^+ \text{ and } \pi|_{V\setminus P} = \tau^-\right] \\ &= \sum_{\tau^+,\tau^-} \rho_{\sigma^+,\sigma^-} \Pr\left[\pi|_{P\setminus\{v^*\}} = \tau^+ \text{ and } \pi|_{V\setminus P} = \tau^-\right] = \rho_{\sigma^+,\sigma^-}.\end{aligned}$$

Finally, Claims 4 and 5 imply that for every set $P \subseteq V$ there is a set of $t = t_P$ pairs of permutations $\{(\sigma_1^+, \sigma_1^-), \ldots, (\sigma_t^+, \sigma_t^-)\}$ on $P\setminus\{v^*\}$ and $V\setminus P$, respectively, such that $\Pi_P = \{\pi \mid \exists i, \pi|_{P\setminus\{v^*\}} = \sigma_i^+ \text{ and } \pi|_{V\setminus P} = \sigma_i^-\}$. We conclude that for a given set $P \subseteq V$:

$$\begin{aligned}\Pr_{\pi\in\Pi_P}\left[\forall u \in P, \pi(v^*) \leq \pi(u)\right] &= \sum_{i=1}^{t} \rho_{\sigma_i^+,\sigma_i^-} \Pr\left[\pi|_{P\setminus\{v^*\}} = \sigma_i^+ \text{ and } \pi|_{V\setminus P} = \sigma_i^- \mid \pi \in \Pi_P\right] \\ &= \frac{1}{|P|}\sum_{i=1}^{t} \Pr\left[\pi|_{P\setminus\{v^*\}} = \sigma_i^+ \text{ and } \pi|_{V\setminus P} = \sigma_i^- \mid \pi \in \Pi_P\right] = \frac{1}{|P|}. \qquad (2)\end{aligned}$$

To complete the proof, we argue that knowing that $\pi(v^{**}) \leq \pi(v^*)$ can only decrease the probability that $\pi(v^*) \leq \pi(u)$ for all $u \in P$. Formally,

$$\begin{aligned}&\Pr_{\pi\in\Pi_P}\left[\forall u \in P, \pi(v^*) \leq \pi(u) \mid \pi(v^{**}) \leq \pi(v^*)\right] \\ =\ &\Pr_{\pi\in\Pi_P}\left[\forall u \in P, \pi(v^*) \leq \pi(u) \text{ and } \pi(v^{**}) \leq \pi(u) \mid \pi(v^{**}) \leq \pi(v^*)\right] \\ =\ &\frac{\Pr_{\pi\in\Pi_P}\left[\forall u \in P, \pi(v^*) \leq \pi(u) \text{ and } \pi(v^{**}) \leq \pi(u) \text{ and } \pi(v^{**}) \leq \pi(v^*)\right]}{\Pr_{\pi\in\Pi_P}\left[\pi(v^{**}) \leq \pi(v^*)\right]} \\ \leq\ &\frac{\Pr_{\pi\in\Pi_P}\left[\forall u \in P, \pi(v^*) \leq \pi(u) \text{ and } \pi(v^{**}) \leq \pi(u)\right]}{\Pr_{\pi\in\Pi_P}\left[\pi(v^{**}) \leq \pi(v^*)\right]}\end{aligned}$$

To bound the above expression, we separate our discussion into three possible cases. In the first $v^{**} \neq v^*$ and $v^{**} \in P$. The value of the expression is clearly 0 in this case. In the second we have $v^* = v^{**}$ and according to Equation (2) we have that the quantity is bounded by $1/|P|$. The last case is the one where $v^{**} \notin P$. Here, because $v^* \in P$ and $v^{**} \notin P$ we have that the events of $\pi(v^*)$ being the minimal in $\{\pi(u)\}_{u \in P}$ and $\pi(v^{**})$ being the smaller than each of the elements of $\{\pi(u)\}_{u \in P}$ are independent for uniform $\pi \in \Pi_P$. This is due to the first event being dependent of the inner order inside P and the second being independent of the same inside order. Hence,

