Distributed MIS via All-to-All Communication

Mohsen Ghaffari
ETH Zurich, Switzerland
ghaffari@inf.ethz.ch

ABSTRACT

Computing a Maximal Independent Set (MIS) is a central problem in distributed graph algorithms. This paper presents an improved randomized distributed algorithm for computing an MIS in an all-to-all communication distributed model, known as the *congested clique* model, defined as follows: Given a graph $G = (V, E)$, initially each node knows only its neighbors. Communication happens in synchronous rounds over a complete graph, and per round each node can send $O(\log n)$ bits to each other node.

We present a randomized algorithm that computes an MIS in $\tilde{O}(\log \Delta/\sqrt{\log n} + 1) \leq \tilde{O}(\sqrt{\log \Delta})$ rounds of congested clique, with high probability. Here Δ denotes the maximum degree in the graph. This improves quadratically on the $O(\log \Delta)$ algorithm of [Ghaffari, SODA'16]. The core technical novelty in this result is a certain *local sparsification* technique for MIS, which we believe to be of independent interest.

CCS CONCEPTS

• **Theory of computation → Distributed algorithms**;

KEYWORDS

Distributed Graph Algorithms; Maximal Independent Set (MIS); Congested Clique

1 INTRODUCTION & RELATED WORK

Maximal Independent Set (MIS) is a central problem in the area of *distributed graph algorithms*, and it has been studied extensively since the 1980s. In this paper, we present an improved randomized distributed MIS algorithm for distributed settings where the communication is all-to-all. Let us first overview the models.

The Three Models of Distributed Graph Algorithms: There are three standard synchronous message passing models of distributed computation:

(1) The CONGEST model [36]: The network is abstracted as an n-node graph $G = (V, E)$. There is one processor per node of the graph, which initially knows only the edges incident on that node. Per round each processor/node can send one B-bit message to each of its neighbors, where typically one assumes $B = O(\log n)$.
(2) The LOCAL model [27, 36]: This is the same as the CONGEST model, except the relaxation that the message sizes are not bounded.
(3) The CONGESTED-CLIQUE model [29, 30]: This is the same as the CONGEST model, except the relaxation that the communication is *all-to-all*, and per round, each node can send $B = O(\log n)$ bits to each other node.

In all three of the models, the solution is output in a distributed format, which means that each node should know its own part of the output, e.g., whether it is in the computed maximal independent set or not. It should be noted that the third model is much younger, and its study started with the work of [29, 30]. However, nowadays, there are a wide range of settings where all-to-all communication is available to the distributed system, which makes the CONGESTED-CLIQUE model far more relevant. As such, this model has attracted a vast amount of attention over the past few years, see e.g. [6–9, 11–13, 15–19, 22, 25, 32, 35].

1.1 Related Work

Algorithms in LOCAL **and** CONGEST **models**: In 1986, Luby[31] and Alon, Babai, and Itai[3] presented randomized algorithms that compute an MIS in $O(\log n)$ rounds of the CONGEST model, with high probability[1]. Due to well-known reductions[28], these algorithm directly lead to $O(\log n)$ round algorithms for a few other classic problems, including maximal matching, $(\Delta + 1)$-vertex coloring, and $(2\Delta - 1)$-edge coloring, where Δ denotes the graph's maximum degree.

Recently, faster MIS algorithms were presented for graphs with small maximum degree: Barenboim et al.[5] gave an $O(\log^2 \Delta) + 2^{O(\sqrt{\log \log n})}$ round algorithm, and Ghaffari[13] improved the bound to $O(\log \Delta) + 2^{O(\sqrt{\log \log n})}$. However, it should be noted that these algorithms work in the LOCAL model and use large messages.

Lower Bounds in LOCAL **and** CONGEST **models**: The following lower bounds hold even in the stronger LOCAL model. Linial[27] showed that $\Omega(\log^* n)$ rounds are necessary for computing an MIS, even on a cycle graph. Kuhn et al. [23, 24] showed that any algorithm for MIS needs $\Omega(\min\{\frac{\log \Delta}{\log \log \Delta}, \sqrt{\frac{\log n}{\log \log n}}\})$ rounds.

Algorithms in the CONGESTED-CLIQUE **model**: Clearly, the $O(\log n)$ round MIS algorithms of [3, 31] work in the CONGESTED-CLIQUE model as well. But ideally we would like to see much faster algorithms in CONGESTED-CLIQUE.

Particularly, since there is no locality limitation here, the aforementioned lower bounds do not apply. Moreover, no other lower bound is known. Because of that, there has been a general question lurking around in the area for a while:

[1] As standard, "*with high probability*" indicates probability at least $1 - 1/n^c$, for a desirably large constant $c \geq 2$.

PODC '17, July 25-27, 2017, Washington, DC, USA

ACM ISBN 978-1-4503-4992-5/17/07…$15.00
https://doi.org/10.1145/3087801.3087830

Can we solve any of the classic local problems — MIS, maximal matching, $(\Delta+1)$-vertex-coloring, or $(2\Delta-1)$-edge-coloring — much faster in the CONGESTED-CLIQUE *model. In particular, can we go below the respective locality-based lower bounds?*

An improved $O(\log\Delta)$ round MIS algorithm in CONGESTED-CLIQUE follows from [13]. By standard reductions (with minor modifications) [28], this round complexity also extends to the other three problems. This $O(\log\Delta)$ upper bound avoids the $\Omega(\log^* n)$ lower bound, but in the worst case it is still $O(\log n)$. It also does not go below the $\Omega(\min\{\frac{\log\Delta}{\log\log\Delta}, \sqrt{\frac{\log n}{\log\log n}}\})$ bound.

Some other related work: It is worth noting that there are few work in the CONGESTED-CLIQUE model, which do not address MIS or the other three problems mentioned above, but provide solutions for some weaker forms of the problems. In particular, Berns et al.[7] presented an algorithm with expected round complexity of $O(\log\log n)$ for computing a 2-ruling set, and Hegeman et al.[18] extended this to an expected $O(\log\log\log n)$-round algorithm for 3-ruling set. A k-ruling set is an independent set S in which any node of the graph has some S-node within its distance k. Finally, Hegeman and Pemmaraju[17] presented an algorithm with expected round complexity $O(\log\log\log n)$ for $O(\Delta)$-vertex-coloring. All of these are significantly more relaxed problems than MIS, and to the best of our understanding, none of the methods seem to imply anything for MIS (in general graphs). Moreover, the fact that the provided complexity guarantees hold only in expectation makes these results even less related to our problem.

1.2 Our Result and Approach

THEOREM 1.1. *There is a randomized distributed algorithm that computes a maximal independent set in $O(\frac{\log\Delta\cdot\log\log\Delta}{\sqrt{\log n}}+\log\log\Delta)$ rounds of the congested clique model, with high probability.*

This improves over the $O(\log\Delta)$ round MIS algorithm of [13] roughly quadratically, and exponentially if $\Delta\leq 2^{\sqrt{\log n}}$. Connecting back to the question stated above, this round complexity provides the first positive answer for it, by going below the aforementioned $\Omega(\min\{\frac{\log\Delta}{\log\log\Delta}, \sqrt{\frac{\log n}{\log\log n}}\})$ lower bound of [24].

Our Approach in a Nutshell: Here, we provide a brief intuition of our approach. For simplicity, we discuss only an $\tilde{O}(\sqrt{\log n})$-round algorithm for computing an MIS in the congested clique model.

The starting point in our approach is an algorithm of [13], which computes an MIS in $O(\log n)$ rounds of the CONGEST model (with some additional nice properties). We are now working in a more powerful model, the congested clique model, which allows all-to-all communications. Thus, wishfully thinking, we may hope to simulate the algorithm much faster in this model. In particular, an ideal case would be if we can simulate each $r=O(\sqrt{\log n})$ rounds of the algorithm, much faster in the CONGESTED-CLIQUE model, say in merely $O(\log\log n)$ rounds. Then, we would get an $O(\sqrt{\log n}\cdot\log\log n)$ round MIS algorithm.

