Distributed Computing (2020) 33:349–366
https://doi.org/10.1007/s00446-020-00376-1

Derandomizing local distributed algorithms under bandwidth restrictions

Keren Censor-Hillel[1] · Merav Parter[2] · Gregory Schwartzman[3]

Received: 21 August 2018 / Accepted: 27 March 2020 / Published online: 18 April 2020

Abstract
This paper addresses the cornerstone family of *local problems* in distributed computing, and investigates the curious gap between randomized and deterministic solutions under bandwidth restrictions. Our main contribution is in providing tools for derandomizing solutions to local problems, when the n nodes can only send $O(\log n)$-bit messages in each round of communication. Our framework mostly follows by the derandomization approach of Luby (J Comput Syst Sci 47(2):250–286, 1993) combined with the power of all to all communication. Our key results are as follows: first, we show that in the *congested clique* model, which allows all-to-all communication, there is a deterministic maximal independent set algorithm that runs in $O(\log^2 \Delta)$ rounds, where Δ is the maximum degree. When $\Delta = O(n^{1/3})$, the bound improves to $O(\log \Delta)$. In addition, we deterministically construct a $(2k-1)$-spanner with $O(kn^{1+1/k} \log n)$ edges in $O(k \log n)$ rounds in the congested clique model.

1 Introduction

1.1 Motivation

A cornerstone family of problems in distributed computing are the so-called *local problems*. These include finding a maximal independent set (MIS), a $(\Delta + 1)$-coloring where Δ is the maximal degree in the network graph, finding a maximal matching, constructing multiplicative spanners, and more. Intuitively, as opposed to *global problems*, local problems admit solutions that do not require communication over the entire network graph.

One fundamental characteristic of distributed algorithms for local problems is whether they are deterministic or randomized. Currently, there exists a curious gap between the known complexities of randomized and deterministic solutions for local problems. Interestingly, the main indistinguishability-based technique used for obtaining the relatively few lower bounds that are known seems unsuitable for separating these cases. A beautiful recent work of Chang et al. [15] sheds some light over this, by proving that the randomized complexity of any local problem is at least its deterministic complexity on instances of size $\sqrt{\log n}$. In addition, building upon a new lower bound technique of Brandt et al. [11], they show an exponential separation between the randomized and deterministic complexity of Δ-coloring trees. These results hold in the *LOCAL* model, which allows unbounded messages.

In this paper, we address the tension between the deterministic and randomized complexities of local problems in the *congested clique* model, where the communication graph is complete but the size of messages is restricted to $O(\log n)$ bits. The processed graph is an arbitrary input graph which, in contrast to the LOCAL model, is not necessarily the same as the communication graph. In some sense, the congested

A preliminary version of these results appeared in the Proceedings of The 31st International Symposium on Distributed Computing (DISC), pages 11:1–11:16, 2017.

Keren Censor-Hillel: Supported in part by the Israel Science Foundation (Grant 1696/14). This project has received funding from the European Union's Horizon 2020 Research And Innovation Program under grant Agreement No. 755839. Gregory Schwartzman: This work was supported by JSPS KAKENHI Grant Numbers 19K20216 and JP18H05291.

✉ Merav Parter
merav.parter@weizmann.ac.il

Keren Censor-Hillel
ckeren@cs.technion.ac.il

Gregory Schwartzman
gregory.schwartzman@gmail.com

[1] Technion, Haifa, Israel

[2] Weizmann Institute, Rehovot, Israel

[3] NII, Tokyo, Japan

clique model is orthogonal to the LOCAL model, because the diameter of the communication graph is 1, but the size of messages is restricted. By showing how to derandomize known algorithms for the LOCAL model, we provide fast deterministic algorithms for constructing an MIS and multiplicative spanners in the congested clique model.

Our starting observation concerns the improved randomized complexity of many local problem in the CONGEST model. The state-of-the-art algorithms for most classical local problems such as MIS, $(\Delta + 1)$ coloring, and maximal matching, follow a two phase structure: a randomized phase which makes significant progress in parts of the graph, followed by a deterministic phase that solves the remaining unsolved pieces deterministically. The efficiency of this approach is due to a shattering phenomena [7] which informally refers to the following situation. At the end of the randomized phase, the unsolved pieces are "small" (of poly-logarithmic size), and therefore these pieces can be solved in parallel within $Det(\text{poly}(\log n))$ rounds, where $Det(n')$ is the deterministic round complexity of the problem in graphs with n' nodes. This is the source of the additive $2^{O(\sqrt{\log \log n})}$ term in the current randomized time complexity for many of the local problems. We show that in the congested clique model the shattering effect is even stronger in the sense that the remaining unsolved part can be solved in $O(1)$ rounds. This is shown here for the MIS problem, and the same approach has been taken later for other problems (e.g., $(\Delta+1)$ coloring) in subsequent papers [14,51,52].

Equipped with the improved shattering complexity, in this paper we mainly zoom into derandomizing the randomized phase. Our approach is based on the derandomization toolbox that was developed for parallel algorithms by [45,46] adapted to the congested clique model. As we will see, in certain cases, the power of all-to-all communication allows one to discover a chunk of $\log n$ bits in the derandomized seed using just $O(1)$ number of rounds.

1.2 Our contribution

Maximal independent set (MIS) We begin by derandomizing the MIS algorithm of Ghaffari [26], which runs in $O(\log \Delta) + 2^{O(\sqrt{\log \log n})}$ rounds, w.h.p.[1] In a nutshell, this algorithm works in constant-round phases, in which nodes choose to mark themselves with probabilities that evolve depending on the previous probabilities of neighbors. A marked node that does not have any marked neighbors joins the MIS and all of its neighbors remove themselves from the graph. The analysis shows that after $O(\log \Delta)$ phases the remaining subgraph consists of a convenient decomposition into small clusters for which the problem can be solved fast.

[1] As standard, *with high probability* means with probability that is at least $1 - 1/n^c$ for a constant c.

We first show that a tighter analysis for the congested clique model of Ghaffari's MIS algorithm can improve its running time from $O(\log \Delta + \log^* n)$ (which follows from combining [26] with the new connectivity result of [28]) to $O(\log \Delta)$ rounds.

Theorem 1 *There is a randomized algorithm that computes MIS in the congested clique model within $O(\log \Delta)$ rounds with high probability.*

The first key ingredient is a slight modification of the constants used by Ghaffari's algorithm. Ghaffari's analysis is based on a definition of *golden nodes*, which are nodes that have a constant probability of being removed in the given phase. We show that, after our slight adaptation of the constants used by the algorithm, this removal-probability guarantee holds also with pairwise independence.

Second, the shattering effect that occurs after $O(\log \Delta)$ rounds of Ghaffari's algorithm with *full independence*, no longer holds under pairwise independence. Instead, we take advantage of the fact that in the congested clique model, once the remaining graph has a linear number of edges then the problem can be solved locally in constant many rounds using Lenzen's routing algorithm [40]. Thus, we modify the algorithm so that after $O(\log \Delta)$ rounds, the remaining graph (containing all undecided nodes) contains $O(n)$ edges. The crux in obtaining this is that during the first $O(\log \Delta)$ phases, we favor the removal of *old* nodes, which, roughly speaking, are nodes that had many rounds in which they had a good probability of being removed. This prioritized (or biased) removal strategy allows us to employ an amortized (or accounting) argument to claim that every node that survives $O(\log \Delta)$ rounds, can blame a distinct set of Δ nodes for not being removed earlier. Hence, the total number of remaining nodes is bounded by $O(n/\Delta)$, implying a remaining number of edges of $O(n)$.

To simulate the $O(\log \Delta)$ randomized rounds of Ghaffari's algorithm, we enjoy the small search space (due to pairwise independence) and employ the method of conditional expectations on a random seed of length $O(\log n)$. This follows the same approach of Luby [45].

Note that once we start conditioning on random variables in the seed, the random choices are no longer pairwise independent as they are in the unconditioned setting. However, we do not use the pairwise independence in the conditioning process. That is, the pairwise independence is important in showing that the *unconditional expectation* is large, and from that point on the conditioning does not reduce this value. As typical in MIS algorithms, the probability of a node being removed stems from the random choices made in its 2-neighborhood. With a logarithmic bandwidth, collecting this information is too costly. Instead, we use a pessimistic estimator to *bound* the conditional probabilities rather than compute them. Interestingly, despite the fact that our algo-

rithm derandomizes a different algorithm (that is, Ghaffari's algorithm rather than Luby's), we can still use the clever approach of Luby [45] in the definition of the local computation of the pessimistic estimator .

Finally, to make the decision of the partial assignment and inform the nodes, we leverage the power of the congested clique by having a leader that collects the relevant information for coordinating the decision regarding the partial assignment. Carefully placing all the pieces of the toolbox we develop, gives the following.

Theorem 2 *There is a deterministic MIS algorithm for the congested clique model that completes in* $O(\log \Delta \log n)$ *rounds.*

If the maximal degree satisfies $\Delta = O(n^{1/3})$ then we can improve the running time in the congested clique model.

Theorem 3 *If* $\Delta = O(n^{1/3})$ *then there is a deterministic MIS algorithm for the congested clique model that completes in* $O(\log \Delta)$ *rounds.*

Combining Theorems 2 and 3 directly gives that the complexity is either $O(\log \Delta)$ rounds in case $\Delta = O(n^{1/3})$, and otherwise it is $O(\log^2 \Delta)$ since $\log n$ is then asymptotically equal to $\log \Delta$.

Corollary 1 *There is a deterministic MIS algorithm for the congested clique model that completes in* $O(\log^2 \Delta)$ *rounds.*

Our techniques immediately extend to the CONGEST model. We show that MIS can be computed in $O(D \cdot \log^2 n)$ rounds where D is the diameter of the graph. Here, we simulate $O(\log n)$ rounds of Ghaffari's algorithm rather than $O(\log \Delta)$ rounds as before. Each such randomized round is simulated by using $O(D \cdot \log n)$ deterministic rounds in which the nodes compute an $O(\log n)$ seed. Computing each bit of the seed requires aggregation of the statistics to a leader, which can be done in $O(D)$ rounds, and since the seed is of length $O(\log n)$, we have the following:

Theorem 1.5 *There is a deterministic MIS algorithm for the CONGEST model that completes in* $O(D \log^2 n)$ *rounds.*

This provides a complete proof for Theorem 2 in [2] that claimed a deterministic $\widetilde{O}(D)$-round for MIS using the approach of Luby [45].

Multiplicative spanners We further exemplify our techniques in order to derandomize the Baswana–Sen algorithm for constructing a multiplicative spanner. For an integer k, a k-spanner S of $G = (V, E)$ is a subgraph (V, E_S) such that for every two neighbors v, u in G, their distance in S is at most k. This implies that also the distance for every other pair of nodes is stretched in S by no more than a multiplicative factor of k. The Baswana–Sen algorithm runs in $O(k^2)$ rounds and produces a $(2k-1)$-spanner with $O(kn^{1+1/k})$ edges. In a nutshell, the algorithm starts with a clustering defined by all singletons and proceeds with k iterations, in each of which the clusters get sampled with probability $n^{-1/k}$ and each node joins a neighboring sampled cluster or adds edges to unsampled clusters.

We need to make several technical modifications of our tools for this to work. The key technical difficulty is that we cannot have a single target function. This arises from the very nature of spanners, in that a small-stretch spanner always exists, but the delicate part is to balance between the stretch and the number of edges. This means that a single function which takes care of having a good stretch alone will simply result in taking all the edges into the spanner, as this gives the smallest stretch. We overcome this challenge by defining two types of bad events which the algorithm tries to avoid simultaneously. One is that too many clusters get sampled, and the other is that too many nodes add too many edges to the spanner in this iteration. The careful balance between the two promises that we can indeed get the desired stretch and almost the same bound on the number of edges.