$$\begin{aligned}
& \frac{\Pr_{\pi \in \Pi_P}\left[\forall u \in P, \pi(v^*) \leq \pi(u) \text{ and } \pi(v^{**}) \leq \pi(u)\right]}{\Pr_{\pi \in \Pi_P}\left[\pi(v^{**}) \leq \pi(v^*)\right]} \\
= \; & \frac{\Pr_{\pi \in \Pi_P}\left[\forall u \in P, \pi(v^{**}) \leq \pi(u)\right]}{\Pr_{\pi \in \Pi_P}\left[\pi(v^{**}) \leq \pi(v^*)\right]} \cdot \Pr_{\pi \in \Pi_P}\left[\forall u \in P, \pi(v^*) \leq \pi(u)\right] \\
\leq \; & \Pr_{\pi \in \Pi_P}\left[\forall u \in P, \pi(v^*) \leq \pi(u)\right] \leq 1/|P|
\end{aligned}$$

□

Lemma 2 and Lemma 3 immediately lead to Theorem 1. Also, as an immediate corollary of Theorem 1 we get

Corollary 6. *A direct distributed implementation of Algorithm 1 has, in expectation, both a single adjustment and round, in both the synchronous and asynchronous models.*

4 A Constant Broadcast Implementation

Theorem 1 promises that the expected number of nodes that need to change their output according to our template algorithm is 1. However, a direct implementation of the template in Algorithm 1 in a dynamic distributed setting may require a much larger broadcast complexity because it may be the case that a node needs to change its state several times until the MIS invariant holds at all nodes. This is because a node can be in more than a single set S_i, as discussed in the previous section. In such a case, despite the fact that the expected number of nodes in S is a constant, it may be that the expected number of state changes is much larger. Specifically, in a naive implementation, the number of broadcasts may be as large as $|S|^2$. Hence, although $\mathbb{E}[|S|] = 1$, the expected number of broadcasts may be as large as n.

We thus take a different approach for implementing the template in Algorithm 1, in the *synchronous* setting, where each node waits until it knows the maximal i for which it belongs to S_i, and changes it state only once. This allows to obtain, for almost all of the possible topology changes a constant broadcast complexity at the cost of a constant, rather than single, round complexity.

In order to implement the random permutation π we assume each node $v \in V$ has a uniformly random and independent ID $\ell_v \in [0, 1]$. We will maintain the property that each node has knowledge of its ℓ value and those of its neighbors. We describe our algorithm in Algorithm 2. This directly applies to the following topology changes: edge-insertion, graceful-edge-deletion, abrupt-edge-deletion, graceful-node-deletion and node-unmuting. An extension of the analysis is provided in Subsection 4.2 for the case of an abrupt node deletion, and a slight modification is provided in Subsection 4.1 for the case of node-insertion. The following summarizes the guarantees of our implementation, and is proven in Lemmas 9, 10, and 13. .

Theorem 7. *There is a complete fully dynamic distributed MIS algorithm which requires in expectation a single adjustment and $O(1)$ rounds for all topology changes. For edge insertions and deletions, graceful node deletion, and node unmuting, the algorithm requires $O(1)$ broadcasts, for an abrupt deletion of a node v^* it requires $O\left(\min\{\log(n), d(v^*)\}\right)$ broadcasts, and for an insertion of a node v^* it requires $O(d(v^*))$ broadcasts, in expectation.*

In the algorithm a node may be in one of four states: M for an MIS node, $\bar{M}$ for a non-MIS node, C for a node that may need to change from M to $\bar{M}$ or vice-versa, and R for a node that is ready to change. We will sometimes abuse notation and consider a state as the set of nodes which are in that state. Our goal is to maintain the MIS invariant.

Algorithm 2 MIS Algorithm for node v

```
1: $v \in M$: If some $u \in I_\pi(v)$ changes to state $C$, change state to $C$.
2: $v \in \bar{M}$: If some $u \in I_\pi(v)$ changes to state $C$ and all other $w \in I_\pi(v)$ are not in $M$, change state to $C$.
3: $v \in C$: If (1) all neighbors $u$ with $\pi(v) < \pi(u)$ are not in state $C$ and (2) $v$ changed to state $C$ at least
   2 rounds ago, change state to $R$.
4: $v \in R$: If all $u \in I_\pi(v)$ are in states $\bar{M}$ or $M$, change state to $M$ if all $u \in I_\pi(v)$ are in $\bar{M}$, and change
   state to $\bar{M}$ otherwise.
```

Any change of state of a node is followed by a broadcast of the new state to all of its neighbors. We now define our implementation as a sequence of state changes. When a topology change occurs at node v^*, if the MIS invariant still holds then v^* does not change its state and algorithm consists of doing nothing. Otherwise, v^* changes its state to C.