Towards that hope, let us recall a well-known *locality* property of distributed models, as observed by Linial[27, 28]: In any r-round algorithm in the CONGEST model, each node v can learn at most the information known at the beginning to the nodes within its r-hop neighborhood, that is, the topology induced by the r-hop neighborhood of v and the bits of randomness in those nodes. This simple property has been used extensively by the lower bounds, and also by some algorithmic results, for instance by Parnas and Ron [34] and the follow up work to transform distributed graph algorithms into *sublinear centralized algorithms* or *local computation algorithms* as introduced by Alon et al. [4] and Rubinfeld et al. [38].

Having this property in mind, if we could somehow make each node v learn its r-hop neighborhood fast, afterward node v would be able to simulate the r-round algorithm locally and compute its own output. However, that is not so feasible! This neighborhood can be quite large, e.g., it can include even up to $O(n^2)$ edges. Thus, learning the r-hop neighborhood is hopeless.

So what do we do? In a rough sense, we design a certain *local graph sparsification* method, coupled with a new $O(\log n)$-round MIS algorithm in the CONGEST model — which builds on the algorithm of [13] — such that the following holds: We can make each node v simulate $r=O(\sqrt{\log n})$ rounds of the latter algorithm if it knows its r-hop neighborhood in a locally sparse graph G_* generated by the *local sparsification* procedure. The sparsification is such that, this whole r-neighborhood is small has size at most n^δ for a desirably small constant $\delta>0$. In particular, the maximum degree in G_* is bounded to $2^{O(\sqrt{\log n})}$. Designing these two coupled algorithms, the sparsification and the corresponding MIS algorithm, is the core technical novelty of this paper.

We also explain a simple approach which allows each node to learn its r-hop neighborhood in a locally-sparse graph G_*, with sufficient sparsity, in only $O(\log r)$ rounds of the congested clique model. In short, the method works by making each node learn its 2-hop neighborhood in $O(1)$ rounds, and then applying this procedure recurisvely on the $G_*^{2^i}$ powers of the base graph G_*.

Putting the above pieces together, we can simulate each $r=O(\sqrt{\log n})$ rounds of the new CONGEST-model MIS algorithm in $\log r=O(\log\log n)$ rounds of CONGESTED-CLIQUE, hence leading to a $O(\sqrt{\log n}\cdot\log\log n)$ round MIS algorithm in this model.

We believe that the particular *local sparsification* methodology set forth here, where the sparsification is coupled with a corresponding algorithm for the graph problem at hand and can be used to simulate the algorithm more efficiently, can be of interest beyond this work. Intuitively, one may be able to design similar procedures for some of the other local problems. Furthermore, it is conceivable that such a *local sparsification* method may lead to more efficient *local computation algorithms* [4, 38] in high-degree graphs, a subarea which remains relatively open.

2 MIS IN $\tilde{O}(\sqrt{\text{LOG}\,\Delta})$ ROUNDS OF CONGESTED CLIQUE

In this section, we present our randomized distributed algorithm that proves Theorem 1.1. In particular, it computes an MIS in $\tilde{O}(\sqrt{\log\Delta})$ rounds of the congested clique model.

See Section 1.2 for a high-level explanation of the approach. As mentioned there, the starting point in our approach is the MIS algorithm of [13]. We will modify this algorithm, in a series of steps, such that at the end, we have a CONGEST-model algorithm coupled

with a *local sparsification* procedure, with the following crucial property: the CONGEST-model MIS algorithm can be simulated much faster in the CONGESTED-CLIQUE model by (iterations of) each node learning its neighborhood in a certain locally sparse graph generated by the sparsification.

In particular, we present an $O(\log \Delta)$ round algorithm in the CONGEST model that computes a nearly-maximal independent set S, with high probability, in the sense that, after removing the S-nodes and their neighbors from the graph, the remaining graph has no more than $O(n)$ edges. This is formalized in Lemma 2.11. We will refer to this still as an MIS algorithm because running it for $O(\log n)$ rounds would indeed generate a maximal independent set, with high probability. However, instead of doing it that way, after the first $O(\log \Delta)$ rounds, we will use standard techniques of the congested clique to solve the remaining graph — i.e., decide about nodes that remain undecided — in $O(1)$ rounds. This will add some nodes to S to ensure that it is a maximal independent set. In the following, the core of our discussion is the $O(\log \Delta)$ round algorithm in the CONGEST model and its simulation in $\tilde{O}(\sqrt{\log \Delta})$ rounds of congested clique.

Since the final CONGEST-model MIS algorithm and its sparsification may seem somewhat unintuitive at a first glance, we present the algorithm in a number of gradual steps. Interestingly, as we will point out, each of the intermediate algorithms has some nice properties, making it interesting on its own. Indeed, the intermediate MIS algorithm and analysis that we present in Section 2.2 can be viewed as improving on [1, 39], as we shall clarify.

2.1 A Brief Recap on the MIS Algorithm of [13]

We start with a brief review of the MIS algorithm of [13]:

The MIS Algorithm of [13]: In each round, each node v gets *marked* with probability $p_t(v)$. If v is marked and none of its neighbors is marked, then v joins the MIS and gets removed along with its neighbors. The marking probabilities are set as follows: Initially $p_0(v) = 1/2$, and

$$p_{t+1}(v) = \begin{cases} p_t(v)/2, & \text{if } d_t(v) = \sum_{u \in N(v)} p_t(u) \geq 2 \\ \min\{2p_t(v), 1/2\}, & \text{if } d_t(v) = \sum_{u \in N(v)} p_t(u) < 2. \end{cases}$$

Despite its extreme simplicity, this algorithm is in a sense *too dynamic/active* for our purposes of fast simulation in the congested clique. To predict whether $p_t(v)$ should increase or decrease in one round, we need to compute $d_t(v) = \sum_{u \in N(v)} p_t(u)$, which itself means we should know the state (e.g. marking probability) of all the neighbors in that round. That requires receiving the values from all the neighbors. Moreover, it in turn essentially requires knowing the state of all 2-hop neighborhood nodes in round $t-1$, and so on. Keeping track of this rapidly growing topology would require a message size larger than what the congested clique model admits.

2.2 Intermediate Algorithm (1): The Beeping MIS Algorithm

Intuition

One observation is that, per round of the above algorithm, each node u needs to send its $p_t(u)$ value to each of its neighbors. This seems inefficient, given that the objective is to just distinguish whether $d_t(v) = \sum_{u \in N(v)} p_t(u)$ is large or small. We can even allow some slack in distinguishing these two situations.

We would like to have a much simpler way to distinguish these two possibilities. Particularly, we wish to have much less communication. We use a simple observation: A good indicator of whether $d_t(v)$ is large or not is whether there is a marked node in $N_t(v)$ or not. If $d_t(v)$ is larger than some constant, there is a constant probability that there is at least one marked neighbor, and if $d_t(v)$ is smaller than some other constant, the chances of having a marked neighbor are less than a constant. Hence, we can intuitively take the existence of a marked neighbor as an indicator signal for whether $d_t(v)$ is large or not, and use it to change $p_t(v)$ accordingly. An important property of this method, which we will formalize and leverage later on, is that when $d_t(v)$ is small, the set of "important" neighbors is locally spare. We will see later that this small $d_t(v)$ is the key regime of interest during the execution. To put the property in different words, we can say that per round only a small number of neighbors have a chance to mark themselves, and knowing these potential marks suffices for simulating the algorithm.

The Algorithm

We present the algorithm in the terminology of the (full-duplex) *beeping* model, which is a very basic distributed model. See [10] for one of the early works introducing (the half-duplex) variant of this model. In the full-duplex beeping model, per round each node either *beeps* or listens, and each node can distinguish only whether zero or at least one of its neighbors is beeping. A node cannot distinguish the number of beeping neighbors, if there is at least one. This model and its close variants are known to capture a wide range of wireless networks [1, 10, 14] and biological networks — see e.g., [2] and the follow up theoretical work[1, 33, 39].