Additional changes we handle are that when we reduce the independence, we cannot go all the way down to pairwise independence and we need to settle for d-wise independence, where $d = \Theta(\log n)$. Further, we can improve the iterative procedure to handle chunks of $\log n$ random bits, and evaluate them in parallel by assigning a different leader to each possible assignment for them. A careful analysis gives a logarithmic overhead compared to the original Baswana–Sen algorithm, but we also save a factor of k since the congested clique allows us to save the k rounds needed in an iteration of Baswana–Sen for communicating with the center of the cluster. This gives the following.

Theorem 1.6 *There is a deterministic algorithm for the congested clique model that completes in* $O(k \log n)$ *rounds and produces a* $(2k-1)$*-spanner with* $O(kn^{1+1/k} \log n)$ *edges.*

The above algorithm works also in the broadcast congested clique model, albeit here we lose the ability to parallelize over many leaders and thus we pay another logarithmic factor in the number of rounds, resulting in $O(k \log^2 n)$ rounds.

1.3 Related work

Distributed computation of MIS The complexity of finding a maximal independent set is a central problem in distributed computing and hence has been extensively studied. The $O(\log n)$-round randomized algorithms date back to 1986, and were given by Luby [44], Alon et al. [1] and Israeli and Itai [36] (the latter for maximal matching). Barenboim et al. [7] showed a randomized MIS algorithm with $O(\log^2 \Delta) + 2^{O(\sqrt{\log \log n})}$ rounds. They also showed

the bound of $O(\log \Delta) + 2^{O(\sqrt{\log \log n})}$ rounds for Maximal Matching and $(\Delta+1)$-coloring. Following [7], a recent breakthrough by Ghaffari [26] obtained a randomized algorithm in $O(\log \Delta) + 2^{O(\sqrt{\log \log n})}$ rounds. Recently, Ghaffari [27] also provided an improved randomized MIS algorithm in the congested clique model that completes in $\widetilde{O}(\sqrt{\log \Delta})$ rounds.

The best deterministic algorithm is by Panconesi and Srinivasan [49], and completes in $2^{O(\sqrt{\log n})}$ rounds. On the lower bound side, Linial [42] gave an $\Omega(\log^* n)$ lower bounds for 3-coloring the ring, which also applies to finding an MIS. Kuhn et al. [39] gave lower bounds of $\Omega(\sqrt{\log n / \log \log n})$ and $\Omega(\sqrt{\log \Delta / \log \log \Delta})$ for finding an MIS.

Barenboim and Elkin [4] provide a thorough tour on coloring algorithms (naturally, excluding recent results). An excellent survey on local problems is given by Suomela [59].

Distributed constructions of spanners The construction of spanners in the distribute setting has been studied extensively both in the randomized and deterministic setting [16–19,55]. We emphasize that the construction of [19] cannot be implemented in the congested clique by simply applying Lenzen's routing scheme because although each node sends $O(n \log n)$ bits of information, this information may need to be received by many nodes, and is not split among receivers. A randomized spanner construction was given by Baswana and Sen in [8]. They show that their well-known centralized algorithm can be implemented in the distributed setting even with small messages. In particular, they show that a $(2k-1)$ spanner with an expected number of $O(n^{1+1/k})$ edges can be constructed in $O(k^2)$ rounds in the CONGEST model (and for unweighted graphs, the algorithm takes $O(k)$ rounds, see [23]).

Derandomization of similar randomized algorithms has been addressed mainly in the *centralized* setting [56]. We emphasize that we need entirely different techniques to derandomize the Baswana–Sen algorithm compared with the centralized derandomization of [56].

The existing *deterministic* distributed algorithms for spanner are not based on derandomization of the randomized construction. They mostly use messages of *unbounded* size and are mainly based on sparse partitions or network decompositions. The state of the art is due to Derbel et al. [18]. They provide a *tight* algorithm for constructing $(2k-1)$-spanners with optimal stretch, size and construction time of k rounds. This was complemented by a matching lower bound, showing that any (even randomized) distributed algorithm requires k rounds in expectation. Much less efficient deterministic algorithms are known for the CONGEST model. [20] showed a construction of a $(2k-1)$-spanner in $O(n^{1-1/k})$ rounds. Deterministic construction with an improved tradeoff was recently obtained by [5], they showed a construction of $O(\log^{k-1} n)$-spanners with $O(n^{1+1/k})$ edges in $O(\log^{k-1} n)$ rounds.

Algorithms in the congested clique The congested clique model was first addressed in Lotker et al. [43], who raised the question of whether the global problem of constructing a minimum spanning tree (MST) can be solved faster on a communication graph with diameter 1. Since then, the model gained much attention, with results about its computational power given in [22], faster MST algorithms [28,31], distance computation [34,35,47], subgraph detection [21], algebraic computations [12,25], and routing and sorting [40,41,53]. Local problems were addressed in [33] who study ruling sets. Connections to the MapReduce model is given in [32].

Derandomization in the parallel/distributed setting Derandomization of local algorithms has attracted much attention in the *parallel* setting [1,10,13,29,30,36,38,46,50,58]. Luby [45] showed that his MIS algorithm (and more) can be derandomized in the PRAM model using $O(m)$ machines and $O(\log^3 n \log \log n)$ time. In fact, this much simpler algorithm can also be executed on the congested clique model, resulting in an $O(\log^4 n)$ running time. Similar variants of derandomization for MIS, maximal matching and $(\Delta+1)$-coloring were presented in [1,36]. Berger and Rompel [10] developed a general framework for removing randomness from RNC algorithms when polylogarithmic independence is sufficient. The parallel setting bears some similarity to the all-to-all communication model but the barriers in these two models are different mainly because the complexity measure in the parallel setting is the computation time while in our setting local computation is for free. This raises the possibility of obtaining much better results in the congested clique model compared to what is known in the parallel setting.

Turning to the distributed setting, Naor and Stockmeyer [48] showed that constant-round randomized algorithms for problems that are locally checkable can be derandomized without an asymptotic overhead, extended by [15,24] for larger time complexities and for a wider range of problems. Finally, Awerbuch et al. [2] claim to use the derandomized MIS algorithm of Luby [45] to obtain a deterministic CONGEST MIS algorithm. In this paper we provide a rigorous proof for this claim.

2 Preliminaries and notation

Our derandomization approach consists of the following ingredients: First we reduce the independence between the coin flips of the nodes. Then, we find some target function we wish to maintain during each iteration of the derandomized algorithm. Finally, we find a pessimistic estimator for the target function and apply the method of conditional expectations to get a deterministic algorithm. Below we elaborate upon the above ingredients.

d-wise independent random variables In the algorithms we derandomize in the paper, a node $v \in V$ flips coins with probability p of being heads. As we show, it is enough to assume only d-wise independence between the coin flips of nodes. We show how to use a randomness *seed* of only $t = d\lceil \max\{\log n, \log 1/p\}\rceil$ bits to generate a coin flip for each $v \in V$, such that the coin flips are d-wise independent.

We first need the notion of d-wise independent hash functions as presented in [60].

Definition 1 ([60, Definition 3.31]) For $N, M, d \in \mathbb{N}$ such that $d \leq N$, a family of functions $\mathcal{H} = \{h : [N] \to [M]\}$ is d-wise independent if for all distinct $x_1, x_2, \ldots, x_d \in [N]$, the random variables $H(x_1), \ldots, H(x_d)$ are independent and uniformly distributed in $[M]$ when H is chosen randomly from $\mathcal{H}$.

In [60] an explicit construction of $\mathcal{H}$ is presented, with parameters as stated in the next Lemma.

Lemma 1 ([60, Corollary 3.34]) *For every $\gamma, \beta, d \in \mathbb{N}$, there is a family of d-wise independent functions $\mathcal{H}_{\gamma,\beta} = \left\{h : \{0,1\}^\gamma \to \{0,1\}^\beta\right\}$ such that choosing a random function from $\mathcal{H}_{\gamma,\beta}$ takes $d \cdot \max\{\gamma, \beta\}$ random bits, and evaluating a function from $\mathcal{H}_{\gamma,\beta}$ takes time $poly(\gamma, \beta, d)$.*

Let us now consider some node $v \in V$ which needs to flip a coin with probability p that is d-wise independent with respect to the coin flips of other nodes. Using Lemma 1 with parameters $\gamma = \lceil \log n \rceil$ and $\beta = \lceil \log 1/p \rceil$, we can construct $\mathcal{H}$ such that every function $h \in \mathcal{H}$ maps the ID of a node to the result of its coin flip. Using only t random bits we can flip d-wise independent biased coins with probability p for all nodes in v.

We define Y to be a vector of t random coins. Note we can also view Y as a vector of length $t/\log n$ where each entry takes values in $[\log n]$. We use the latter when dealing with Y. From Y each node v can generate its random coin toss by accessing the corresponding $h \in \mathcal{H}$ and checking whether $h(ID(v)) = 0$. From Definition 1 it holds that $Pr[h(ID(v)) = 0] = 1/p$, as needed.

The method of conditional expectations Next, we consider the method of conditional expectations. Let $\phi : A^\ell \to \mathbb{R}$, and let $X = (X_1, \ldots, X_\ell)$ be a vector of random variables taking values in A. If $E[\phi(X)] \geq \alpha$ then there is an assignment of values $Z = (z_1, \ldots, z_\ell)$ such that $\phi(Z) \geq \alpha$. We describe how to *find* the vector Z. We first note that from the law of total expectation it holds that $E[\phi(X)] = \sum_{a \in A} E[\phi(X) \mid X_1 = a] Pr[X_1 = a]$, and therefore for at least some $a \in A$ it holds that $E[\phi(X) \mid X_1 = a] \geq \alpha$. We set this value to be z_1. We then repeat this process for the rest of the values in X, which results in the vector Z. In order for this method to work we need it to be possible to *compute* the conditional expectation of $\phi(X)$. We now wish to use the method of conditional expectations after reducing the number of random bits used by the algorithm. Let us denote by $\bar{\rho}$ the original vector of random bits used by the algorithm. Taking Y as before to be the seed vector for $\bar{\rho}$, we have that $\bar{\rho}$ is a function of Y. We need to be able to compute $E[\phi(\bar{\rho}(Y)) \mid y[1] = a_1, \ldots, y[i] = a_i]$ for all possible values of i and a_j, $j \leq i$.

Computing the conditional expectations for ϕ might be expensive. For this reason we use a *pessimistic estimator*. A pessimistic estimator of ϕ is a function $\phi' : A^\ell \to \mathbb{R}$ such that that for all values of i and a_j, $j \leq i$ it holds that $E[\phi(\bar{\rho}(Y)) \mid y_1 = b_1, \ldots, y_i = b_i] \geq E[\phi'(\bar{\rho}(Y)) \mid y_1 = b_1, \ldots, y_i = b_i]$. If ϕ' is a pessimistic estimator of ϕ whose expected value is still bounded by α, then we can use the method of conditional expectations on ϕ' and obtain $z_1, \ldots, z_n$, such that $\phi(z_1, \ldots, z_n) \geq \phi'(z_1, \ldots, z_n) \geq \alpha$.

Lenzen's routing algorithm We make heavy use of the deterministic routing algorithm of Lenzen [40], which guarantees that if each node needs to send at most $O(n \log n)$ bits and receive at most $O(n \log n)$ bits then $O(1)$ rounds are sufficient.

3 Randomized MIS as a starting point

To prove Theorem 1, we consider the following modification of the randomized algorithm of Ghaffari [26]. The algorithm of Ghaffari consists of two parts. The first part (shown to have a good local complexity) consists of $O(\log \Delta)$ phases, each with $O(1)$ rounds. After this first part, it is shown that sufficiently many nodes are removed from the graph. The MIS for what remains is computed in the second part deterministically in time $2^{O(\sqrt{\log \log n})}$. We only use the first part of Ghaffari's algorithm, and the only change to it is a slight modification of the constants that are used.