From states M or $\bar{M}$, a node changes to state C when it discovers it is in the set S of influenced nodes, as defined in Equation (1). From state C, a node v changes to state R when (1) none of its neighbors u for which $\pi(v) < \pi(u)$ are in state C and (2) v changed its state to C at least two rounds ago. Finally, from state R a node v returns to states M or $\bar{M}$ when all of its neighbors u for which $\pi(u) < \pi(v)$ are in states M or $\bar{M}$. In order to bound the complexity of the algorithm we first show that every node can change from state R to either M or $\bar{M}$ at most once.

Lemma 8. *In Algorithm 2, a node u changes its state from R to another state at most once.*

Proof. First, note that every $u \notin S$ never changes its state. Consider a node u changing its state from R to either M or $\bar{M}$. Since u changes from state R, if $u \neq v^*$ then it must have a neighbor $w \in I_\pi(u)$ that was in state C, changed to state R and then changed to M or $\bar{M}$. It follows that v^* must be the first node to change its state from R to M or $\bar{M}$. This event occurs only when all neighbors u of v^* are not in C, which in turn can happen only when all neighbors of each such u with higher π value have changed from C to R at least once. But, since no node could have changed its state from R to another state before v^* has done so, we have that when v^* changes its state from R to another, all $u \in S$ are in state R.

In particular, we have that at the round of the first change of a node from R to another state, there are no nodes in state C. Since a node can only change to state C due to a neighbor at state C we have that any node changing its state from R to M or $\bar{M}$ will not change its state again, thus proving our claim. ☐

Lemma 9. *For edge-insertion, graceful-edge-deletion, abrupt-edge-deletion, graceful-node-deletion and node-unmuting, Algorithm 2 requires in expectation a single adjustment, $O(1)$ rounds, and $O(1)$ broadcasts.*

Proof. Since only nodes in S can change their outputs, the number of adjustments is bounded by $|S|$, and hence is 1 in expectation, by Theorem 1. According to Lemma 8, if a node changes its state then it does so exactly three times. First it changes from either M or $\bar{M}$ to C, then it changes to R, and finally it changes to either M or $\bar{M}$ again. Since only nodes in S change their states and since the round and broadcast complexities are clearly bounded by the number of state changes plus 1 (due to the forced waiting round before changing from C to R), the claim follows. ☐

4.1 Node and Edge Insertion

When v^* is inserted, or an edge (v^*, v^{**}) is inserted, we assume that all new pairs of neighbors are notified that they are now connected, along with each other's ID. In a setting where this is not the case, we make the following adjustment, before applying Algorithm 2.

When v^* is inserted, in the first round, v^* broadcasts its random ℓ value and a temporary state $\bar{M}$ to its neighbors. In the second round, the neighbors of v^* broadcast their states and their ℓ values. Now, it may be the case that the MIS invariant does not hold at v^*, but it still holds for any other node in the graph. We now execute Algorithm 2. The expected number of adjustments hence remains 1, the number of rounds increases by two and is therefore still $O(1)$, and the number of broadcasts is now bounded by the degree $d(v^*)$.

When an edge (v^*, v^{**}) is inserted, in the first round, v^* and v^{**} broadcast their random ℓ value and state. Now it may be the case that the MIS invariant no longer holds at v^*, but it still holds for any other node in the graph. We then execute Algorithm 2. The expected number of adjustments hence remains 1, the number of rounds increases by one and is therefore still $O(1)$, and the number of broadcasts is still bounded by $O(1)$.

We therefore get the following:

Lemma 10. *For a node-insertion of a node v^*, our algorithm requires in expectation a single adjustment, $O(1)$ rounds, and $O(d(v^*))$ broadcasts. For an edge-insertion of an edge (v^*, v^{**}), our algorithm requires in expectation a single adjustment, $O(1)$ rounds, and $O(1)$ broadcasts.*

4.2 Abrupt Node Deletion

Consider a node v^* that is abruptly deleted. We denote by $v_1^*, v_2^*, \ldots, v_x^*$ the set $S_1 = S(G_{\text{old}}, G_{\text{new}}, \pi, v^*)_1$. We execute Algorithm 2, where in the first round, every v_i^*, $1 \leq i \leq x$ changes its state to C (instead of having v^* broadcast its change state to C). It is straightforward to verify that despite the above modification, only nodes in $S = S(G_{\text{old}}, G_{\text{new}}, \pi, v^*)$ can change to state C throughout the execution. With this modification, a node may change to state C more than once. However, we show the amount of times this can happen is bounded by both the degree of v^* and by $\log(n)$.