The algorithm is a simple and natural dynamic. It is essentially the same algorithm as [39], and also somewhat close to the algorithms presented in [13]. Our novelty here is in the analysis.

The Beeping MIS Algorithm: The algorithm works in iterations, each made of two rounds:

(R1) In the first round of iteration t, each node v *beeps* with probability $p_t(v)$ and remains silent otherwise. Initially $p_1(v) = 1/2$. If v beeps and all its neighbors are silent, then v joins the MIS. Node v updates its beeping probability for the next iteration as follows:

$$p_{t+1}(v) = \begin{cases} p_t(v)/2, & \text{if some neighbor of } v \text{ beeps,} \\ \min\{2p_t(v), \frac{1}{2}\}, & \text{otherwise.} \end{cases}$$

(R2) In the second round of iteration t, all nodes in the MIS beep. If a non-MIS node hears a beep, it learns that it has an MIS neighbor. MIS nodes and their neighbors get removed from the problem, for the next iterations.

The Analysis

In Theorem 2.1 we show that after $O(\log \Delta + \log 1/\varepsilon)$ rounds of this algorithm, each node is removed with probability at least $1-\varepsilon$. This can be viewed as an improvement/refinement of the analysis of

Scott, Jeavons, and Xu[39], which proved an $O(\log n)$ global round complexity, with high probability[2].

To analyze the algorithm, we focus on an arbitrary node v. Suppose that $\deg(v)$ denotes the degree of node v at the beginning of the algorithm. We prove that:

THEOREM 2.1. *For each node v, during $T = C(\log \deg(v) + \log 1/\varepsilon)$ rounds, where C is a large enough constant, with probability at least $1 - \varepsilon$, either node v or a neighbor of v is added to MIS. Furthermore, this guarantee holds independent of the randomness outside 2-hop neighborhood of v.*

This theorem implies that the expected time for removal of each node v is at most $O(\log \deg(v))$, and perhaps more importantly, we have an *exponential* decay of the probability of remaining after that point. Thus, after $O(\log \Delta)$ rounds, the probability of node v remaining is at most $\varepsilon = 1/\operatorname{poly}(\Delta)$. This, in addition to the stated independence property, allows us to prove that after $O(\log \Delta)$ rounds, the graph is shattered, the remaining components are "small", and at most $O(n)$ edges remain. See Lemma 2.11 for the formal statement.

For the analysis, we focus on the first rounds of the iterations; the second rounds are the basic clean up steps for removing MIS nodes and their neighbors. In fact, ignoring those second rounds per iteration, and with a slight abuse of terminology, in the sequel, we use *rounds* to refer to the first rounds of the corresponding iterations.

We start with some notations. Define $d_t(v) = \sum_{u \in N(v)} p_t(v)$. We call a node v in iteration t *heavy*, *moderate*, and *light* if we have $d_t(v) > 10$, $0.01 \leq d_t(v) \leq 10$, and $d_t(v) < 0.01$, respectively. Also we define $d'_t(v)$ to be the summation of the beeping probabilities over non-heavy neighbors of v, that is, $d'_t(v) = \sum_{u \in N(v),\, d_t(u) \leq 10} p_t(u)$. We also define two types of golden rounds:

Golden Type-1 Round: $p_t(v) = 1/2$ and $d_t(v) \leq 0.02$, or

Golden Type-2 Round: $d_t(v) > 0.01$ and $d'_t(v) \geq 0.01 d_t(v)$.

One can see with simple calculations that in each golden round, node v gets removed with at least a positive constant probability. This is because of node v joining MIS, or a neighbor joining MIS, in type-1 and type-2 golden rounds, respectively. This statement is captured by the following lemma, the proof of which is left as a simple exercise.

LEMMA 2.2. *In each golden round, node v gets removed with at least a constant probability $\gamma > 0$.*

As the core of the analysis, we prove Lemma 2.3, stated below. This lemma shows that during the first T rounds, with probability at least $1 - \varepsilon/2$, there are at least $0.05T$ golden rounds for v. Having this, we can conclude that the probability of v remaining after all these golden rounds is at most $\varepsilon/2 + (1-\gamma)^{0.05T} < \varepsilon/2 + \varepsilon/2 = \varepsilon$, where the inequality holds by choosing a large enough constant C in the definition of T. Thus, to prove Theorem 2.1, it suffices to prove the following key lemma.

LEMMA 2.3. *For each node v, during $T = C(\log \deg(v) + \log 1/\varepsilon)$ rounds, where C is a large enough constant, with probability at least $1 - \varepsilon/2$, there are at least $0.05T$ golden rounds.*

To prove Lemma 2.3, we first present two simple helper lemmas, Lemma 2.4 and Lemma 2.5, which intuitively state that the system is far more likely to move in the desirable direction. Let us say we have a *wrong move* if one of the following two holds: (1) $d_t(v) \leq 0.02$ and $p_{t+1}(v) = p_t(v)/2$, or (2) $d_t(v) > 0.01$ and $d'_t(v) < 0.01 d_t(v)$ but $d_{t+1}(v) > 0.6 d_t(v)$. In the following two lemmas, we show that the probability of a wrong move is at most 0.02. Then, we use these two lemmas to prove Lemma 2.3.

LEMMA 2.4. *If $d_t(v) \leq 0.02$, then with probability at least 0.98, we have $p_{t+1}(v) = \min\{2p_t(v), \frac{1}{2}\}$.*

PROOF OF LEMMA 2.4. Suppose that $d_t(v) \leq 0.02$. If no neighbor of v beeps, then $p_{t+1}(v) = \min\{2p_t(v), \frac{1}{2}\}$. The probability that no neighbor of v beeps is

$$\prod_{u \in N(v)} (1 - p_t(u)) \geq 4^{-\sum_{u \in N(v)} p_t(u)} = 4^{-d_t(v)} \geq 0.98.$$

□

LEMMA 2.5. *If $d_t(v) > 0.01$ and $d'_t(v) < 0.01 d_t(v)$, with probability 0.98, we have $d_{t+1}(v) \leq 0.6 d_t(v)$.*

PROOF OF LEMMA 2.5. Suppose that $d_t(v) > 0.01$ and $d'_t(v) < 0.01 d_t(v)$. Hence, a 0.99 fraction of $d_t(v)$ is contributed by heavy neighbors of v. Let us call a heavy neighbor u bad if it has no beeping neighbor and thus sets $p_{t+1}(u) = \min\{2p_t(u), \frac{1}{2}\}$. We would like to upper bound $d_{t+1}(v)$. The contributions to $d_{t+1}(v)$ comes from three sources: (1) bad heavy neighbors, which will at most double their probability (2) good heavy neighbors, which will halve their probability, (3) non-heavy neighbors, which will at most double their probability. The contribution of the last two categories to $d_{t+1}(v)$ is at most $0.5 d_t(v) + 0.02 d_t(v) = 0.52 d_t(v)$, and this holds deterministically. What is crucial to analyze is the contribution of the first group, that is, bad heavy neighbors. For each heavy neighbor u of v, the probability that u is bad—that is, the probability that no neighbor of u beeps—is

$$\prod_{w \in N(v)} (1 - p_t(w)) < e^{-\sum_{w \in N(u)} p_t(u)} = e^{-d_t(v)} < e^{-10} < 0.0001.$$

Thus, by linearity of the expectation, the expected contribution of bad heavy neighbors to $d_{t+1}(v)$ is at most $0.0002 d_t(v)$, as each at most doubles its probability. By Markov's inequality, the probability that this contribution is more than $0.01 d_t(v)$ is at most 0.02. In other words, with probability at least 0.98, the contribution is no more than $0.01 d_t(v)$. Therefore, considering all the three categories, with probability at least 0.98, we have $d_{t+1}(v) \leq (0.52 + 0.01) \cdot d_t(v) < 0.6 d_t(v)$. □

Now that we are equipped with these two helper lemmas, we go back to proving Lemma 2.3.