Slight modification to the first part of Ghaffari's MIS Algorithm

Set $p_0(v) = 1/4$.

$$p_{t+1}(v) = \begin{cases} 1/2 \cdot p_t(v), & \text{if } d_t(v) \geq 1/2 \\ \min\{2p_t(v), 1/4\}, & \text{if } d_t(v) < 1/2, \end{cases}$$

where $d_t(v) = \sum_{u \in N(v)} p_t(u)$ is the *effective degree* of node v in phase t. In each phase t, the node v gets marked with probability $p_t(v)$ and if none of its neighbors is marked, v joins the MIS and gets removed along with its neighbors.

3.1 An $O(\log \Delta)$ round randomized MIS algorithm in the congested clique

We begin by observing that in the congested clique, what remains after $O(\log \Delta)$ phases of Ghaffari's algorithm can be solved in $O(1)$ rounds. This provides an improved *randomized* runtime compared to [26], and specifically, has no dependence on n.

The algorithm consists of two parts. In the first part, we run Ghaffari's algorithm for $O(\log \Delta)$ phases. We emphasize that this works with both Ghaffari's algorithm and with our modified Ghaffari's algorithm, since the values of the constants do not affect the asymptotic running time and correctness of the randomized first part of the algorithm.

Then, in the second part, a leader collects all surviving edges and solves the remaining MIS deterministically on that subgraph. We show that the total number of edges incident to these nodes is $O(n)$ w.h.p., and hence using the deterministic routing algorithm of Lenzen [40], the second part can be completed in $O(1)$ rounds w.h.p.

Lemma 2 ([26, Theorem 3.1]) *For every $\epsilon \in (0,1)$, there exists a constant c', such that for each node v, the probability that v has not made its decision within the first $c' \cdot (\log \Delta + \log 1/\epsilon)$ phases is at most ϵ.*

Since the decision whether to join the MIS or to be removed by having a neighbor in the MIS depends only on the 2-neighborhood of a node, decisions made by nodes that are at least 4 hops from each other are independent. We make use of the following variant of Chernoff's bound.

Fact 4 (Bounded dependency Chernoff bound [54]) *Let $X_1, \ldots, X_n$ denote a set of binary random variables with bounded dependency $\widehat{d}$, and let $\mu = \mathbb{E}(\sum_i X_i)$. Then:*

$$\Pr\left[\sum_i X_i \geq (1+\delta)\mu\right] \leq O(\widehat{d}) \cdot e^{-\Theta(\delta^2 \mu/\widehat{d})}.$$

Corollary 2 *After $O(\log \Delta)$ phases, the number of edges incident to nodes that have not made their decisions is $O(n)$.*

Proof By Lemma 2, there is a constant c, such that the probability that a node has not made its decision after $O(\log \Delta)$ phases is at most $1/\Delta^c$. Letting X denote the random variable of the number of nodes that have not made their decision after $O(\log \Delta)$ phases gives that $\mu = \mathbb{E}[X] = n/\Delta^c$.

Conditioned on the outcomes of the previous iterations, each node is mutually independent from all nodes except in its 4-neighborhood, in which there are up to Δ^4 nodes. By using Fact 4 with $\delta = \Theta(\Delta^{c-1})$ and $\widehat{d} = \Delta^4$, and taking c to be large enough, we get that

$$\Pr[X \geq n/\Delta] \leq O(\Delta^4) \cdot e^{-\Theta(n)}.$$

Hence, w.h.p., there are $O(n/\Delta)$ remaining nodes and therefore the number of their incident edges is at most $O(n)$. □

We conclude:

Theorem 1 *There is a randomized algorithm that computes MIS in the congested clique model within $O(\log \Delta)$ rounds with high probability.*

Note that the proof of Corollary 2 cannot be extended to the case of pairwise independence, which is needed for derandomization, since the concentration guarantees are rather weak. For this, we need to develop some new machinery, as we describe in the following section.

3.2 Derandomizing the modified MIS algorithm

We first turn to consider the modified Ghaffari's algorithm when the random decisions made by the nodes are only *pairwise* independent.

3.2.1 Ghaffari's algorithm with pairwise independence

We review the main terminology and notation from [26], up to our modification of constants. Recall that $d_t(v) = \sum_{u \in N(v)} p_t(u)$. A node v is called *light* if $d_t(v) < 1/4$.

Golden phases and golden nodes We define two types of *golden phases* for a node v. This definition is a modification of the corresponding definitions in [26].

Definition 2 Type-1 golden phase: $p_t(v) = 1/4$ and $d_t(v) \leq 1/2$;
Type-2 golden phase: $d_t(v) > 1/4$ and at least $d_t(v)/10$ of it arises from light nodes.

A node v is called *golden* in phase t, if phase t is a golden phase for v (of either type). Intuitively, a node v that is golden in phase t is shown to have a constant probability of being removed. Specifically, in a golden phase of type-1, v has a constant probability to join the MIS and in a golden phase of type-2, there is a constant probability that v has a neighbor that joins the MIS and hence v is removed.

We now prove the analogue of Lemma 3.3 in [26] for the setting in which the coin flips made by the nodes are not completely independent but are only *pairwise independent*. We show that a golden node for phase t is still removed with constant probability even under this weaker bounded independence guarantee.

Lemma 3 (Type-1 golden nodes with pairwise independence) *Consider the modified Ghaffari's algorithm with pairwise independent coin flips. If t is a type-1 golden phase for a node v, then v joins the MIS in phase t with probability at least $1/8$.*

Proof In each type-1 golden phase, v gets marked with probability $p_t(v) = 1/4$. By the inclusion-exclusion principle,

the probability that v is marked but none of its neighbors is marked in phase t can be bounded by:

$$\begin{aligned}&\Pr[v \text{ is the only marked node in } N(v)]\\ &\geq p_t(v) - \sum_{u\in N(v)} p_t(v)\cdot p_t(u)\\ &\geq p_t(v)(1-d_t(v)) \geq 1/4(1-1/2) = 1/8.\end{aligned}$$

Hence, v joins the MIS with probability at least $1/8$. □

We next show that also the golden nodes of type-2 are removed with constant probability assuming only pairwise independence.

Lemma 4 (Type-2 golden nodes with pairwise independence) *Consider the modified Ghaffari's algorithm with pairwise independent coin flips. If t is a type-2 golden phase for a node v then v is removed in phase t with probability at least $\alpha = 1/160$.*

Proof For node v, fix a subset of light neighbors $W(v) \subseteq N(v)$ satisfying that $\sum_{u\in W(v)} p_t(u) \in [1/40, 1/4]$. Such a set exists because by the definition of a type-2 golden phase, the sum of probabilities of light neighbors of v is at least $1/40$, and every single probability is at most $1/4$ by the constants taken in the algorithm (this probability either halves in each phase, or is bounded from above by $1/4$).

For $u \in W(v)$, let $\Upsilon_t(v,u)$ denote the indicator variable of the event that in phase t the following happens: u gets marked, *none* of u's neighbors get marked and u is the *only* neighbor of v in $W(v)$ that got marked. For node u, v and phase t, let $m_{u,v,t}$ be the indicator random variable that both u and v get marked in phase t. Due to the pairwise independence, we have: $\Pr[m_{u,v,t} = 1] = p_t(u)\cdot p_t(v)$. Hence, the probability of the event indicated by $\Upsilon_t(v,u)$ can be bounded by:

$$\begin{aligned}&\Pr[\Upsilon_t(v,u) = 1]\\ &\geq p_t(u) - \sum_{w\in N(u)} \Pr[m_{u,w,t} = 1]\\ &\quad - \sum_{u'\in W(v)\setminus\{u\}} \Pr[m_{u,u',t} = 1]\\ &= p_t(u) - \sum_{w\in N(u)} p_t(u)p_t(w) - \sum_{u'\in W(v)\setminus\{u\}} p_t(u)p_t(u')\\ &\geq p_t(u)\left(1 - \sum_{w\in N(u)} p_t(w) - \sum_{u'\in W(v)\setminus\{u\}} p_t(u')\right)\\ &\geq p_t(u)\,(1 - d_t(u) - 1/4) \geq p_t(u)\,(1-1/2-1/4)\\ &= 1/4\cdot p_t(u).\end{aligned}$$

Since the events indicated by $\Upsilon_t(v,u)$ are mutually exclusive for different $u, u' \in N(v)$, it holds that the probability that v gets removed is at least

$$\begin{aligned}\Pr[v \text{ is removed in phase } t] &\geq \sum_{u\in W(v)} \Pr[\Upsilon_t(v,u) = 1]\\ &\geq 1/4\cdot \sum_{u\in W(v)} p_t(u) \geq \frac{1}{160}\end{aligned}$$

□

Finally, we claim the analogue of Lemma 3.2 in [26].

Lemma 5 *Consider the modified Ghaffari's algorithm with pairwise independent randomness and $\epsilon = 1/\Delta^c$. For a large enough c', for every v, at the end of $c'\cdot\log\Delta$ phases, either v has joined the MIS, or it has a neighbor in the MIS, or at least one of its golden phase counts reached $c\cdot\log\Delta$.*

The proof here is exactly that same as in [26]. The reason is that the proof does not assume independence and is only affected by the update rule of the probabilities. Note that similarly to [26], we had to define type-2 with a threshold on $d_t(v)$ which is factor 2 smaller than that of type-1. As a result, the following holds in the pairwise independence setting:

Lemma 6 *Within $O(\log\Delta)$ phases, every node remains with probability at most $1/\Delta$.*

We emphasize that the proof of Corollary 2 cannot be extended to pairwise independence since the concentration guarantees are rather weak. Our algorithm will use pairwise independence but with some crucial modifications required in order to guarantee that after $O(\log\Delta)$ phases, only $O(n/\Delta)$ nodes remain undecided.

4 Deterministic MIS

4.1 Deterministic $O(\log n \log\Delta)$-round algorithm in the congested clique

We now turn to consider the derandomization procedure. We show the following:

Theorem 2 *There is a deterministic MIS algorithm for the congested clique model that completes in $O(\log\Delta\log n)$ rounds.*

The challenge Consider phase t in the modified Ghaffari's algorithm and let V_t be the set of golden nodes in this phase. Our goal is to select additional nodes into the MIS so that at least a constant fraction of the golden nodes are removed. Let $v_1,\ldots,v_{n'}$ be the nodes that are *not* removed in phase t. For each node, define corresponding random variables $x_1,\ldots,x_{n'}$ indicating whether v_i is marked in phase t or not. Let $X_i = (x_1 = b_1,\ldots,x_i = b_i)$ define a partial assignment for the nodes $v_1,\ldots,v_i$ (i.e., whether or not they are marked

in phase t). Let $X_0 = \emptyset$ denote the case where none of the decisions is fixed.

For a golden node v, let $r_{v,t}$ be the random variable indicating whether v gets removed in phase t, and let R_t be the random variable of the number of removed golden nodes. By linearity of expectation, $\mathbb{E}(R_t) = \sum_v \mathbb{E}(r_{v,t})$ is the *expected* number of removed golden nodes in phase t. By Lemmas 3 and 4, there is a constant c such that $\mathbb{E}(R_t) \geq c \cdot |V_t|$. We then use the similar approach of Luby [45] and in particular define a local way to estimate some bound on the individual conditional expectation of the local progress for each node. A new challange that does not appear in Luby's algorithm concerns the shattering effect. Our goal is to show that after $O(\log \Delta)$ phases of the derandomizion the remaining unsolved graph has $O(n)$ edges. While this property holds in the full independence setting, it does not necessary hold with pairwise independence. That is, the proof of Corollary 2 inherently needs full independence. To address this challange, we use of a priority-based scheme for choosing the nodes that join the MIS, which requires a novel age-based weighting approach to be added to the MIS algorithm. Next, we describe our main derandomization tool and then provide our algorithm.