Lemma 11. *The algorithm completes after at most $3|S| + 2$ rounds.*

Proof. Consider a node v that changes to state C in round t. If $t > 1$, then there is a node u, for which $\pi(u) < \pi(v)$, that changes to C in round $t-1$. By induction, it follows that there exists a path $(v_1, v_2, \ldots, v_t)$, where $v_1 = v_i^*$ for some $1 \leq i \leq x$ and $v_t = v$, and in addition $\pi(v_i) < \pi(v_{j+1})$ for all $j < t$. Since the latter implies that the nodes are distinct, we have that $t \leq |S|$, meaning that after round $|S|$ no node changes to C. This gives that after an additional round, nodes begin to change to R. The same argument now gives that no node changes to R after round $2|S| + 1$. After an additional round nodes start changing from R, and therefore, using the same argument again, we have that no node changed its state from R after round $3|S| + 2$. To complete the proof, we argue that indeed this procedure progresses, hence eventually all nodes are in either M or $\bar{M}$, with the MIS invariant holding. This holds since two consecutive rounds with no state changes implies that the algorithm terminated: After a round with no state change, if there are nodes in state C then the node with the maximal π order among them changes to R. Otherwise, if there are nodes in state R then the node with the minimal π order among them changes to either M or $\bar{M}$. □

Lemma 12. *If v changes from either M or $\bar{M}$ to C at rounds t, it does not change to neither M or $\bar{M}$ again before round $3t + 1$. Further, each change of v to C can be associated with a different node v_i^*, for some $1 \leq i \leq x$.*

Proof. We prove the lemma by induction on t, where the base case for $t = 1$ trivially holds, as a node needs to make 3 changes. For $t > 1$, as in the previous lemma, there exists a path $(v_1, v_2, \ldots, v_t)$, where $v_1 = v_i^*$ for some $1 \leq i \leq x$ and $v_t = v$, and in addition $\pi(v_j) < \pi(v_{j+1})$ for all $j < t$.

Let $t_R \geq t + 2$ be the first round in which $v = v_t$ changes to R. Notice that until that round, v_{t-1} does not change to R, and the same holds inductively for v_j for all $1 \leq j \leq t - 1$. Let t'_R be the first round in which v_1 changes to R, and let t_S be the first round in which v_1 changes to either M or $\bar{M}$. It follows that $t'_R \geq t_R + t - 1 \geq 2t + 1$, and hence $t_S \geq 2t + 2$.

Now, let t'_S be the first round in which $v = v_t$ changes to either M or $\bar{M}$. In round $t'_S - 1$ the node v_{t-1} must be in either M or $\bar{M}$, and inductively we have that at round $t'_S - t + 1$, the node v_1 is in either M or $\bar{M}$. This implies that $t'_S - t + 1 \geq t_S \geq 2t + 2$, giving $t'_S \geq 3t + 1$ as required.

Further, when v changes its state from R to M or $\bar{M}$, the state of v_i^* is already M or $\bar{M}$. This implies that if v change its state to C again in time $t' > t$, then it is due to a change in $v_{i'}^*$, for some $i' \neq i$, $1 \leq i' \leq x$. (Although it is possible that the path from $v_{i'}^*$ to v goes through v_i^* again.) This means that each change of v to C can be associated with a different node v_i, for some $1 \leq i \leq x$. □

Lemma 13. *For an abrupt deletion of a node v^*, our algorithm requires in expectation a single adjustment, $O(1)$ rounds, and $O(\min\{\log(n), d(v^*)\})$ broadcasts.*