[2] We emphasize that the algorithm that we discuss works in the full-duplex variant of the beeping model, where each node v can detect whether any of its neighbors are beeping or not, even if v is beeping at the same time. We refer the interested reader to a recent work of Holzer and Lynch [20, 21], which discusses a beeping MIS algorithm in the half-duplex model—where only a listening node can detect whether any of its neighbors are beeping or not. Their analysis gives the guarantee that each node decides after $O((\log \Delta + \log 1/\varepsilon) \log 1/\varepsilon)$ rounds with probability at least $1 - \varepsilon$.

Proof of Lemma 2.3. By Lemma 2.4 and Lemma 2.5, the probability of a wrong move is at most 0.02. Each wrong move depends only on the randomness of the its own round. Thus, the events of wrong moves in different rounds are independent. Therefore, using a Chernoff bound, we can infer that over $T = C(\log \deg(v) + \log 1/\varepsilon)$ rounds, with probability at least $1 - \varepsilon/2$, we have at most $0.04T$ wrong moves. In the rest of this proof, we assume that the number of wrong moves is no more than $0.04T$. The $\varepsilon/2$ probability of the incorrectness of this assumption is handled via a union bound.

Let g_1 and g_2 be the number of golden rounds of type-1 and type-2, respectively. Also, let h be the number of rounds in which $d_t(v) > 0.02$. We show that, assuming at most $0.04T$ wrong moves, $g_2 < 0.05T$ implies $h < 0.27T$, which in turn implies $g_1 > 0.05T$, hence proving the lemma.

Small g_2 implies small h: Suppose that $g_2 \leq 0.05T$. Consider rounds in which $d_t(v) \geq 0.01$. Divide these rounds into two categories: (A1) those in which $d'_t(v) \geq 0.01 d_t(v)$ and (A2) those in which $d'_t(v) < 0.01 d_t(v)$. Note that the (A1) rounds are the same as golden type-2 rounds. In each (A1) round or an (A2) round with a wrong move, $d_t(v)$ increases by at most a 2 factor. There are at most $0.09T$ rounds such rounds, because $g_2 \leq 0.05T$ and the number of wrong moves are upper bounded to $0.04T$. In each (A2) round without a wrong move, d_t decreases by a 0.6 factor. Each 2 factor increase cancels the effect of at most two rounds with 0.6 factor decrease, because $(0.6)^2 < 1/2$. Thus, there are at most $0.18T$ rounds in (A2) without a wrong move. Therefore, in total, we have at most $0.27T$ rounds in which $d_t(v) > 0.02$. That is, $h < 0.27T$.

Small h implies large g_1: Assume that $h < 0.27T$, which means that in at least $0.73T$ rounds, we have $d_t(v) \leq 0.02$. In each of these rounds, if we have a wrong move, $p_{t+1}(v) = p_t(v)/2$, and otherwise, $p_{t+1}(v) = \min\{2p_t(v), \frac{1}{2}\}$. Thus, overall, we have at most $0.31T$ rounds in which $p_{t+1}(v) = p_t(v)/2$. This includes the at most $0.27T$ rounds in which $d_t(v) > 0.01$. Now, every round of halving $p_t(v)$ cancels the effect of at most one round of doubling $p_t(v)$. Since we start with $p_1(v) = 1/2$, we can conclude that there are at least $(1 - 2 \times 0.31)T$ rounds in which $p_1(v) = 1/2$. That is, there are at least $0.38T$ such rounds. Among these, at most $h < 0.27T$ can be rounds in which $d_t(v) > 0.02$. Therefore, there are at least $(0.38 - 0.27)T > 0.05T$ rounds in which $p_t(v) = 1/2$ and $d_t(v) \leq 0.02$. These are golden rounds of type-1 by definition. Hence, $g_1 \geq 0.05T$. □

2.3 Intermediate Algorithm (2): The Sparsified Beeping MIS Algorithm

Intuition

In the algorithm presented in the previous subsection, there is a clear difference between the amount of beeps in neighborhoods with small $d_t(v)$ and those with large $d_t(v)$. The former are good, because in a rough sense, the communications in those neighborhoods are *sparse*. On the other hand, the latter might be problematic, unless we do something about them. Here we explain intuitively what we want to do about each branch of this dichotomous situation. We then formalize this intuition in the algorithm and its analysis. We imagine that we divide the rounds of the algorithm into phases, each having $R = \sqrt{\delta \log n}/10$ rounds for a desirably small constant $\delta > 0$.

Sparsely Marked Neighborhoods: Consider one phase made of $R = \sqrt{\delta \log n}$ rounds, starting at a round t_0. Imagine a node v that has a small $d_{t_0}(v) \leq L$ for $L = 2^{\sqrt{\delta \log n}/5}$. In each round t of this phase, each neighbor u gets marked with probability $p_t(v) \leq \min\{1/2, 2^R p_{t_0}(u)\}$. This is because, per round their marking probability at most doubles. Hence, if at the beginning of the phase, for each round t in this phase, we *sample* the node u with probability $\min\{1/2, 2^R p_{t_0}(u)\}$, the set of sampled neighbors stochastically dominates the set of the nodes that will be actually marked. That is—in a sense to be formalized—the sampled set contains all the marked nodes, though potentially also much more other nodes.

On the positive side, this sampled set has size at most at most $2^{3\sqrt{\delta \log n}/10}$. Thus, if we locally gather these sampled sets, and then carefully work through the simulation of the markings only among these sampled nodes, we will be able to simulate the marking process in these neighborhoods with small $d_t(v)$. Since the sampled set has size at most $2^{3\sqrt{\delta \log n}/10}$, gathering the local topology induced by sampled nodes is essentially similar to learning the local topology in a graph with maximum degree $2^{3\sqrt{\delta \log n}/10}$. As we see later, this level of sparsity is sufficient for us.

Stabilizing Super-Heavy Neighborhoods: But what do we do about nodes with large $d_t(v)$? For these, the set of marked neighbors per round can be quite large. Let us call a node v *super-heavy* in a phase if in the first round t of that phase, we have $d_t(v) = \sum_{u \in N(v)} p_t(u) \geq L$ for $L = 2^{\sqrt{\delta \log n}/5}$. We argue that for each such super-heavy node v, to determine the behavior of v, we do not really need to simulate the markings of the neighbors of v within this phase — instead we can just assume that v always has beeping neighbors.

During the $R = \sqrt{\delta \log n}/10$ rounds of this phase, some of neighbors of v might get removed because of joining the MIS or having a neighbor join. But let us ignore that possibility for the moment. Then, per round $d_t(v)$ shrinks at most by a 2 factor, which means throughout the whole period, $d_t(v) \geq 2^{\sqrt{\delta \log n}/10} \gg \Theta(\log n)$. Hence, indeed in each of those rounds at least one neighbor of v is marked, w.h.p. Thus, v practically has no chance of entering the MIS during this phase.

For each node v that is super-heavy in a given phase, we will hedge our bets and assume that v will see marked neighbors in all rounds of this phase. Because of that, v has a perfectly predictable change in its $p_t(v)$. Namely, $p_t(v)$ decreases by a 2 factor in each round of this phase. This means we do not need to really simulate the effect of markings of the neighbors of v to compute its $p_t(v)$ throughout the phase. Recall that the above ignores the possibility of many neighbors of v getting kicked out during the phase, which could imply that v has rounds with no marked neighbor. We will show later in the analysis that ignoring this possibility does not slow us down significantly.

The Algorithm

We now present the sparsified variant of the algorithm. The concrete advantage of this sparsified algorithm will become clear later when we simulate it quickly in the congested clique model.