Derandomization tools The first two tools described are followed by Luby [45] adapted to Ghaffari's algorithm, and the third one is new in this context. We define a pessimistic estimator to the conditional expectation $\mathbb{E}(R_t \mid X_i)$, which can be computed efficiently in our model. Then, we describe how to reduce the search space using pairwise independence. In our algorithm, the nodes will apply the method of conditional expectations on the estimator in order to find a "good" seed of length $O(\log n)$.

Tool 1: The pessimistic estimator function Consider phase t and recall that V_t are the golden nodes in this phase. Similarly to the clever approach of [45], we define a variable $\psi_{v,t}$ that will satisfy that $r_{v,t} \geq \psi_{v,t}$. The idea is to account for a removed node of type-2 only in the case that it is removed because a *single* one of its neighbors joins the MIS. Since this can only occur for one of its neighbors, we avoid double-counting when computing the probabilities. Let $m_{v,t}$ be the random variable indicating the event that v is *marked* in round t. Let $m_{v,u,t}$ indicate the event that both u and v are marked in round t. Define

$$\psi_{v,t} = \begin{cases} m_{v,t} - \sum_{u \in N(v)} m_{v,u,t}, & \text{if } v \text{ is of type-1.} \\ \sum_{u \in N(v)} (m_{u,t} - \sum_{w \in N(u)} m_{u,w,t} \\ -\sum_{w' \in W(v) \setminus \{u\}} m_{u,w',t}), & \text{if } v \text{ is of type-2.} \end{cases}$$

Denoting $\Psi_t = \sum_{v \in V_t} \psi_{v,t}$ gives that Ψ_t is a lower bound on the number of removed golden nodes, i.e., $\Psi_t \leq R_t$. For a partial assignment $X_i = (x_1 = b_1, \ldots, x_i = b_i)$ indicating which of the nodes are marked, we have

$$\mathbb{E}(\psi_{v,t} \mid X_i) = \begin{cases} \Pr[m_{v,t} = 1 \mid X_i] - \\ \sum_{u \in N(v)} \Pr[m_{v,u,t} \mid X_i], \\ \text{if } v \text{ is of type-1.} \\ \sum_{u \in N(v)} (\Pr[m_{u,t} = 1 \mid X_i] - \\ \sum_{w \in N(u)} \Pr[m_{u,w,t} = 1 \mid X_i] - \\ \sum_{w' \in W(v) \setminus \{u\}} \Pr[m_{u,w',t} = 1 \mid X_i)], \\ \text{if } v \text{ is of type-2,} \end{cases} \quad (1)$$

where $W(v) \subseteq N(v)$ is a subset of v's neighbors satisfying that $\sum_{w \in W(v)} p_t(w) \in [1/40, 1/4]$ (as used in the proof of Lemma 4). By Lemmas 3 and 4, it holds that $\mathbb{E}(\psi_{v,t}) \geq \alpha$ for $v \in V_t$. Hence, we have that:

$$\mathbb{E}(r_{v,t}) \geq \mathbb{E}(\psi_{v,t}) \geq \alpha.$$

Since $r_{v,t} \geq \psi_{v,t}$ even upon conditioning on the partial assignment X_i, we get:

$$\mathbb{E}(R_{v,t} \mid X_i) \geq \mathbb{E}(\Psi_t \mid X_i) = \sum_{v \in V_t} \mathbb{E}(\psi_{v,t} \mid X_i) \geq \alpha \cdot |V_t|.$$

Our algorithm will employ the method of conditional expectations on a *weighted* version of $\mathbb{E}(\Psi_t \mid X_i)$, as will be discussed later.

Tool 2: Pairwise independence We now combine the method of conditional expectations with a small search space. We use Lemma 1 with $d = 2, \gamma = \Theta(\log n)$ and a prime number $\beta = O(\log \Delta)$. This is because we need the marking probability, $p_t(v)$, to be $\Omega(1/\text{poly } \Delta)$.

Consider phase t. Using the explicit construction of Lemma 1, if all nodes are given a shared random seed of length γ, they can sample a random hash function $h : \{0,1\}^\gamma \to \{0,1\}^\beta$ from $\mathcal{H}_{\gamma,\beta}$ which yields n pairwise independent choices. Specifically, flipping a biased coin with probability of $p_t(v)$ can be trivially simulated using the hash value $h(ID_v)$ where ID_v is an $O(\log n)$-bit ID of v.[2] Since h is a random function in the family, all random choices are pairwise independent and the analysis of of the golden phases goes through.

Even though using a seed of length $O(\log n)$ reduces the search space to be of polynomial size, still, exploring all possible $2^{O(\log n)} = O(n^c)$ seeds in a brute force manner is too time consuming. Instead, we employ the method of conditional expectations to find a *good seed*. That is, we will consider $\mathbb{E}(\Psi_t \mid Y_i)$ where $Y_i = (y_1 = b_1, \ldots, y_i =$

[2] Flipping a biased coin with probability $1/2^i$, is the same as getting a uniformly distributed number y in $[1, b]$ and outputting 1 if and only if $y \in [1, 2^{b-i}]$.

b_i) is a partial assignment to the *seed* $Y = (y_1, \ldots, y_a)$. The crux here is that since a *random* seed is good, then so is the expectation over seeds that are sampled uniformly at random. Hence, the method of conditional expectations will find a seed that is at least as good as the random selection. Specifically, we still use the pessimistic estimator of Eq. (1), but we condition on the small seed Y_i rather than on X_i.

Tool 3: An age-based weighted adaptation To handle the shattering effect, we compute the expectation of a weighted version of Ψ_t, which favors *old* nodes where the age of a node is counted as the number of golden phases it experienced. Let $age_t(v)$ be the number of golden phases v has till phase t and recall that a golden node is removed with probability at least α. Define $\psi'_{v,t} = (1+\alpha)^{age_t(v)} \cdot \psi_{v,t}$, and $\Psi'_t = \sum_{v \in V_t} \psi'_{v,t}$. We use the method of conditional expectations for:

$$\mathbb{E}(\Psi'_t \mid Y_i) = \sum_{v \in V_t} \mathbb{E}(\psi'_{v,t} \mid Y_i), \tag{2}$$

rather than for $\mathbb{E}(\Psi_t \mid Y_i)$. The choice of this function will be made clear in the proof of Lemma 7.

Algorithm description The first part of the algorithm consists of $\Theta(\log \Delta)$ phases, where in phase t, we derandomize phase t in the modified Ghaffari's algorithm using $O(\log n)$ deterministic rounds. In the second part, all nodes that remain undecided after the first part, send their edges to the leader using the deterministic routing algorithm of Lenzen. The leader then solves locally and notifies the relevant nodes to join the MIS. In the analysis section, we show that after the first part, only $O(n/\Delta)$ nodes remain undecided, and hence the second part can be implemented in $O(1)$ rounds.

From now on we focus on the first part. Consider phase t in the modified Ghaffari's algorithm. Note that at phase t, some of the nodes are already removed from the graph (either because they are part of the MIS or because they have a neighbor in the MIS). Hence, when we refer to nodes or neighboring nodes, we refer to the remaining graph induced on the undecided nodes.

Let $Y = (y_1, \ldots, y_\gamma)$ be the γ random variables that are used to select a hash function and hence induce a deterministic algorithm. We now describe how to compute the value of y_i in the seed, given that we already computed $y_1 = b_1, \ldots, y_{i-1} = b_{i-1}$. By exchanging IDs (of $\Theta(\log n)$ bits), as well as the values $p_t(v)$ and $d_t(v)$ with its neighbors, a node can check if it is a golden type-1 or type-2 node according to the conditions of Definition 2. In addition, every node maintains a counter, $age(v)$ referred to as the age of v, which measures the number of golden phases it had so far.

Depending on whether the node v is a golden type-1 or type-2 node, based on Eq. (1), it computes a lower bound on the conditional probability that it is removed given the partial seed assignment $Y_{i,b} = (y_1, \ldots, y_i = b)$ for every $b \in \{0, 1\}$. These lower bound values are computed according to the proofs of Lemmas 3 and 4.

Specifically, a golden node v of type-1, uses the IDs of its neighbors and their $p_t(u)$ values to compute the following:

$$\mathbb{E}(\psi_{v,t} \mid Y_{i,b}) = \Pr[m_{v,t} = 1 \mid Y_{i,b}] - \sum_{u \in N(v)} \Pr[m_{u,t} = 1 \mid Y_{i,b}],$$

where $\Pr[m_{v,t} = 1 \mid Y_{i,b}]$ is the conditional probability that v is marked in phase t (see Sect. A for full details about this computation).

For a golden node v of type-2 the lower bound is computed differently. First, v defines a subset of neighbors $W(v) \subseteq N(v)$, satisfying that $\sum_{w \in W(v)} p_t(w) \in [1/40, 1/4]$, as in the proof of Lemma 4. Let $M_{t,b}(u)$ be the conditional probability on $Y_{i,b}$ that u is marked but none of its neighbors are marked. Let $M_{t,b}(u, W(v))$ be the conditional probability on $Y_{i,b}$ that another node other than u is marked in $W(v)$.[3] By exchanging the values $M_{t,b}(u)$, v computes:

$$\mathbb{E}(\psi_{v,t} \mid Y_{i,b}) = \sum_{u \in W(v)} \Pr[m_{u,t} = 1 \mid Y_{i,b}] - M_{t,b}(u) - M_{t,b}(u, W(v)).$$

Finally, as in Eq. (2), the node sends to the leader the values $\mathbb{E}(\psi'_{v,t} \mid Y_{i,b}) = 1/(1-\alpha)^{age(v)} \cdot \mathbb{E}(\psi_{v,t} \mid Y_{i,b})$ for $b \in \{0, 1\}$. The leader computes the sum of the $\mathbb{E}(\psi'_{v,t} \mid Y_{i,b})$ values of all golden nodes V_t, and declares that $y_i = 0$ if $\sum_{v \in V_t} \mathbb{E}(\psi'_{v,t} \mid Y_{i,b}) \geq \sum_{v \in V_t} \mathbb{E}(\psi'_{v,t} \mid Y_{i,b})$, and $y_i = 1$ otherwise. This completes the description of computing the seed Y.

Once the nodes compute Y, they can simulate phase t of the modified Ghaffari's algorithm. In particular, the seed Y defines a hash function $h \in \mathcal{H}_{\gamma,\beta}$ and $h(ID(v))$ can be used to simulate the random choice with probability $p_t(v)$. The nodes that got marked send a notification to neighbors and if none of their neighbors got marked as well, they join the MIS and notify their neighbors. Nodes that receive join notification from their neighbors are removed from the graph. This completes the description of the first part of the algorithm. For completeness, a pseudocode appears in Appendix A.

Analysis The correctness proof of the algorithm uses a different argument than that of Ghaffari [26]. Our proof does not involve claiming that a constant fraction of the golden nodes are removed, because in order to be left with $O(n/\Delta)$ undecided nodes we have to favor removal of old nodes. The entire correctness is based upon the following lemma, which justifies the definition of the expectation given in Eq. (2).

[3] The term $M_{t,b}(u, W(v))$ is important as it is what prevents double counting, because the corresponding random variables defined by the neighbors of v are mutually exclusive.

Lemma 7 *The number of undecided nodes after $\Theta(\log \Delta)$ phases is $O(n/\Delta)$ and hence the total number of edges incident to these nodes is $O(n)$.*

Proof For every phase t, denote by V'_t as the set of undecided nodes at the beginning of phase t, and let $V_t \subseteq V'_t$ be the set of golden nodes in that phase. We also define a potential function $\Phi_t = \sum_{v \in V'_t} (1+\alpha)^{age_t(v)}$ where $age_t(v)$ is the number of golden phases $v \in V_t$ had till phase t (not including t). Hence, intuitively, a node is *old* if it has experienced many golden phases.