Proof. Now, the adjustment complexity is bounded by $|S|$ and thus by Theorem 1 it is 1 in expectation. By Lemma 11 the round complexity is $O(|S|)$, and thus by Theorem 1 it is $O(1)$ in expectation. Finally, as a corollary from Lemma 12 we get that the sequence of rounds in which a node v has changed to C is $t_1, t_2, \ldots, t_r$ with $t_1 \geq 1$, $t_{i+1} \geq 3t_i + 1$ and by Lemma 11 we have also $t_r < 3|S| + 1$. It follows that $r \leq \log_3(O(|S|))$. Moreover, since by Lemma 12 every change to C by a node v can be accounted to a different node v_i^*, we have that $r \leq x \leq d(v^*)$. This gives that the total number of broadcasts is at most $O(|S| \min\{\log(|S|), d(v^*)\})$. The claim immediately follows as since $\mathbb{E}[|S|] \leq 1$ (Theorem 1) and $|S| \leq n$. □

5 History Independence

In this section, we define and discuss the *history independence* property.

Motivation: In many well known problems, in addition to computing a feasible solution, it is sometimes required to compute an optimal solution with respect to a given objective function. For example, one may wish to find an MIS that has a maximal cardinality. Usually, obtaining optimal solutions, or even good approximate solutions, is NP-hard and therefore, we cannot expect to obtain such solutions in the distributed model. Furthermore, typically such optimization algorithms are tailored to a specific objective function, and hence it is required to handle each objective separately. Moreover, in a dynamic setting, it is more cumbersome to analyze the guarantees for the value of the objective function.

Therefore, although we do not wish to consider any specific objective function in the problem description, we do wish that the adversary will lack the ability to choose a feasible solution as she pleases (in this case we may assume she chooses the worst solution). In fact, we require that the adversary will even not be able to *bias* the output of the algorithm towards any specific solution. Formally, we define our requirement as follows.

Definition 14. *Let A be an algorithm for maintaining a combinatorial graph structure P in a dynamic distributed setting. We say that A is* history independent *if given a graph G, the output P of A is a random variable whose distribution depends only on G, and does not depend on the history of topology changes that constructed G.*

Composability: Another advantage of history independent algorithms is that they compose nicely. For example, given a history independent algorithm A for MIS, we can simulate A on the line graph $L(G)$ for obtaining a history independent algorithm B for maximal matching. Alternatively, we can simulate A on a graph $G' = f(G)$ in which every node v in G corresponds to a clique of size $\Delta + 1$ in G', and every edge in G corresponds to a matching between the appropriate cliques. This standard reduction by [43] is known to give an algorithm C for $(\Delta + 1)$-coloring in G, and since A is history independent then so is C. We note that performing the above simulations is non-trivial, as the simple topological changes in G translate into more complex changes in $L(G)$ or G', yet, the challenges are only technical and require no additional insights, and hence we omit the details.

Examples: It is straightforward to see that Algorithm 2 is a history independent MIS algorithm, because its output for any graph G is identical to the output of the greedy sequential algorithm on G, regardless of the topology changes that resulted in G. To further exemplify what this important property gives, consider the following examples, in which we compare executions of history independent algorithms to worst case solutions. Although history dependent algorithms do not necessarily yield the worst solution, the *natural* history dependent algorithm indeed yields the worst solution in these examples. Here we think of the natural algorithm as the greedy algorithm that gives every new node or edge the best value that is possible without making any global changes. For this natural algorithm, one can easily verify that for any feasible output there is a pattern of topology changes that can force the algorithm to produce it.

Example: MIS in a Star. Assume that an adversary controls the topology changes in the graph and chooses them so that a graph G_{star} is created, where G_{star} is a star on n nodes. Since our MIS algorithm simulates random greedy, there is a probability of $1/n$ that the center of the star has the lowest order among all nodes, in which case it is the only node in the MIS. With the remaining probability of $1-1/n$, a different node has the lowest order, which results in the MIS being all nodes except the center, which is the largest MIS possible in G_{star}. The expected size of the resulting MIS is therefore linear in n, implying that it is within a constant factor of the size of the *maximum* independent set (maximal cardinality independent set). For comparison, recall that the worst-case MIS in a star is the center alone, and its size is 1.

Example 2: Maximal matching of many 3-paths. Assume an adversary constructs a graph G_{3paths}, which contains $n/4$ disjoint paths of length 3 edges. Since our maximal matching algorithm simulates a random greedy MIS algorithm on the line graph $L(G_{3paths})$, we have that for every 3-path independently, with probability $2/3$ its matching is of size 2 and with probability $1/3$ its matching is of size 1. Therefore, the expected size of the matching we obtain is $5n/12$. For comparison, notice that the worst-case maximal matching in G_{3paths} has size $n/4$.