The Sparsified MIS Algorithm: The algorithm works in phases, each made of $P = \sqrt{\delta \log n}/10$ iterations and one additional round. At the beginning each phase, before the iterations, we have one round where each node sends its $p_t(v)$ to its neighbors. Then, node v sets $d_t(v) = \sum_{u \in N(v)} p_t(u)$ and if $d_t(v) \geq 2^{\sqrt{\delta \log n}/5}$, we call node v *super-heavy* throughout this phase. The algorithm then proceeds in iterations, each having two rounds, as follows:

(R1) In the first round of iteration t, each node v *beeps* with probability $p_t(v)$ and remains silent otherwise. As before, initially $p_1(v) = 1/2$. Now a node v joins MIS if three conditions are satisfied: (A) v is not super-heavy, (B) v beeps, and (C) all its neighbors are silent. Node v updates its beeping probability for the next iteration as follows:

$$p_{t+1}(v) = \begin{cases} p_t(v)/2, & \text{if node } v \text{ is super-heavy, or if} \\ & \text{at least one neighbor beeped} \\ \min\{2p_t(v), \frac{1}{2}\}, & \text{otherwise.} \end{cases}$$

(R2) In the second round of iteration t, all nodes in the MIS beep. If a non-MIS node hears a beep, it learns that it has an MIS neighbor. MIS nodes and their neighbors get removed from the problem, for the next iterations.

The Analysis

Again, the analysis will focus only on the first rounds of the iterations, and will ignore the second rounds which are there for basic clean up operations. With a slight abuse of terminology, in the sequel, we use *rounds* to refer to the first rounds of the corresponding iterations.

Theorem 2.6. *For each node v, during $T = C(\log \deg(v) + \log 1/\varepsilon)$ rounds, where C is a large enough constant, with probability at least $1 - \varepsilon$, either node v or a neighbor of v is added to MIS. Furthermore, this guarantee holds independent of the randomness outside 2-hop neighborhood of v.*

We start with redefining some notations. We use the notation $v \in \mathsf{SH}_t$ to indicate that node v is *super-heavy*, that is, it had $d_{t'}(v) \geq 2^{\sqrt{\delta \log n}/5}$ at the first round t' of the corresponding phase. We call a node v in round t *heavy* if either $v \in \mathsf{SH}_t$ or $d_t(v) > 10$. Define $d'_t(v)$ to be the summation of the beeping probabilities over non-heavy neighbors of v in round t, that is,

$$d'_t(v) = \sum_{u \in N(v),\, u \notin \mathsf{SH}_t,\, d_t(u) \leq 10} p_t(u).$$

Golden rounds are redefined as follows:

Golden Type-1 Round: $p_t(v) = 1/2, v \notin \mathsf{SH}_t, \& \, d_t(v) \leq 0.02$,

Golden Type-2 Round: $d_t(v) > 0.01$ and $d'_t(v) \geq 0.01 d_t(v)$.

One can see that these new definitions give us the property that, in the new algorithm, node v gets removed with at least a constant probability per golden round.

Lemma 2.7. *In each golden round, node v gets removed with at least a constant probability $\gamma > 0$.*

As before, the heart of the argument is to prove that there are many golden rounds for node v, which we show in Lemma 2.8. Then, Theorem 2.6 follows directly from Lemma 2.8 and Lemma 2.7.

Lemma 2.8. *For each node v, during $T = C(\log \deg(v) + \log 1/\varepsilon)$ rounds, where C is a large enough constant, with probability at least $1 - \varepsilon/2$, there are at least $0.05T$ golden rounds.*

To prove Lemma 2.8, as before, we have two helper lemmas, Lemma 2.9 and Lemma 2.9, which show that the system is more likely to move in the desired direction. In other words, they show that some *wrong moves* have small probability. Though, the statements are slightly different.

Lemma 2.9. *If $d_t(v) \leq 0.02$ and $v \notin \mathsf{SH}_t$, then with probability at least 0.98, we have $p_{t+1}(v) = \min\{2p_t(v), \frac{1}{2}\}$.*

Proof. Suppose that $d_t(v) \leq 0.01$ and $v \notin \mathsf{SH}_t$. If no neighbor of v beeps, then $p_{t+1}(v) = \min\{2p_t(v), \frac{1}{2}\}$. The probability of that is $\prod_{u \in N(v)}(1 - p_t(u)) \geq 4^{-\sum_{u \in N(v)} p_t(u)} \geq 0.98$. □

Lemma 2.10. *If $d_t(v) > 0.01$ and $d'_t(v) < 0.01 d_t(v)$, then with probability at least 0.98, we have $d_{t+1}(v) \leq 0.6 d_t(v)$.*

Proof. Suppose that $d_t(v) > 0.01$ and $d'_t(v) < 0.01 d_t(v)$. Hence, a 0.99 fraction of $d_t(v)$ is contributed by heavy neighbors of v. Notice that now a node u is heavy either because $u \in \mathsf{SH}_t$ or because $d_t(u) > 10$. The first category will reduce their $p_t(u)$ by a $1/2$ factor, as the algorithm indicates. The behavior of the second category is as analyzed before, they are more likely to reduce their $p_t(u)$ by a $1/2$ factor, but they also have a small chance of not doing that, and instead, even raising it up by to a 2 factor. The worst case of the increase in $d_{t+1}(v)$ would be if all heavy neighbors actually are in the second category, in which case the analysis follows the exact proof of Lemma 2.5, concluding that with probability at least 0.98, we have $d_{t+1}(v) < 0.6 d_t(v)$. □

Now that we are equipped with these two helper lemmas, we go back to proving Lemma 2.8.

Proof of Lemma 2.8. First note that by Lemma 2.4 and Lemma 2.5, the probability of a wrong move is at most 0.02. Each wrong move depends only on the randomness of the corresponding round. Thus, the events of wrong moves in different rounds are independent. Therefore, using a Chernoff bound, we can infer that over $T = C(\log \deg(v) + \log 1/\varepsilon)$ rounds, with probability at least $1 - \varepsilon/2$, we have at most $0.04T$ wrong moves. In the rest of this proof, we assume that the number of wrong moves is no more than $0.04T$. The $\varepsilon/2$ probability of the incorrectness of the assumption is taken into account later via a union bound.

Let g_1 and g_2 be the number of golden rounds of type-1 and type-2, respectively. Also, let h be the number of rounds in which $d_t(v) > 0.02$ or $v \in \mathsf{SH}_t$. We show that, assuming at most $0.04T$

wrong moves, $g_2 < 0.05T$ implies $h < 0.27T$, which in turn implies $g_1 > 0.05T$.

Small g_2 implies small h: Suppose that $g_2 \leq 0.05T$. We prove that, assuming at most $0.04T$ wrong moves, this implies $h < 0.27T$. Notice that h counts the number of rounds in which $v \in \mathsf{SH}_t$ or $d_t(v) > 0.01$. Let us first examine super-heavy rounds. For each phase in which v is super-heavy, if there is at least one round during the phase in which $d_t(v) \leq 0.02$, call that phase a *drop* phase. Notice that in a drop phase, at the beginning of the phase, we have $d_t(v) \geq 2^{\sqrt{\delta \log n}/5}$ and at the end of the phase, we have $d_t(v) \leq 0.02 \cdot 2^{\sqrt{\delta \log n}/10} \ll 2^{\sqrt{\delta \log n}/10}$. Hence, the phase observes a significant drop — at least a $2^{\sqrt{\delta \log n}/10}$ factor — in the value of $d_t(v)$. Amortized, this is even stronger than a $1/2$ factor decrease per each round of the phase. For the sake of our bookkeeping in the next paragraph, we bundle all these rounds together and simply imagine that each round of a drop phase gives a $1/2$ factor decrease in $d_t(v)$.