We first show that the potential function Φ_t is non-increasing with t. Fix a phase t, and recall that $r_{t,v}$ is the random variable indicating the event that a golden node $v \in V_t$ is removed at phase t, and $\psi_{v,t}$ is used to obtain a pessimistic estimator for v being removed. The nodes compute the conditional expectation for:

$$\mathbb{E}(\Psi'_t) = \sum_{v \in V_t} \Pr[\psi_{v,t} \geq 1] \cdot (1+\alpha)^{age_t(v)}.$$

By Lemmas 3 and 4, $\Pr[\psi_{v,t} \geq 1] \geq \alpha$, hence:

$$\mathbb{E}(\Psi'_t) \geq \alpha \sum_{v \in V_t} (1+\alpha)^{age_t(v)}. \tag{3}$$

Let $A = V'_t \backslash V_t$ be the non-golden vertices at the beginning of phase t. Let $B \subseteq V_t$ be the set of removed golden nodes in this phase, and let $C = V_t \backslash B$ be the remaining undecided golden nodes. We will now bound Φ_{t+1} by considering the contribution of each subset A, B and C separately.

Let $\Phi_t(A) = \sum_{v \in A} (1+\alpha)^{age_t(v)}$, and define $\Phi_t(B)$ and $\Phi_t(C)$ analogously. We then have that $\Phi_t = \Phi_t(A) + \Phi_t(B) + \Phi_t(C)$. Since the nodes in A did not age in this phase, we have that $\Phi_{t+1}(A) = \Phi_t(A)$. In addition by Eq. (3), we have that $\Phi_t(B) \geq \alpha(\Phi_t(B) + \Phi_t(C))$, and therefore

$$\Phi_t(C) \leq (1-\alpha)(\Phi_t(B) + \Phi_t(C)).$$

Finally, $\Phi_{t+1}(C) = (1+\alpha)\Phi_t(C) \leq (1+\alpha)(1-\alpha)(\Phi_t(B) + \Phi_t(C)) \leq \Phi_t(B) + \Phi_t(C)$. Overall, we have:

$$\Phi_{t+1} = \Phi_{t+1}(A) + \Phi_{t+1}(C) \leq \Phi_t(A) + \Phi_t(B) + \Phi_t(C) = \Phi_t,$$

as desired.

We are now ready to complete the proof. By Lemma 5, for a sufficiently large constant β, a node that remains undecided after $\tau = \beta \log \Delta$ phases is of age at least $\log \Delta / \log(1+\alpha)$. Since $\Phi_0 = n$, and by the above argument we have that

$$n \geq \Phi_n \geq \Phi_\tau \geq \sum_{v \in V'_\tau} (1+\alpha)^{\log \Delta / \log(1+\alpha)} = |V_\tau| \cdot \Delta.$$

Thus, we get that the number of remaining undecided vertices after $\tau = O(\log \Delta)$ phases is $|V'_\tau| = O(n/\Delta)$, concluding that the size of unsolved subgraph is $O(n)$ as desired. □

The remaining $O(n)$ edges incident to the undecided nodes can be collected at the leader in $O(1)$ rounds using the deterministic routing algorithm of Lenzen [40]. The leader then solves MIS for the remaining graph locally and informs the nodes. This completes the correctness of the algorithm. Theorem 2 follows.

4.2 An $O(\log \Delta)$ deterministic MIS algorithm for $\Delta = O(n^{1/3})$

In the case where the maximal degree is bounded by $\Delta = O(n^{1/3})$, our deterministic bounds match the randomized ones.

Theorem 3 *If $\Delta = O(n^{1/3})$ then there is a deterministic MIS algorithm for the congested clique model that completes in $O(\log \Delta)$ rounds.*

Proof The algorithm consists of two parts as before, namely, $O(\log \Delta)$ phases that simulate the modified Ghaffari's algorithm and collecting the remaining topology at a leader and solving MIS for it locally. The second part works exactly the same as before, and so we focus on the first part which simulates the $O(\log \Delta)$ phases of the modified Ghaffari's algorithm in $O(\log \Delta)$ *deterministic* rounds. The main advantage of having a small degree $\Delta = O(n^{1/3})$ is that in the congested clique, it is possible for each node to collect the entire topology of its 2-neighborhood in $O(1)$ rounds. This because the 2-neighborhood of a node contains $O(\Delta^2)$ nodes and $O(\Delta^3)$ edges, and hence there are $O(\Delta^3) = O(n)$ messages a node needs to send or receive, which can be done in $O(1)$ rounds using Lenzen's routing algorithm [40].

We now consider phase t of the modified Ghaffari's algorithm and explain how the seed of length $O(\log n)$ can be computed in $O(1)$ rounds. Unlike the algorithm of the previous section, which computes the seed bit by bit, here the nodes compute the assignment for a *chunk* of $z = \lfloor \log n \rfloor$ variables at a time.

To do so, consider the i'th chunk of the seed $Y'_i = (y'_1, \ldots, y'_z)$. For each of the n possible assignments $(b'_1, \ldots, b'_z) \in \{0,1\}^z$ to the z variables in Y', we assign a node u that receives the conditional expectation values from all the golden nodes, where the conditional expectation is computed based on assigning $y'_1 = b'_1, \ldots, y'_z = b'_z$. The node u then sums up all these values and obtains the expected number of removed nodes conditioned on the assignment $y'_1 = b'_1, \ldots, y'_z = b'_z$. Finally, all nodes send to the leader their computed sum and the leader selects the assignment $(b^*_1, \ldots, b^*_z) \in \{0,1\}^z$ of largest value. □

As mentioned in the introduction, combining Theorems 2 and 3 directly gives that the complexity is either $O(\log \Delta)$

rounds in case $\Delta = O(n^{1/3})$, and otherwise it is $O(\log^2 \Delta)$ since $\log n$ is then asymptotically equal to $\log \Delta$.

Corollary 1 There is a deterministic MIS algorithm for the congested clique model that completes in $O(\log^2 \Delta)$ rounds.

4.3 An $O(D \log^2 n)$ deterministic MIS algorithm for CONGEST

Here we provide a fast deterministic MIS algorithm for the harsher CONGEST model. For comparison, in terms of n alone, the best deterministic MIS algorithm is by Panconesi and Srinivasan [49] from more than 20 years ago and is bounded by $2^{O(\sqrt{\log n})}$ rounds. However, the algorithm requires large messages and hence is suitable for the LOCAL model but not for CONGEST. The only known non-trivial deterministic solution for CONGEST is to use any $(\Delta+1)$-coloring algorithm running in $O(\Delta + \log^* n)$ rounds (for example [3,6]) to obtain the same complexity for deterministic MIS in CONGEST (notice that there are faster coloring algorithms, but the reduction has to pay for the number of colors anyhow).

The following is our main result for CONGEST.

Theorem 1.5 *There is a deterministic MIS algorithm for the CONGEST model that completes in $O(D \log^2 n)$ rounds.*

Proof The algorithm is very similar to that of Theorem 2 with two main differences. First, we run Ghaffari's algorithm for $O(\log n)$ rounds instead of $O(\log \Delta)$ rounds. Each round is simulated by a phase with $O(D \log n)$ rounds. Specifically, in each phase, we need to compute the seed of length $O(\log n)$, this is done bit by bit using the method of conditional expectations exactly as described earlier and aggregating the result at some leader node (aggregation is done in the standard way). The leader then notifies the assignment of the bit to the entire graph. Since each bit in the seed is computed in $O(D)$ rounds, overall the run time is $O(D \log^2 n)$.

The correctness follows by applying the proof of Lemma 7 with $\Delta = n$. □

5 Deterministic spanner construction

In this section we present a derandomization algorithm in the congested clique for the spanner construction of Baswana–Sen [8]. We use the same general outline as in the MIS derandomization: We first reduce the dependence between the coins used by the algorithm and then use the method of conditional expectations for every iteration of the algorithm. However, here we face different challenges that we need to overcome.

The following is the main theorem of this section.

Theorem 1.6 *There is a deterministic algorithm for the congested clique model that completes in $O(k \log n)$ rounds and produces a $(2k-1)$-spanner with $O(kn^{1+1/k} \log n)$ edges.*

We first present the original algorithm of Baswana–Sen [8], which constructs a $(2k-1)$-spanner with $O(kn^{1+1/k})$ edges in $O(k^2)$ rounds. Next, we consider the same algorithm with only limited independence between its coin tosses. We prove some properties of the algorithm and show it can be derandomized. Finally we present our deterministic algorithm for the congested clique which constructs a $(2k-1)$-spanner with $O(kn^{1+1/k} \log n)$ edges within $O(k \log n)$ rounds.

The randomized spanner algorithm We begin by presenting a simplified version of the Baswana–Sen algorithm. For the full details of the Baswana–Sen algorithm we refer the reader to [8].[4]

At each iteration of this phase, the algorithm maintains a *clustering* of the vertices. A *cluster* is a subset of vertices, and a clustering is a set of disjoint clusters. In the distributed setting each cluster has a leader, and a spanning tree rooted at the leader is maintained inside the cluster. We will abuse notation and say that a cluster performs a certain action. When we say this, it means that the leader gathers the required information from the cluster vertices to make a decision, and propagates relevant data down the cluster tree. We will also refer at times to the ID of a cluster, which is the ID of the cluster leader.

We denote the clustering maintained at iteration i by $\mathcal{C}_i$, where initially $\mathcal{C}_0 = \{\{v\} \mid v \in V\}$. At each iteration, $\mathcal{C}_i$ is sampled from $\mathcal{C}_{i-1}$, by having every cluster in $\mathcal{C}_{i-1}$ join $\mathcal{C}_i$ with probability $n^{-1/k}$. In the final iteration we force $\mathcal{C}_k = \emptyset$. A vertex v that belongs to a cluster $C \in \mathcal{C}_i$ is called *i-clustered*, and otherwise it is *i-unclustered*.

The algorithm also maintains a set of edges, E', initialized to E. For every edge $e = (u, v)$ removed from E' during the algorithm, it is guaranteed that there is a path from u to v in the constructed spanner, H, consisting of at most $(2k-1)$ edges, each of weight not greater than the weight of e.

Let $v \in V$ be a vertex that stopped being clustered at iteration i, and let $E'(v, C) = \{(v, u) \in E' \mid u \in C\}$ be the set of edges between a vertex and a cluster C, for every $C \in \mathcal{C}_i$. Let $e_{v,C}$ be the lightest edge in $E'(v, C)$.

Let L be the set of lightest edges $e_{v,C}$ between v and the clusters C in $\mathcal{C}_{i-1}$. We go over L in ascending order of edge weight, adding an edge connecting v to cluster C and then discarding $E'(v, C)$ from E'. We say that v adds these edges at iteration i. If we reach a cluster $C \in \mathcal{C}_i$, we continue to the next vertex. Since in the last iteration, we

[4] The simplified version that we describe consists of only one phase that includes both phases mentioned in [8]. This is because the number of clusters in the last iteration k is forced to be zero and hence all vertices are unclustered in the last iteration and so they connect to all clusters of the previous iteration.

force the number of clusters to be zero, all vertices become unclustered at *some* iteration $i \in \{1, \ldots, k\}$ and hence their incident edges are taken care at that point. The pseudocode appears in Algorithms 1 and 2.