Example 3: Coloring. Regarding coloring algorithms, there is much more room for improvement upon using the standard reduction with our MIS algorithm. As in our MIS algorithm, we would have liked to have an algorithm that simulates the sequential random greedy algorithm for coloring as well, since, for example, it would imply the following.

Assume that an adversary controls the topology changes in the graph and chooses them so that a graph $G = (V, E)$ is created, where G is a bipartite graph on the set of nodes $V = L \cup R$, for $L = \{u_1, \dots, u_{n/2}\}$ and $R = \{v_1, \dots, v_{n/2}\}$. In E we have an edge between the nodes u_i and v_j for every $i \neq j$. Thus, G is a complete bipartite graph minus a perfect matching. If we run a random greedy coloring algorithm, then the first node, say u_i gets the color 1 when being inserted into the graph. If the next inserted node after u_i is v_j, for any $j \neq i$, then it gets the color 2, and afterwards every node in L gets color 1 and every node in R gets color 2. If the next inserted node after u_i is u_j, for any $j \neq i$, then it gets the color 1, and afterwards every node in L gets color 1 and every node in R gets color 2. That is, with probability $1 - 1/n$, we get an optimal 2-coloring. With the remaining $1/n$ probability we might get a coloring as bad as linear in Δ. This gives that in expectation, we get a coloring whose palette size is a constant factor away from optimal.

We can, of course, simulate the random greedy sequential coloring, but the problem is that we pay a cost of 2^{Δ} adjustments. The way we can do this is to maintain that every color i is an MIS in the graph $G - \{C_j\}_{1 \leq j < i}$, where C_j is the set of nodes of color j. However, a change to a node may cause each node in S to induce 2 sources of sets for the next color, which would result in a total set of 2^{Δ} adjustments. Notice that this is much worse compared to what a naive dynamic distributed coloring algorithm would give. It is a curious question whether we can indeed enjoy both worlds here, or whether any lower bound can be proved.

The above examples illustrate how a property of the output of a history independent algorithm on a graph G can be analyzed as a simple combinatorial problem. This can lead to better guarantees compared to only being able to assume the worst case.

6 Discussion

This paper studies computing an MIS in a distributed dynamic setting. The strength of our analysis lies in obtaining that for an algorithm that simulates the sequential random greedy algorithm, the size of the set of nodes that need to change their output is in expectation 1. This brings the locality of the fundamental MIS problem to its most powerful setting.

We believe our work sets the ground for much more research in this crucial setting. Below we discuss some open problems that arise from our work.

An immediate open question is whether our analysis can be extended to cope with more than a single failure at a time. Second, there are many additional problems that can be addressed in the dynamic distributed setting, especially in the synchronous case. We believe that our contribution can find applications in solving many additional dynamic distributed tasks.

A major open question is whether our techniques can be adapted to sequential dynamic graph algorithms, which constitutes a major area of research in the sequential setting [12, 13, 16, 22–24, 27–30, 33, 48, 49, 51, 52]. A formal definition and description of typical problems can be found in, e.g., [15]. Notice that our algorithms are *fully dynamic*, which means that they handle both insertions and deletions (of edges and nodes). Although our template for finding an MIS can be easily implemented in a sequential dynamic setting, it would come with a cost of at least $O(\Delta)$ for the update complexity in a direct implementation. This is because we would have to access neighbors of the set of nodes analyzed in Theorem 1. Our distributed implementation avoids this by having them simply not respond since they do not need to change their output, and hence they do not contribute to the communication. Nevertheless, we believe that our approach may be useful for designing an MIS algorithm for the dynamic sequential setting, and leave this for future research.

References

[1] N. Ailon, M. Charikar, and A. Newman. Aggregating inconsistent information: Ranking and clustering. *J. ACM*, 55(5), 2008.

[2] N. Alon, L. Babai, and A. Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *J. Algorithms*, 7(4):567–583, 1986.

[3] C. Avin, M. Koucký, and Z. Lotker. How to explore a fast-changing world (cover time of a simple random walk on evolving graphs). In *Automata, Languages and Programming, 35th International Col-*

loquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part I: Tack A: Algorithms, Automata, Complexity, and Games, pages 121–132, 2008.