Aside from the rounds of drop phases, what remains in the count h are rounds which have $d_t(v) > 0.02$ (which may be in super-heavy phases or not). We consider two (slightly broader) classes of rounds, as before: (A1) rounds in which $d_t(v) > 0.01$ and $d'_t(v) \geq 0.01 d_t(v)$ and (A2) rounds in which rounds in which $d_t(v) > 0.01$ and $d'_t(v) < 0.01 d_t(v)$. Note that the (A1) rounds are the same as golden type-2 rounds and their count is g_2. In each (A1) round or an (A2) round with a wrong move, $d_t(v)$ increases by at most a 2 factor. There are at most $0.09T$ rounds such rounds. In each (A2) round without a wrong move, $d_t(v)$ decreases by a 0.6 factor. In each round in a drop phase, $d_t(v)$ decreases by a $1/2 < 0.6$ factor. Each 2 factor increase cancels the effect of at most two rounds with a 0.6 factor decrease, because $(0.6)^2 < 1/2$. Thus, there are at most $0.18T$ rounds in (A2) without a wrong move or in drop phases. Therefore, in total, we have at most $0.27T$ rounds in which $v \in \mathsf{SH}_t$ or $d_t(v) > 0.01$. That is, $h < 0.27T$.

Small h implies large g_1: Assume that $h < 0.27T$. This implies that in at least $0.73T$ rounds, we have $d_t(v) \leq 0.02$ and $v \notin \mathsf{SH}_t$. In each of these rounds, if we have a wrong move, $p_{t+1}(v) = p_t(v)/2$, and otherwise, $p_{t+1}(v) = \min\{2p_t(v), \frac{1}{2}\}$. Thus, overall, we have at most $0.31T$ rounds in which $p_{t+1}(v) = p_t(v)/2$. This includes the at most $0.27T$ rounds in which $d_t(v) > 0.02$ or $v \in \mathsf{SH}_t$. Now, every round of halving $p_t(v)$ cancels the effect of at most one round of doubling $p_t(v)$. Since we start with $p_1(v) = 1/2$, we can conclude that there are at least $(1 - 2 \times 0.31)T$ rounds in which $p_t(v) = 1/2$. That is, there are at least $0.38T$ such rounds. Among these, at most $h < 0.27T$ can be rounds in which $d_t(v) > 0.02$ or $v \in \mathsf{SH}_t$. Therefore, there are at least $(0.38 - 0.27)T > 0.05T$ rounds in which we have the following three properties: $p_t(v) = 1/2$, $v \notin \mathsf{SH}_t$, and $d_t(v) \leq 0.02$. These are golden rounds of type-1 by definition. Hence, $g_1 \geq 0.05T$. □

Lemma 2.11. *The graph remaining after $\Theta(\log \Delta)$ rounds of the sparsified MIS algorithm has at most $O(n)$ edges, w.h.p.*

Proof. As can be inferred from Theorem 2.6, after $\Theta(\log \Delta)$ rounds, each vertex remains with probability at most $1/\Delta^{50}$. If $\Delta \geq n^{0.1}$, w.h.p., no edge remains. Suppose $\Delta \leq n^{0.1}$. Then, we expect at most $\frac{n\Delta}{\Delta^{50}} \ll n$ remaining edges. Whether a node v remains is independent of the randomness outside its 2-hop neighborhood. Thus, each edge's event of remaining depends on no more than $2\Delta^5 \leq n^{0.6}$ edges. By applying the extension of the Chernoff bound to settings with bounded dependency[37], we get that, w.h.p., at most $O(n)$ edges remain. □

2.4 Simulation in Congested Clique

We now describe our distributed algorithm that computes an MIS in $\tilde{O}(\sqrt{\log \Delta})$ rounds of congested clique, thus proving Theorem 1.1.

Overall, the algorithm has two parts. The first part is the main ingredient, the second is a simple and standard clean up step. In the first part, we simulate $O(\log \Delta)$ rounds of the sparsified variant of the beeping-based MIS algorithm, presented in Section 2.3, but much faster, in just $\tilde{O}(\sqrt{\log \Delta})$ rounds of congested clique. The first part may leave a small graph, which as proven by Lemma 2.11 can have at most $O(n)$ edges, with high probability. In the second part, we explain how to solve this remaining graph in $O(1)$ additional rounds of the congested clique.

The First Part of the Congested Clique Algo.

As outlined above, the heart of the algorithm is the first part, that is, faster simulation of the sparsified algorithm in the congested clique. Recall that the sparsified algorithm is made of phases, each having $\sqrt{\delta \log n}/10$ rounds. We explain how to simulate each of these phases in merely $O(\log \log \Delta)$ rounds of the congested clique. Thus, overall, simulating $O(\log \Delta)$ rounds of the sparsified algorithm takes $O(\frac{\log \Delta \cdot \log \log \Delta}{\sqrt{\delta \log n}})$ rounds in the congested clique model. We next explain how the simulation of each phase works.

Before that, we remark that in this subsection, without loss of generality, we will assume that $\log \log \Delta = \Theta(\log \log n)$. This is without loss of generality, because in the complementary case, there is a simpler variant of our approach that solves the problem in $O(\log \log \Delta)$ rounds of the congested clique, see Lemma 2.15.

Simulating One Phase in $\Theta(\log \log n) = O(\log \log \Delta)$ Rounds: The first round of the phase, which is simply exchanging the initial values of $p_t(v)$, can be performed in one round. We explain how to simulate the rest of the phase, which is made of $\frac{\sqrt{\delta \log n}}{10}$ iterations, in $O(\log \log \Delta)$ rounds. At the end of the simulation, we want each node v to know two things about its configuration at the end of the phase: (1) whether v is in the MIS or not, (2) the value of $p_t(v)$ for the last round of the phase. Given this, in one additional round, we can remove the MIS nodes and those adjacent to them, and then proceed to simulating the next iteration.

We now discuss how we simulate the iterations. Since we are dealing with randomized algorithms, it is crucial to be formal what a simulation means. We disentangle the randomness from the simulation, as follows: note that the only randomized step in the algorithm is deciding whether to beep or not, with probability $p_t(v)$ for beeping. We imagine that for this, node v has a random variable $r_t(v)$ chosen uniformly at random from $[0, 1]$, with $\Theta(\log \Delta)$ bits of precision. Then, if $r_t(v) \leq p_t(v)$, node v beeps, and otherwise, it remains silent. To separate the randomness from the simulation, we imagine that each node v draws all of its random values $r_t(v)$ for all rounds t of this phase at the beginning of the phase. Having fixed these values, we are left with simulating a deterministic procedure.

First, notice that the simulation for the super-heavy nodes is straightforward. For each super-heavy node v, its $p_t(v)$ will decrease by a 2 factor in each round. Let each super-heavy node locally calculate its $p_t(v)$ for the rounds of this phase, and then determine whether it beeps or not in each round, according to these calculated probabilities $p_t(v)$ and its random values $r_t(v)$ by checking whether $r_t(v) \leq p_t(v)$ or not. These outcomes of these decisions can be represented in a binary string of $\sqrt{\delta \log n}/10$ bits, one bit per round. We call this the *beep vector* of node v for this phase. We use one round to have super-heavy nodes send their *beep vectors* to all their neighbors.

What remains is simulating other nodes, who are not super heavy and thus at the beginning of the phase have $d_t(v) \leq 2^{\sqrt{\delta \log n}/5}$. Consider the following subset S of vertices. Let t_0 be the first round the phase. Include each node v in S if there is at least one round t of this phase for which $r_t(v) \leq 2^{\sqrt{\delta \log n}/10} p_{t_0}(v)$. Notice that if $v \notin S$, then v does not beep during this phase. However, the converse is not true and it is possible that v is included in the set but does not beep in this phase. Therefore, S is a super-set of nodes that beep during this phase. In particular, S includes all nodes that join the MIS, but most likely also many other nodes.

The set S has one extremely useful property: with high probability, the subgraph $G[S]$ induced by S is locally sparse and in particular, it has a small maximum degree.