Algorithm 1: Randomized $(2k-1)$-spanner construction

```
H = ∅
C_0 = {{v} | v ∈ V}
E' = E
for i from 1 to k do
    if i = k then
        C_k = ∅  // This means that in the last iteration no node is covered, i.e., we add edges for all nodes to the clusters of C_{k-1}.
    else
        C_i is sampled from C_{i-1} by sampling each cluster with probability n^{-1/k}
    Run Algorithm 2
```

Algorithm 2: Baswana–Sen iteration

```
foreach vertex v that stopped being clustered at iteration i simultaneously do
    L = {e_{v,C} | C ∈ C_{i-1}}
    Let e_j be the j-th edge in L in ascending weight
    for j = 1 to |L| do
        H = H ∪ {e_j}
        Let C be the cluster of the other endpoint of e_j.
        E' = E' \ E'(v, C)
        if C ∈ C_i then
            break  // This means that we added also one edge E(v, C*) to the cluster C* ∈ C_i with the minimum weight edge.
```

Algorithm 1 is guaranteed to finish after $O(k^2)$ communication rounds in the distributed setting, and return a $(2k-1)$-spanner of expected size $O(kn^{1+1/k})$.

d-wise independence and derandomization Our main focus is devoted to the first phase that forms the clustering in a randomized manner. This phase does not work as is with reduced independence, because the bound on the spanner size relies on full independence between the coin flips of the clusters. However, we proceed by establishing properties of the Baswana–Sen algorithm (Algorithm 1) that do hold in the case of limited independence between its coin tosses. We use following result of Benjamini et al. [9].

Theorem 5 *Let $M(n, d, p)$ be the maximal probability of the AND event for n binary d-wise independent random variables, each with probability p of having the value 1. If d is even, then:*

$$M(n, d, p) \leq \frac{p^n}{\Pr[Bin(n, 1-p) \leq d/2]},$$

and if d is odd, then:

$$M(n, d, p) = pM(n-1, d-1, p).$$

We also use the following Chernoff bound for d-wise independent random variables from [57].

Theorem 6 *Let $X_1, \ldots, X_n$ be d-wise independent random variables taking values in $[0, 1]$, where $X = \sum_{i=1}^{n} X_i$ and $\mathbb{E}[X] = \mu$. Then for all $\epsilon \leq 1$ we have that if $d \leq \lfloor \epsilon^2 \mu e^{-1/3} \rfloor$ then:*

$$\Pr[|X - \mu| \geq \epsilon\mu] \leq e^{-\lfloor d/2 \rfloor}.$$

And if $d > \lfloor \epsilon^2 \mu e^{-1/3} \rfloor$ then:

$$\Pr[|X - \mu| \geq \epsilon\mu] \leq e^{-\lfloor \epsilon^2 \mu/3 \rfloor}.$$

We implement Algorithm 1 with only $O(\log n)$-wise independence between the coin tosses of clusters at each iteration. We also assume that $k \leq 0.5 \log n$. We prove the following two lemmas that will be used later for derandomization.

Let $d = 2 \log 2n$ be the independence parameter, and define $\xi = e^{1/3} 2 \log 2n$, and $\alpha_i = \prod_{j=1}^{i} (1 + 1/(k-j))$.

Lemma 8 *For every $1 \leq i \leq k-1$, if $|\mathcal{C}_{i-1}| \leq \xi \alpha_{i-1} n^{1-(i-1)/k}$, then $\Pr[|\mathcal{C}_i| \geq \xi \alpha_i n^{1-i/k}] < 0.5$. In addition, $\xi \alpha_{k-1} n^{1/k} = O(kn^{1/k} \log n)$.*

Proof We define for every cluster C the indicator random variable $X(C)$ for the event that the cluster remains for the next iteration. Note that $|\mathcal{C}_i| = \sum X(C)$ and $\mathbb{E}[X(C)] = n^{-1/k}$. By the assumption of the lemma, for the $(i-1)$-th iteration we know that we have at most $\xi \alpha_{i-1} n^{1-(i-1)/k}$ clusters left from the previous iteration. Thus $\mathbb{E}[\sum X(C)] \leq \xi \alpha_{i-1} n^{1-(i-1)/k} \cdot n^{-1/k} \leq \xi \alpha_{i-1} n^{1-i/k}$.

We wish to apply Theorem 6 with $d = 2 \log 2n$, $\mu_i = \xi \alpha_{i-1} n^{1-i/k}$ and $\epsilon_i = 1/(k-i)$. We note that $\alpha_i \geq 1$ for every i. We now show that it is always the case that $d \leq \lfloor \epsilon_i^2 \mu_i e^{-1/3} \rfloor$, so we can use the first case of Theorem 6. Plugging in ϵ_i, μ_i, d gives that we need to prove that:

$$2 \log 2n \leq 2 \log(2n) \cdot e^{1/3} e^{-1/3} \alpha_{i-1} n^{1-i/k} / (k-i)^2,$$

which holds if and only if

$$\alpha_{i-1} n^{1-i/k} / (k-i)^2 \geq 1.$$

We bound the left hand side from below by

$$\alpha_{i-1} n^{1-i/k} / (k-i)^2 \geq n^{1-i/k} / (k-i)^2.$$

To prove that the above is at least 1, we claim that (a) the function $n^{1-i/k}/(k-i)^2$ is monotonically decreasing for $1 \leq i \leq k-1$, and (b) that $n^{1-i/k}/(k-i)^2 \geq 1$ when $i = k-1$. To prove (a), we prove that $n^{1-i/k}/(k-i)^2 \leq n^{1-(i-1)/k}/(k-(i-1))^2$. Taking the square root of both sides gives that we need to prove that

$$n^{1/2-i/2k}/(k-i) \leq n^{1/2-(i-1)/2k}/(k-(i-1)),$$

which holds if and only if

$$(k-(i-1))/(k-i) \leq n^{1/2k}.$$

For the left hand side of the above, it holds that

$$(k-(i-1))/(k-i) \leq 1 + 1/(k-i) \leq 2,$$

and since we assumed that $k < 0.5 \log n$, we have that $n^{1/2k} \geq n^{1/\log n} = 2$. Therefore, $n^{1/2k} \geq (k-(i-1))/(k-i)$ as required for (a).

We now show (b), that is, that $n^{1-i/k}/(k-i)^2 \geq 1$ when $i = k-1$. This holds since for $i = k-1$ we have $n^{1-i/k}/(k-i)^2 = n^{1-(k-1)/k}/(k-(k-1))^2 = n^{1/k} \geq 1$, giving (b). This establishes that $d \leq \lfloor \epsilon_i^2 \mu_i e^{-1/3} \rfloor$, and thus the first condition of Theorem 6 always holds.

Since $\alpha_i = (1 + 1/(k-i))\alpha_{i-1} = (1+\epsilon_i)\alpha_{i-1}$ we have

$$\begin{aligned}
&\Pr[|\mathcal{C}_i| \geq \xi \alpha_i n^{1-i/k}] \\
&= \Pr\left[\sum X(C) \geq \xi(1+\epsilon_i)\alpha_{i-1} n^{1-i/k}\right] \\
&= \Pr\left[\sum X(C) \geq (1+\epsilon_i)\mu_i\right].
\end{aligned}$$

We now apply Theorem 6 and obtain

$$\Pr\left[\sum X(C) - \mu_i \geq \epsilon_i \mu_i\right] < e^{-\lfloor d/2 \rfloor} < 0.5,$$

which proves the first part of the lemma, that $\Pr[|\mathcal{C}_i| \geq \xi \alpha_i n^{1-i/k}] < 0.5$.

Finally, we have

$$\begin{aligned}
\alpha_{k-1} &= \prod_{j=1}^{k-1} (1 + 1/(k-j)) \\
&\leq e^{\sum_{j=1}^{i} 1/(k-j)} = e^{\sum_{j=1}^{k-1} 1/j} = O(k).
\end{aligned}$$

Which implies the second part of the lemma, that $\xi \alpha_{k-1} n^{1/k} = O(k n^{1/k} \log n)$, and completes the proof. □

Fix an iteration i and consider an i-unclustered vertex v. Denote by X_v the indicator variable for the event that vertex v adds more than $t = 2n^{1/k} \log n$ edges in this iteration.

Lemma 9 *The probability that there exists a vertex v at some iteration which adds more than t edges to the spanner is less than 0.5. Formally,* $\Pr[\vee_{v \in V} X_v = 1] < 0.5$.

Proof Let v be the node that maximizes $\Pr[X_v = 1]$. From the union bound it holds that $\Pr[\vee_{u \in V} X_u = 1] \leq \Pr[X_v = 1] \cdot n$. Next, we bound $\sum \Pr[X_v = 1]$. We show that every $\Pr[X_v = 1]$ is smaller than $1/2n$, completing the proof by applying a union bound over all vertices. Let ℓ be the number of neighboring clusters of v in $\mathcal{C}_{i-1}$. If $\ell \leq t$ then $\Pr[X_v = 1] = 0$. Otherwise, we might add t edges to H, if and only if the clusters corresponding to the t lightest edges in L are not in $\mathcal{C}_i$. This is the value $M(t, 2d, p)$ (we use $2d$ to avoid fractions in the binomial coefficient) with $p = 1 - n^{-1/k}$. Let us bound $M(t, 2d, p)$ as follows.

$$\begin{aligned}
M(t, 2d, p) &\leq \frac{p^t}{\Pr[Bin(t, 1-p) \leq d]} \\
&\leq \frac{p^t}{\binom{t}{d}(1-p)^d p^{t-d}} = \frac{p^d}{\binom{t}{d}(1-p)^d} \\
&\leq \frac{1}{\binom{t}{d}(1-p)^d} \leq \frac{1}{(t/d)^d (1-p)^d} \\
&\leq \frac{d^d}{t^d (1-p)^d}.
\end{aligned}$$

Plugging in $p = 1 - n^{-1/k}$ and $t = 2n^{1/k} \log n$ gives

$$\begin{aligned}
M(2n^{1/k} \log n, 2d, 1 - n^{-1/k}) &\leq \frac{d^d}{(2n^{1/k} \log n)^d (n^{-1/k})^d} \\
&= \frac{d^d}{(2 \log n)^d}.
\end{aligned}$$

Now let us plug in $d = 2 \log 2n$ and we get:

$$M(2n^{1/k} \log n, 2 \log 2n, 1 - n^{-1/k}) \leq (1/2)^{2 \log 2n} < 1/2n.$$

Finally, as explained, we use a union bound to get that $\Pr[\vee_{u \in V} X_u = 1] \leq \sum_{v \in V} \Pr[X_v = 1] < 0.5$. □

The above lemmas do not guarantee that the algorithm yields the same expected spanner size as the algorithm with full independence, but using these lemmas we can now construct a deterministic algorithm.

Let us define two bad events that can occur during some iteration i of the algorithm. Let A be the event that not enough clusters were removed during the iteration, and let B be the event that there exists a vertex that adds too many edges to the spanner. We will define these events formally later on. Let X_A, X_B be the corresponding indicator random variables for the events. Assume that it holds that $\mathbb{E}[X_A]+\mathbb{E}[X_B] < 1$. In this case we can use the method of conditional expectations in order to get an assignment to our random coins such that no bad event occurs.

Let $\bar{\rho}$ be the vector of coin flips used by the clusters. Let Y be the seed randomness from Lemma 1 used to generate $\bar{\rho}$ such that its entries are d-wise independent, where $d = O(\log n)$. We use Y to select a function $h \in \mathcal{H}_{\gamma,\beta}$, where $\gamma = \log n$ and $\beta = \log n^{1/k}$. Each vertex v uses Y to generate h and then uses the value $h(ID(v))$ to generate $\bar{\rho}[v]$.