[4] B. Awerbuch, I. Cidon, and S. Kutten. Communication-optimal maintenance of replicated information. In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume II*, pages 492–502, 1990.

[5] B. Awerbuch and M. Sipser. Dynamic networks are as fast as static networks (preliminary version). In *29th Annual Symposium on Foundations of Computer Science, White Plains, New York, USA, 24-26 October 1988*, pages 206–220, 1988.

[6] B. Awerbuch and G. Varghese. Distributed program checking: a paradigm for building self-stabilizing distributed protocols (extended abstract). In *32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1-4 October 1991*, pages 258–267, 1991.

[7] Y. Azar, S. Kutten, and B. Patt-Shamir. Distributed error confinement. *ACM Trans. Algorithms*, 6(3):48:1–48:23, July 2010.

[8] L. Barenboim and M. Elkin. *Distributed Graph Coloring: Fundamentals and Recent Developments*. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, 2013.

[9] L. Barenboim, M. Elkin, and F. Kuhn. Distributed (delta+1)-coloring in linear (in delta) time. *SIAM J. Comput.*, 43(1):72–95, 2014.

[10] L. Barenboim, M. Elkin, S. Pettie, and J. Schneider. The locality of distributed symmetry breaking. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 321–330, 2012.

[11] S. Baswana, S. Khurana, and S. Sarkar. Fully dynamic randomized algorithms for graph spanners. *ACM Transactions on Algorithms*, 8(4):35, 2012.

[12] A. Bernstein and C. Stein. Fully dynamic matching in bipartite graphs. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, pages 167–179, 2015.

[13] S. Bhattacharya, M. Henzinger, and G. F. Italiano. Deterministic fully dynamic data structures for vertex cover and matching. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 785–804, 2015.

[14] S. Cicerone, G. D. Stefano, D. Frigioni, and U. Nanni. A fully dynamic algorithm for distributed shortest paths. *Theor. Comput. Sci.*, 297(1-3):83–102, 2003.

[15] C. Demetrescu, D. Eppstein, Z. Galil, and G. F. Italiano. Dynamic graph algorithms. In M. J. Atallah and M. Blanton, editors, *Algorithms and Theory of Computation Handbook*, chapter 9. Chapman & Hall/CRC, 2010.

[16] C. Demetrescu and G. F. Italiano. Fully dynamic transitive closure: Breaking through the o(n^2) barrier. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pages 381–389, 2000.

[17] E. W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Commun. ACM*, 17(11):643–644, 1974.

[18] S. Dolev. *Self-Stabilization*. MIT Press, 2000.

[19] S. Dolev and T. Herman. Superstabilizing protocols for dynamic distributed systems. *Chicago J. Theor. Comput. Sci.*, 1997, 1997.

[20] S. Dolev and N. Tzachar. Empire of colonies: Self-stabilizing and self-organizing distributed algorithm. *Theor. Comput. Sci.*, 410(6-7):514–532, 2009.

[21] M. Elkin. A near-optimal distributed fully dynamic algorithm for maintaining sparse spanners. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Principles of Distributed Computing, PODC 2007, Portland, Oregon, USA, August 12-15, 2007*, pages 185–194, 2007.

[22] D. Eppstein, Z. Galil, G. F. Italiano, and A. Nissenzweig. Sparsification - a technique for speeding up dynamic graph algorithms. *J. ACM*, 44(5):669–696, 1997.

[23] S. Even and Y. Shiloach. An on-line edge-deletion problem. *J. ACM*, 28(1):1–4, 1981.

[24] G. N. Frederickson. Data structures for on-line updating of minimum spanning trees, with applications. *SIAM J. Comput.*, 14(4):781–798, 1985.

[25] M. Ghaffari. Towards an optimal distributed algorithm for maximal independent set. *CoRR*, abs/1506.05093, 2015.

[26] N. Guellati and H. Kheddouci. A survey on self-stabilizing algorithms for independence, domination, coloring, and matching in graphs. *J. Parallel Distrib. Comput.*, 70(4):406–415, 2010.

[27] M. Henzinger, S. Krinninger, and D. Nanongkai. Dynamic approximate all-pairs shortest paths: Breaking the o(mn) barrier and derandomization. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 538–547, 2013.