Lemma 2.12. *W.h.p., each node $s \in S$ has no more than $2^{1+\sqrt{\delta \log n}/2}$ neighbors in S.*

We decorate the vertices of $G[S]$ with some additional information, and call the resulting decorated graph $G_*[S]$. For each node $v \in S$, in $G_*[S]$, we record two additional items of information, besides the edges of v: (1) the bit-wise OR of the beep vectors that v received from its super-heavy neighbors, and (2) the values of $r_t(v)$ for all the rounds of this phase. Notice that overall this is no more than $O(\log^2 n)$ bits of information per node. It is convenient to imagine that these bits are appended to the ID of the vertices, and we now have a graph $G_*[S]$ with IDs of size $O(\log^2 n)$ bits. The next lemma shows that once a node $s \in S$ learns a small neighborhood around itself in G_*S, it can simulate its behavior in this phase locally. The proof appears in Lemma 2.13.

Lemma 2.13. *If each node $s \in S$ learns its $\sqrt{\delta \log n}/10$-hop neighborhood in $G_*[S]$, then node s can simulate its behavior in this phase locally. In particular, node s learn two things: (1) whether s joined the MIS in this phase or not, (2) the value of $p_t(s)$ at the end of the phase.*

Proof of Lemma 2.13. The proof is by a simple induction. We argue that for any k, the behavior of each node $v \in S$ in the first k rounds of the phase can be determined from its k-hop neighborhood in $G_*[S]$. The base of the induction where $k = 0$ is trivial. To know the behavior of node v in the first $k+1$ rounds, we need to know two things: (1) the behavior of node v in the first k rounds, and (2) for each neighbor u of v in G, whether u beeps in the $(k+1)^{th}$ of the phase or not. The latter can be divided into three categories: (A) if u is super-heavy, then node v already has the beeping vector of u and thus it knows whether u beeps in the $(k+1)^{th}$ round or it. (B) if u is not super-heavy and $u \notin S$, then u clearly does not beep in any round of the phase and thus also not in the $(k+1)^{th}$ round of the phase. (C) if $u \in S$, then whether u beeps or not in the $(k+1)^{th}$ round can be determined by knowing the behavior of u in the first k rounds, which dictates $p_t(u)$ in that round, and $r_t(u)$. The latter is in the decoration of u, and the former can be learned from the k-hop neighborhood of u in $G_*[S]$, by the induction assumption. Hence, by knowing the $k+1$-hop neighborhood of v in $G_*[S]$, node v can determine its behavior in the first $k+1$ rounds. □

Lemma 2.14. *There is an $O(\log \log n)$ round algorithm that allows each node $s \in S$ to learns its $\sqrt{\delta \log n}/10$-hop neighborhood in $G_*[S]$.*

We have been informed that the simple *graph exponentiation* idea that we use for proving this lemma has been used before, and in this regard, one can view as reproving a brief announcement result of Lenzen and Wattenhofer [26]. Regardless, to be self-contained, we include a proof here, especially as it is short and simple.

Proof Lemma 2.14. First notice that the size of the neighborhood to be learned by each node is at most

$$\left(2^{1+\sqrt{\delta \log n}/2} \cdot \log^2 n\right)^{2\sqrt{\delta \log n}/10} \ll n^{\delta/5}.$$

Let H be an arbitrary n-node graph with maximum degree at most $n^{\delta/4}$ and suppose that each node v knows its neighbors, i.e. the edges incident on it. We first explain how in $O(1)$ rounds, we can make each node v know all the edges incident on its neighbors. This will in particular all the node to know H^2, the graph where there is an edge between each two nodes with distance at most 2.

The procedure on H is as follows: each vertex v with degree $deg(v)$ prepares $deg^2(v)$ packets, each describing one of its edges, and to be sent to one of its neighbors. Thus, each vertex v is the source of $(deg(v))^2 \leq n^{\delta/2}$ packets. Moreover, each node v is the destination for at most n packets, as node v has at most $deg(v) \leq n^{\delta/4}$ neighbors and v is the destination for $deg(u) \leq n^{\delta/4}$ packets beginning at each neighbor u of v. Thus, each node is the source or destination for at most $n^{\delta/2}$ packets. We apply Lenzen's routing method[25] to deliver all these packets from their sources to their destinations in just $O(1)$ round of the congested clique.

We apply the above procedure recursively on graphs $G_*[S]$, $G_*^2[S]$, $G_*^4[S]$, $G_*^8[S]$ …, $G_*^\kappa[S]$, where κ is the least power of 2 no less than $\sqrt{\delta \log n}/10$. The procedure is applicable as the maximum degree in each of these graphs is at most $(2^{1+\sqrt{\delta \log n}/2} \cdot \log^2 n)^\kappa \leq n^{\delta/4}$. At the end, each node learns its $\sqrt{\delta \log n}/10$-neighborhood in $G_*[S]$. Moreover, this whole scheme takes $O(\log \kappa) = O(\log \log n)$ rounds, that is, $O(1)$ rounds for each recursive application. □

The Second Part of the Congested Clique Algo.

After simulating $O(\log \Delta)$ rounds of the CONGEST-model algorithm, which computes a nearly-maximal independent set S, a number of nodes may remain. Let B be these remaining nodes. The set B is the set of all nodes who have no neighbor in the computed nearly-maximal independent set independent set S. Note that each node has probability at most $1/\Delta^{10}$ to be in B. As Lemma 2.11 proves, the subgraph induced by B has at most $O(n)$ edges. This remaining graph $G[B]$ can be handled easily, via a standard idea in the congested clique. In particular, we make each node in B send its $G[B]$ edges to the leader node (which can be elected in $O(1)$ round). This can be done in $O(1)$ rounds, using Lenzen's routing method[25]. At

the end, the leader computes an MIS S_B of $G[B]$ and informs those MIS nodes. Then, the overall MIS is $S \cup S_B$.

2.5 Faster MIS in Low-Degree Graphs

Lemma 2.15. *Suppose that $\Delta \leq 2^{c\sqrt{\delta \log n}}$ for a sufficiently small constant c. Then, there is a distributed algorithm that computes an MIS in merely $O(\log\log\Delta)$ rounds of the congested clique.*

Proof Sketch. Each node can learn its $O(\log\Delta)$-hop neighborhood in G via Lemma 2.14 in $O(\log\log\Delta)$ rounds. This is possible because the related $O(\log\Delta)$ neighborhood has size at most n^{δ}. Then, the nodes can simulate $O(\log\Delta)$ rounds of the MIS algorithm of [13] locally. The remaining graph would have $O(n)$ edges and could be solved using $O(1)$ rounds as explained in the second part of the algorithm. □

Acknowledgment: I am grateful to Merav Parter. Our discussions about her recent work with Keren Censor-Hillel and Gregory Schwartzman[9] — which presents an $O(\log\Delta\log n)$ round congested clique derandomization of the $O(\log\Delta)$ round MIS algorithm of [13] — reignited my interest in looking for a sublogarithmic MIS algorithm in the congested clique model. I am also thankful to Stephan Holzer and Nancy Lynch. Our discussions about their (half-duplex) beeping model MIS algorithm [20] and its *local complexity* guarantees made me try to find a (tighter) characterization of the *local complexity* of the beeping MIS algorithm of [39], as discussed in Section 2.2. Though, at the time, I did not expect that this improvement and more generally a beeping MIS algorithm could be useful as an intermediate step towards a faster congested clique algorithm; in fact, even in hindsight, I find this extraordinary. I am also thankful for the valuable feedback and comments by the anonymous reviewers of PODC17.