Let $Z = (z_1, \ldots, z_n)$ be the final assignment generated by the method of conditional expectations. Then, $\mathbb{E}[X_A \mid Y = Z] + \mathbb{E}[X_B \mid Y = Z] < 1$. Because X_A and X_B are binary variables that are functions of Y, it must be the case that both are zero. We can write our expectation as follows:

$$\begin{aligned}\mathbb{E}[X_A] + \mathbb{E}[X_B] &= \Pr[X_A = 1] + \Pr[X_B = 1] \\ &= \Pr[X_A = 1] + \Pr[\vee X_v = 1]\end{aligned}$$

At every iteration of the algorithm we would like to keep $\mathbb{E}[X_A] + \mathbb{E}[X_B]$ below 1, which would guarantee both bad events do not occur. Unfortunately, it is unclear how to compute $\Pr[\vee X_v = 1]$ conditioned on some assignment to Y. Thus, we must use a pessimistic estimator. We consider $\sum_v \Pr[X_v = 1]$, and we have that:

$$\begin{aligned}&\Pr[X_A = 1] + \Pr[\vee X_v = 1] \\ &\leq \Pr[X_A = 1] + \sum_v \Pr[X_v = 1].\end{aligned}$$

We define our pessimistic estimator $\Psi = X_A + \sum_v X_v$. Note that the above inequality holds conditioned on any partial assignment to Y, because it is derived via a union bound. Thus, if we show that $\mathbb{E}[\Psi] = \Pr[X_A = 1] + \sum \Pr[X_v = 1] < 1$, it is enough to apply the method of conditional expectations for Ψ, keeping the expression below 1. For the assignment Z resulting from this process it will hold that $\mathbb{E}[X_A \mid Y = Z] + \mathbb{E}[X_B \mid Y = Z] < \mathbb{E}[\Psi \mid Y = Z] < 1$, as required.

It remains only to bound the pessimistic estimator Ψ. This can be achieved using Lemmas 8 and 9. In each iteration of the algorithm, because the bad event A did not occur in the previous iteration, the condition that $|\mathcal{C}_{i-1}| \leq \xi\alpha_{i-1}n^{1-(i-1)/k}$ holds for Lemma 9. This yields $\Pr[X_A = 1] + \sum \Pr[X_v = 1] < 1$.

The deterministic spanner construction in the congested clique We are now ready to describe our algorithm in the congested clique. We first show the we can indeed compute the conditional expectation of our pessimistic estimator Ψ.

We are interested in $\Pr[X_A = 1 \mid y_1 = b_1, \ldots, y_i = b_i]$ and $\Pr[X_v = 1 \mid y_1 = b_1, \ldots, y_i = b_i]$. Knowing some partial assignment to Y, we can iterate over all possible selections of $h \in \mathcal{H}_{\gamma,\beta}$ and compute the coin flip for every cluster using its ID alone. The expression $\Pr[X_A = 1 \mid y_1 = b_1, \ldots, y_i = b_i]$ is just the probability of enough clusters getting removed given some partial assignment. It does not depend on the graph topology, and can easily be computed by a vertex only knowing the IDs of clusters currently active. To compute $\Pr[X_v = 1 \mid y_1 = b_1, \ldots, y_i = b_i]$ the vertex v can collect all of the IDs from neighboring clusters and go over all possibilities for calculating the probability of adding too many edges.

Algorithm 3: deterministic $(2k-1)$-spanner algorithm

```
H = ∅
C_0 = {{v} | v ∈ V}
E' = E
for i from 1 to k do
    φ = ∅ //partial assignment
    foreach v ∈ V simultaneously do
        if v is cluster leader for C ∈ C_{i-1} then
            Send ID(v) to all vertices
        for j ∈ [log n] do
            for τ ∈ [log n] do
                Compute x_τ = Pr[X_v | φ, y_j = τ]
                Send (x_τ, τ) to u ∈ V, ID(u) = τ
            if ID(v) = τ, τ ∈ [log n] then
                s = Pr[X_A | φ, y_j = τ] + Σ_{(x,τ)} x
                Send (τ, s) to main leader
            if v is a leader then
                τ_min = argmin_τ {s | (τ, s)}
                φ = φ ∪ {y_j = τ_min}
                send updated φ to all vertices
    if i = k then
        C_i = ∅
    else
        C_i sampled from C_{i-1} using Y
    Run Algorithm 2
```

Algorithm 3 is the pseudocode, where before running an iteration of the Baswana–Sen algorithm we first find a seed randomness Y, such that both bad events A and B do not occur. We then execute an iteration of the Baswana–Sen algorithm using the seed to assign a random coin for each cluster. Because neither of the bad events occur in any iteration, no vertex adds more than $2n^{1/k}\log n$ edges in any iteration, and we reach the final iteration with $O(n^{1/k})$ clusters. Therefore, each iteration adds no more than $O(n^{1+1/k}\log n)$ edges, and the final iteration adds no more than $O(kn^{1+1/k}\log n)$ edges

(assuming a loose bound of having all vertices connect to all remaining clusters). Since in the second phase of the algorithm, we add one edge for each pair of vertex and cluster in $\mathcal{C}_{k-1}$, since there are $O(n^{1/k})$ such cluster, we conclude that our spanner has $O(kn^{1+1/k}\log n)$ edges.

We find Y via the method of conditional expectations, keeping the pessimistic estimator below 1. We consider the value of the pessimistic estimator under some partial assignment to Y, and extend the assignment such that the pessimistic estimator is kept below 1.

When finding Y we bring the power of the congested clique to our aid. The sequential approach would go over Y bit by bit, setting it to the value which optimizes the pessimistic estimator until all values of Y are fully set. In the congested cliques we can go over blocks of Y of size $\log n$, calculating the value of the pessimistic estimator for each one of the n possible assignments of the block. We achieve this by assigning each vertex to be responsible for aggregating the data in order to calculate the pessimistic estimator for one of the possible n values. This speeds up our calculation by a $\log n$ factor.

The above is implemented in the algorithm as follows: each vertex $v \in V$ iterates over all $\log n$ blocks of Y, each of size $\log n$. For each block it computes $\Pr[X_v]$ conditioned on all n values of the block. For every value τ of the block it sends each the conditional probability to u_τ which is responsible for computing the value of the pessimistic estimator conditioned on the value τ for the block. Knowing the conditional value of $\Pr[X_v]$ for every $v \in V$ and the IDs of the active clusters, the vertex u_τ can now compute the value of the conditional pessimistic estimator. All of the conditional values of the pessimistic estimator are then aggregated to a leader vertex which picks the value that minimizes the pessimistic estimator. Finally, the leader broadcasts the selected value for the block to all vertices. All vertices then continue to the next iteration. After computing Y we run an iteration of Baswana–Sen where the coin tosses of clusters are generated from Y.

Another benefit of running the Baswana–Sen algorithm in the congested clique is that we save an $O(k)$ factor in our round complexity. This is because cluster vertices may now communicate with the cluster leader directly, instead of propagating their message via other cluster vertices. This takes $O(k)$ in the standard distributed setting because the distance to the center of each cluster is at most the iteration number.

We conclude that the round complexity of our algorithm is the number of iterations of the Baswana–Sen main loop in the congested clique, which is $O(k)$, multiplied by the overhead of guaranteeing the bad events A, B will not happen during the iteration. We guarantee this by applying the method of conditional expectation over Y, using a block of size $\log n$ at each step of the method of conditional expectations.

We note that each cluster flips a biased coin with probability $n^{-1/k}$, and we require d-wise independence between the coin flips. We conclude from Lemma 1 that the size of Y is $O(d \max\left\{\log n^{1/k}, \log n\right\}) = O(\log^2 n)$ bits. Because, we pay $O(1)$ rounds for every $\log n$ chunk of Y, we conclude from the above that our algorithm takes a total of $O(k \log n)$ communication rounds. This completes the proof of Theorem 1.6.

6 Discussion

We have shown how to derandomize an MIS algorithm and a spanner construction in the congested clique model, and derandomize an MIS algorithm in the CONGEST model. This greatly improves upon the previously known results. Whereas our techniques imply that many local algorithms can be derandomized in the congested-clique (e.g., hitting set, ruling sets, coloring, matching etc.), the situation appears to be fundamentally different for global tasks such as connectivity, min-cut and MST. For instance, the best randomized MST algorithm in the congested-clique has time complexity of $O(1)$ rounds [37], but the best deterministic bound is $O(\log \log n)$ rounds [43]. Derandomization of such global tasks might require different techniques.

The importance of randomness in *local* computation lies in the fact that recent developments [15] show separations between randomized and deterministic complexities in the unlimited bandwidth setting of the LOCAL model. While some distributed algorithms happen to use small messages, our understanding of the impact of message size on the complexity of local problems is in its infancy.

This work opens a window to many additional intriguing questions. First, we would like to see many more local problems being derandomized despite congestion restrictions. Alternatively, significant progress would be made by otherwise devising deterministic algorithms for this setting. Finally, understanding the relative power of randomization with bandwidth restrictions is a worthy aim for future research.

Acknowledgements We are very grateful to Mohsen Ghaffari for many helpful discussions and useful observations involving the derandomization of his MIS algorithm.

A Pseudocode of the deterministic MIS algorithm

Let $\mathcal{H} = \mathcal{H}_{\gamma,\beta}$ with $\gamma = \Theta(\log n)$ and $\beta = \Theta(\log \Delta)$ be given by Lemma 1. Let $\mathcal{H}(Y_i) \subseteq \mathcal{H}_{\gamma,\beta}$ be the collection of all hash functions that agree with the partial seed Y_i. Each

function $h \in \mathcal{H}(Y_i)$ corresponds to a deterministic MIS algorithm.

For a hash function $h \in \mathcal{H}$ and value $p = 1/2^i$ representing the probability of a node to be marked, define $m_h(v, p) = 1$ if $h(ID(v)) \in [1, 2^{\beta-i}]$ and $m_h(v, p) = 0$ otherwise. That is, marking a node v with probability p is simulated deterministically by computing $m_h(v, p)$, since we output 1 only if the value $h(ID(v))$ appears in the top $1/2^i$ fraction of the range $[1, \beta]$. For $p = 1/2^i$ and $p' = 1/2^{i'}$, define $m_h(v, u, p, p') = 1$ if $h(ID(v)) \in [1, 2^{\beta-i}]$ and $h(ID(u)) \in [1, 2^{\beta-i'}]$ and $m_h(v, u, p, p') = 0$ otherwise. That is $m_h(v, u, p, p') = 1$ is the deterministic simulation of having both v and u being marked when v, u are marked with probability p, p'. Let $\alpha \in (0, 1]$ be the constant such that every golden node is removed with probability at least α when using pairwise independence. Algorithm 5 gives the pseudocode of our deterministic MIS algorithm.

Algorithm 4: `DetMIS`(t, Y_i): Code for node v in step i of phase t.

Input:
  A partial graph induced on the undecided nodes
  A partial assignment $Y_i = (y_1 = b_1, \ldots, y_i = b_i)$
Output:
  An assignment $Y_{i+1} = (y_1 = b_1, \ldots, y_i = b_i, y_{i+1} = b_{i+1})$
  // the goal is to extend the assignment for one more variable in the seed
for $u \in N(v) \cup \{v\}$ *and* $b \in \{0, 1\}$ **do**
  $m_{t,b}(u) \leftarrow \sum_{h \in \mathcal{H}(Y_{i,b})} m_h(u, p_t(u))$
for $u \in N(v)$, *and* $b \in \{0, 1\}$ **do**
  $m_{t,b}(v, u) \leftarrow \sum_{h \in \mathcal{H}(Y_{i,j})} m_h(v, u, p_t(v), p_t(u))$
$M_{t,b}(v) \leftarrow \sum_{u \in N(u)} m_{t,b}(v, u)$
Exchange $M_{t,0}(v)$, $M_{t,1}(v)$ with neighbors
if v *is a golden type-1 node* **then**
  $\chi(v, b) \leftarrow m_{t,b}(v) - M_{t,b}(v)$ for $b \in \{0, 1\}$
  // $\chi(v, b)$ corresponds to $\mathbb{E}(\psi_{v,t} \mid Y_{i,b})$
if v *is a golden type-2 node* **then**
  Define $W(v) \subseteq N(v)$ such that $\sum_{u \in W(v)} p_t(u) \in [1/40, 1/4]$
  for $u \in W(v)$, *and* $b \in \{0, 1\}$ **do**
    $M_{t,b}(u, W(v)) \leftarrow \sum_{h \in \mathcal{H}(Y_{i,j})} \sum_{w \in W(v) \setminus \{u\}} m_h(u, w, p_t(u), p_t(w))$
  $\chi(v, b) \leftarrow \sum_u m_{t,b}(u) - M_{t,b}(u) - M_{t,b}(u, W(v))$
  // $\chi(v, b)$ corresponds to $\mathbb{E}(\psi_{v,t} \mid Y_{i,b})$
if v *is golden* **then**
  Send to the leader $(1 + \alpha)^{age(v)} \cdot \chi(v, b)$ for $b \in \{0, 1\}$
  // All of the above fits in an $O(\log n)$-bit message
  $age(v) \leftarrow age(v) + 1$
Receive j^* from the leader
$y_{i+1} \leftarrow j^*$

Algorithm 5: `DetMIS`(t): Code for node v in phase t.