[28] M. R. Henzinger and V. King. Fully dynamic biconnectivity and transitive closure. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, 23-25 October 1995*, pages 664–672, 1995.

[29] M. R. Henzinger and V. King. Randomized fully dynamic graph algorithms with polylogarithmic time per operation. *J. ACM*, 46(4):502–516, 1999.

[30] J. Holm, K. de Lichtenberg, and M. Thorup. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 79–89, 1998.

[31] A. Israeli and A. Itai. A fast and simple randomized parallel algorithm for maximal matching. *Inf. Process. Lett.*, 22(2):77–80, 1986.

[32] G. F. Italiano. Distributed algorithms for updating shortest paths (extended abstract). In *Distributed Algorithms, 5th International Workshop, WDAG '91, Delphi, Greece, October 7-9, 1991, Proceedings*, pages 200–211, 1991.

[33] B. M. Kapron, V. King, and B. Mountjoy. Dynamic graph connectivity in polylogarithmic worst case time. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 1131–1142, 2013.

[34] M. König and R. Wattenhofer. On local fixing. In *Principles of Distributed Systems - 17th International Conference, OPODIS 2013, Nice, France, December 16-18, 2013. Proceedings*, pages 191–205, 2013.

[35] A. Korman and D. Peleg. Labeling schemes for weighted dynamic trees. In *Automata, Languages and Programming, 30th International Colloquium, ICALP 2003, Eindhoven, The Netherlands, June 30 - July 4, 2003. Proceedings*, pages 369–383, 2003.

[36] F. Kuhn, N. A. Lynch, and R. Oshman. Distributed computation in dynamic networks. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 513–522, 2010.

[37] F. Kuhn, T. Moscibroda, and R. Wattenhofer. What cannot be computed locally! In *Proceedings of the Twenty-Third Annual ACM Symposium on Principles of Distributed Computing, PODC 2004, St. John's, Newfoundland, Canada, July 25-28, 2004*, pages 300–309, 2004.

[38] S. Kutten and D. Peleg. Fault-local distributed mending. *J. Algorithms*, 30(1):144–165, 1999.

[39] S. Kutten and D. Peleg. Tight fault locality. *SIAM J. Comput.*, 30(1):247–268, 2000.

[40] C. Lenzen, J. Suomela, and R. Wattenhofer. Local algorithms: Self-stabilization on speed. In *Stabilization, Safety, and Security of Distributed Systems, 11th International Symposium, SSS 2009, Lyon, France, November 3-6, 2009. Proceedings*, pages 17–34, 2009.

[41] R. Levi, R. Rubinfeld, and A. Yodpinyanee. Local computation algorithms for graphs of non-constant degrees. *CoRR*, abs/1502.04022, 2015.

[42] N. Linial. Locality in distributed graph algorithms. *SIAM J. Comput.*, 21(1):193–201, 1992.

[43] M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM J. Comput.*, 15(4):1036–1053, 1986.

[44] Y. Mansour, A. Rubinstein, S. Vardi, and N. Xie. Converting online algorithms to local computation algorithms. In *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I*, pages 653–664, 2012.

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

[46] H. N. Nguyen and K. Onak. Constant-time approximation algorithms via local improvements. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 327–336, 2008.

[47] D. Peleg. *Distributed Computing: A Locality-sensitive Approach*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[48] L. Roditty and U. Zwick. On dynamic shortest paths problems. *Algorithmica*, 61(2):389–401, 2011.

[49] L. Roditty and U. Zwick. Dynamic approximate all-pairs shortest paths in undirected graphs. *SIAM J. Comput.*, 41(3):670–683, 2012.

[50] M. Schneider. Self-stabilization. *ACM Comput. Surv.*, 25(1):45–67, 1993.

[51] R. E. Tarjan. Efficiency of a good but not linear set union algorithm. *J. ACM*, 22(2):215–225, 1975.

[52] M. Thorup. Near-optimal fully-dynamic graph connectivity. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, pages 343–350, 2000.

[53] V. Turau. Linear self-stabilizing algorithms for the independent and dominating set problems using an unfair distributed scheduler. *Inf. Process. Lett.*, 103(3):88–93, 2007.

[54] Y. Yoshida, M. Yamamoto, and H. Ito. Improved constant-time approximation algorithms for maximum matchings and other optimization problems. *SIAM J. Comput.*, 41(4):1074–1093, 2012.