REFERENCES

[1] Yehuda Afek, Noga Alon, Ziv Bar-Joseph, Alejandro Cornejo, Bernhard Haeupler, and Fabian Kuhn. 2013. Beeping a maximal independent set. *Distributed computing* 26, 4 (2013), 195–208.
[2] Yehuda Afek, Noga Alon, Omer Barad, Eran Hornstein, Naama Barkai, and Ziv Bar-Joseph. 2011. A biological solution to a fundamental distributed computing problem. *science* 331, 6014 (2011), 183–185.
[3] Noga Alon, László Babai, and Alon Itai. 1986. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of algorithms* 7, 4 (1986), 567–583.
[4] Noga Alon, Ronitt Rubinfeld, Shai Vardi, and Ning Xie. 2012. Space-efficient local computation algorithms. In *Pro. of ACM-SIAM Symp. on Disc. Alg. (SODA)*. 1132–1139.
[5] Leonid Barenboim, Michael Elkin, Seth Pettie, and Johannes Schneider. 2012, also coRR abs/1202.1983v3. The locality of distributed symmetry breaking. In *Foundations of Computer Science (FOCS) 2012*. IEEE, 321–330.
[6] Florent Becker, Antonio Fernandez Anta, Ivan Rapaport, and Eric Reémila. 2015. Brief Announcement: A Hierarchy of Congested Clique Models, from Broadcast to Unicast. In *the Proc. of the Int'l Symp. on Princ. of Dist. Comp. (PODC)*. ACM, 167–169.
[7] Andrew Berns, James Hegeman, and Sriram V Pemmaraju. 2012. Super-fast distributed algorithms for metric facility location. In *the Pro. of the Int'l Colloquium on Automata, Languages and Programming (ICALP)*. 428–439.
[8] Keren Censor-Hillel, Petteri Kaski, Janne H. Korhonen, Christoph Lenzen, Ami Paz, and Jukka Suomela. 2015. Algebraic Methods in the Congested Clique. In *the Proc. of the Int'l Symp. on Princ. of Dist. Comp. (PODC)*. ACM, 143–152.
[9] Keren Censor-Hillel, Merav Parter, and Gregory Schwartzman. 2016. Derandomizing Local Distributed Algorithms under Bandwidth Restrictions. *arXiv preprint arXiv:1608.01689* (2016).
[10] Alejandro Cornejo and Fabian Kuhn. 2010. Deploying wireless networks with beeps. In *International Symposium on Distributed Computing*. Springer, 148–162.
[11] Danny Dolev, Christoph Lenzen, and Shir Peled. 2012. âĂIJTri, Tri Againâ Ăİ: Finding Triangles and Small Subgraphs in a Distributed Setting. In *Distributed Computing*. Springer, 195–209.
[12] Andrew Drucker, Fabian Kuhn, and Rotem Oshman. 2014. On the Power of the Congested Clique Model. In *the Proc. of the Int'l Symp. on Princ. of Dist. Comp. (PODC)*. ACM, 367–376.
[13] Mohsen Ghaffari. 2016. An Improved Distributed Algorithm for Maximal Independent Set. In *Pro. of ACM-SIAM Symp. on Disc. Alg. (SODA)*.
[14] Mohsen Ghaffari and Bernhard Haeupler. 2013. Near optimal leader election in multi-hop radio networks. In *Pro. of ACM-SIAM Symp. on Disc. Alg. (SODA)*. 748–766.
[15] Mohsen Ghaffari and Merav Parter. 2016. MST in Log-Star Rounds of Congested Clique. In *the Proc. of the Int'l Symp. on Princ. of Dist. Comp. (PODC)*.
[16] James W. Hegeman, Gopal Pandurangan, Sriram V. Pemmaraju, Vivek B. Sardeshmukh, and Michele Scquizzato. 2015. Toward Optimal Bounds in the Congested Clique: Graph Connectivity and MST. In *the Proc. of the Int'l Symp. on Princ. of Dist. Comp. (PODC)*. ACM, 91–100.
[17] James W Hegeman and Sriram V Pemmaraju. 2014. Lessons from the congested clique applied to MapReduce. In *the Proceedings of the International Colloquium on Structural Information and Communication Complexity*. Springer, 149–164.
[18] James W Hegeman, Sriram V Pemmaraju, and Vivek B Sardeshmukh. 2014. Near-constant-time distributed algorithms on a congested clique. In *Proc. of the Int'l Symp. on Dist. Comp. (DISC)*. Springer, 514–530.
[19] Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. 2016. A deterministic almost-tight distributed algorithm for approximating single-source shortest paths. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing*. ACM, 489–498.
[20] Stephan Holzer and Nancy Lynch. 2016. Brief Announcement - Beeping a Maximal Independent Set Fast. In *Proc. of the Int'l Symp. on Dist. Comp. (DISC)*.
[21] Stephan Holzer and Nancy Lynch. April, 2017. Beeping a Maximal Independent Set Fast. *preprint arXiv:1704.07133* (April, 2017).
[22] Janne H Korhonen. 2016. Deterministic MST Sparsification in the Congested Clique. *arXiv preprint arXiv:1605.02022* (2016).
[23] Fabian Kuhn, Thomas Moscibroda, and Rogert Wattenhofer. 2004. What Cannot Be Computed Locally!. In *the Proc. of the Int'l Symp. on Princ. of Dist. Comp. (PODC)*. 300–309.
[24] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. 2016. Local Computation: Lower and Upper Bounds. *J. ACM* 63, 2, Article 17 (March 2016), 44 pages.
[25] Christoph Lenzen. 2013. Optimal Deterministic Routing and Sorting on the Congested Clique. In *the Proc. of the Int'l Symp. on Princ. of Dist. Comp. (PODC)*. 42–50.
[26] Christoph Lenzen and Roger Wattenhofer. 2010. Brief Announcement: Exponential Speed-up of Local Algorithms Using Non-local Communication. In *the Proc. of the Int'l Symp. on Princ. of Dist. Comp. (PODC)*. ACM, New York, NY, USA, 295–296.
[27] Nathan Linial. 1987. Distributive graph algorithms Global solutions from local data. In *Proc. of the Symp. on Found. of Comp. Sci. (FOCS)*. IEEE, 331–335.
[28] Nathan Linial. 1992. Locality in distributed graph algorithms. *SIAM J. Comput.* 21, 1 (1992), 193–201.
[29] Zvi Lotker, Boaz Patt-Shamir, Elan Pavlov, and David Peleg. 2005. Minimum-weight spanning tree construction in O(log log n) communication rounds. *SIAM J. Comput.* 35, 1 (2005), 120–131.
[30] Zvi Lotker, Elan Pavlov, Boaz Patt-Shamir, and David Peleg. 2003. MST construction in O(log log n) communication rounds. In *the Proceedings of the Symposium on Parallel Algorithms and Architectures*. ACM, 94–100.
[31] Michael Luby. 1985. A simple parallel algorithm for the maximal independent set problem. In *Proc. of the Symp. on Theory of Comp. (STOC)*. 1–10.
[32] Danupon Nanongkai. 2014. Distributed Approximation Algorithms for Weighted Shortest Paths. In *Proc. of the Symp. on Theory of Comp. (STOC)*.
[33] Saket Navlakha and Ziv Bar-Joseph. 2015. Distributed information processing in biological and computational systems. *Commun. ACM* 58, 1 (2015), 94–102.
[34] Michal Parnas and Dana Ron. 2007. Approximating the minimum vertex cover in sublinear time and a connection to distributed algorithms. *Theoretical Computer Science* 381, 1 (2007), 183–196.
[35] Boaz Patt-Shamir and Marat Teplitsky. 2011. The round complexity of distributed sorting. In *the Proc. of the Int'l Symp. on Princ. of Dist. Comp. (PODC)*. 249–256.
[36] David Peleg. 2000. *Distributed Computing: A Locality-sensitive Approach*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.
[37] Sriram V. Pemmaraju. 2001. Equitable Colorings Extend Chernoff-Hoeffding Bounds. In *Pro. of ACM-SIAM Symp. on Disc. Alg. (SODA)*. 924–925.
[38] R. Rubinfeld, G. Tamir, S. Vardi, and N. Xie. 2011. Fast Local Computation Algorithms. In *Proc. 2nd Symp. on Innovations in Computer Science (ICS)*. 223–238.
[39] Alex Scott, Peter Jeavons, and Lei Xu. 2013. Feedback from nature: an optimal distributed algorithm for maximal independent set selection. In *the Proc. of the Int'l Symp. on Princ. of Dist. Comp. (PODC)*. 147–156.