Input:
  A partial graph induced on the undecided nodes
Output:
  Decide whether to join the MIS, be removed from the graph, or remain undecided
Let $p_t(v)$ be the desired level for joining MIS, initially $p_0(v) \leftarrow 1/2$
Let $d_t(v) = \sum_{u \in N(v)} p_t(u)$ be the effective degree of node v in phase t
Let $age(v)$ be the number of golden phases it had so far. Initially, $age(v) \leftarrow 0$
Exchange the $p_t(v)$, $d_t(v)$ and ID with neighbors
$Y_0 \leftarrow \emptyset, \beta \leftarrow \Theta(\log n)$
for $i = 0, \ldots, \beta$ **do**
  $(b_{i+1}) \leftarrow$ `DetMIS`(t, Y_i)
  $Y_{i+1} \leftarrow (y_1 = b_1, \ldots, y_{i+1} = b_{i+1})$
Let h_t be selected from $\mathcal{H}_{a,\beta}$ using Y_β
Let $m_t(v) \leftarrow m_{h_t}(v, p_t(v))$
Exchange $m_t(v)$ with neighbor
if $m_t(v) = 1$ *and none of your neighbors has* $m_t(u) = 1$ **then**
  join MIS
  notify neighbors
if $\exists$ *neighbor that joins MIS* **then**
  notify neighbors of being removed from graph

References

1. Alon, N., Babai, L., Itai, A.: A fast and simple randomized parallel algorithm for the maximal independent set problem. J. Algorithms **7**(4), 567–583 (1986)
2. Awerbuch, B., Goldberg, A.V., Luby, M., Plotkin, S.A.: Network decomposition and locality in distributed computation. In: FOCS, pp. 364–369 (1989)
3. Barenboim, L.: Deterministic $(\Delta + 1)$-coloring in sublinear (in Δ) time in static, dynamic and faulty networks. In: PODC, pp. 345–354 (2015)
4. Barenboim, L., Elkin, M.: Distributed graph coloring: fundamentals and recent developments. Synth. Lect. Distrib. Comput. Theory **4**(1), 1–171 (2013)
5. Barenboim, L., Elkin, M., Gavoille, C.: A fast network-decomposition algorithm and its applications to constant-time distributed computation. In: SIROCCO, pp. 209–223 (2015)
6. Barenboim, L., Elkin, M., Kuhn, F.: Distributed $(\Delta + 1)$-coloring in linear (in Δ) time. SIAM J. Comput. **43**(1), 72–95 (2014)
7. Barenboim, L., Elkin, M., Pettie, S., Schneider, J.: The locality of distributed symmetry breaking. In: FOCS, pp. 321–330 (2012)
8. Baswana, S., Sen, S.: A simple and linear time randomized algorithm for computing sparse spanners in weighted graphs. Random Struct. Algorithms **30**(4), 532–563 (2007)
9. Benjamini, I., Gurel-Gurevich, O., Peled, R.: On k-wise independent distributions and Boolean functions. arXiv preprint arXiv:1201.3261 (2012)
10. Berger, B., Rompel, J.: Simulating (log CN)-wise independence in NC. J. ACM (JACM) **38**(4), 1026–1046 (1991)

11. Brandt, S., Fischer, O., Hirvonen, J., Keller, B., Lempiäinen, T., Rybicki, J., Suomela, J., Uitto, J.: A lower bound for the distributed Lovász local lemma. In: STOC, pp. 479–488 (2016)
12. Censor-Hillel, K., Kaski, P., Korhonen, J.H., Lenzen, C., Paz, A., Suomela, J.: Algebraic methods in the congested clique. In: PODC, pp. 143–152 (2015)
13. Chandrasekaran, K., Goyal, N., Haeupler, B.: Deterministic algorithms for the Lovász local lemma. SIAM J. Comput. **42**(6), 2132–2155 (2013)
14. Chang, Y.-J., Fischer, M., Ghaffari, M., Uitto, J., Zheng, Y.: The complexity of $(\Delta + 1)$ coloring in congested clique, massively parallel computation, and centralized local computation. In: Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29–August 2, 2019, pp. 471–480 (2019)
15. Chang, Y.-J., Kopelowitz, T., Pettie, S.: An exponential separation between randomized and deterministic complexity in the LOCAL model. In: FOCS, pp. 615–624 (2016)
16. Derbel, B., Gavoille, C.: Fast deterministic distributed algorithms for sparse spanners. In: SIROCCO, pp. 100–114 (2006)
17. Derbel, B. Gavoille, C., Peleg, D.: Deterministic distributed construction of linear stretch spanners in polylogarithmic time. In: DISC, pp. 179–192 (2007)
18. Derbel, B., Gavoille, C., Peleg, D., Viennot, L.: On the locality of distributed sparse spanner construction. In: PODC, pp. 273–282 (2008)
19. Derbel, B., Gavoille, C., Peleg, D., Viennot, L.: Local computation of nearly additive spanners. In: DISC, pp. 176–190 (2009)
20. Derbel, B., Mosbah, M., Zemmari, A.: Sublinear fully distributed partition with applications. Theory Comput. Syst. **47**(2), 368–404 (2010)
21. Dolev, D., Lenzen, C., Peled, S.: "tri, tri again": finding triangles and small subgraphs in a distributed setting—(extended abstract). In: DISC, pp. 195–209 (2012)
22. Drucker, A., Kuhn, F., Oshman, R.: On the power of the congested clique model. In: PODC, pp. 367–376 (2014)
23. Elkin, M.: A near-optimal distributed fully dynamic algorithm for maintaining sparse spanners. In: PODC, pp. 185–194 (2007)
24. Feuilloley, L., Fraigniaud, P.: Randomized local network computing. In: SPAA, pp. 340–349 (2015)
25. Le Gall, F.: Further algebraic algorithms in the congested clique model and applications to graph-theoretic problems. In: DISC, pp. 57–70 (2016)
26. Ghaffari, M.: An improved distributed algorithm for maximal independent set. In: SODA, pp. 270–277 (2016)
27. Ghaffari, M.: Distributed MIS via all-to-all communication. In: Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2017, Washington, DC, USA, July 25–27, 2017, pp. 141–149 (2017)
28. Ghaffari, M., Parter, M.: MST in log-star rounds of congested clique. In: PODC, pp. 19–28 (2016)
29. Goldberg, M., Spencer, T.: A new parallel algorithm for the maximal independent set problem. SIAM J. Comput. **18**(2), 419–427 (1989)
30. Han, Y.: A fast derandomization scheme and its applications. SIAM J. Comput. **25**(1), 52–82 (1996)
31. Hegeman, J.W., Pandurangan, G., Pemmaraju, S.V., Sardeshmukh, V.B., Scquizzato, M.: Toward optimal bounds in the congested clique: graph connectivity and MST. In: PODC, pp. 91–100 (2015)
32. Hegeman, J.W., Pemmaraju, S.V.: Lessons from the congested clique applied to mapreduce. In: SIROCCO, pp. 149–164 (2014)
33. Hegeman, J.W., Pemmaraju, S.V., Sardeshmukh, V.: Near-constant-time distributed algorithms on a congested clique. In: DISC, pp. 514–530 (2014)
34. Henzinger, M., Krinninger, S., Nanongkai, D.: A deterministic almost-tight distributed algorithm for approximating single-source shortest paths. In: STOC, pp. 489–498 (2016)
35. Holzer, S., Pinsker, N.: Approximation of distances and shortest paths in the broadcast congest clique. In: OPODIS, pp. 6:1–6:16 (2015)
36. Israeli, A., Itai, A.: A fast and simple randomized parallel algorithm for maximal matching. Inf. Process. Lett. **22**(2), 77–80 (1986)
37. Jurdzinski, T., Nowicki, K.: MST in $O(1)$ rounds of congested clique. In: SODA, pp. 2620–2632. SIAM (2018)
38. Karp, R.M., Wigderson, A.: A fast parallel algorithm for the maximal independent set problem. In: STOC, pp. 266–272 (1984)
39. Kuhn, F., Moscibroda, T., Wattenhofer, R.: Local computation: lower and upper bounds. J. ACM **63**(2), 17 (2016)
40. Lenzen, C.: Optimal deterministic routing and sorting on the congested clique. In: PODC, pp. 42–50 (2013)
41. Lenzen, C., Wattenhofer, R.: Tight bounds for parallel randomized load balancing: extended abstract. In: STOC, pp. 11–20 (2011)
42. Linial, N.: Locality in distributed graph algorithms. SIAM J. Comput. **21**(1), 193–201 (1992)
43. Lotker, Z., Pavlov, E., Patt-Shamir, B., Peleg, D.: MST construction in O(log $logn$) communication rounds. In: SPAA, pp. 94–100 (2003)
44. Luby, M.: A simple parallel algorithm for the maximal independent set problem. SIAM J. Comput. **15**(4), 1036–1053 (1986)
45. Luby, M.: Removing randomness in parallel computation without a processor penalty. J. Comput. Syst. Sci. **47**(2), 250–286 (1993)
46. Motwani, R., Naor, J., Naor, M.: The probabilistic method yields deterministic parallel algorithms. J. Comput. Syst. Sci. **49**(3), 478–516 (1994)
47. Nanongkai, D.: Distributed approximation algorithms for weighted shortest paths. In: STOC, pp. 565–573 (2014)
48. Naor, M., Stockmeyer, L.J.: What can be computed locally? SIAM J. Comput. **24**(6), 1259–1277 (1995)
49. Panconesi, A., Srinivasan, A.: Improved distributed algorithms for coloring and network decomposition problems. In: STOC, pp. 581–592 (1992)
50. Pantziou, G., Spirakis, P., Zaroliagis, C.: Fast parallel approximations of the maximum weighted cut problem through derandomization. In: FSTTCS, pp. 20–29 (1989)
51. Parter, M.: $(\Delta+1)$ coloring in the congested clique model. In: 45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9–13, 2018, Prague, Czech Republic, pp. 160:1–160:14 (2018)
52. Parter, M., Su, H.-H.: Randomized $(\Delta+1)$-coloring in $O(\log *\Delta)$ congested clique rounds. In: 32nd International Symposium on Distributed Computing, DISC 2018, New Orleans, LA, USA, October 15–19, 2018, pp. 39:1–39:18 (2018)
53. Patt-Shamir, B., Teplitsky, M.: The round complexity of distributed sorting: extended abstract. In: PODC, pp. 249–256 (2011)
54. Pemmaraju, S.V.: Equitable coloring extends Chernoff-hoeffding bounds. In: Approximation, Randomization and Combinatorial Optimization: Algorithms and Techniques, 4th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2001 and 5th International Workshop on Randomization and Approximation Techniques in Computer Science, RANDOM 2001 Berkeley, CA, USA, August 18–20, 2001, Proceedings, pp. 285–296 (2001). https://doi.org/10.1007/3-540-44666-4_31
55. Pettie, S.: Distributed algorithms for ultrasparse spanners and linear size skeletons. Distrib. Comput. **22**(3), 147–166 (2010)
56. Roditty, L., Thorup, M., Zwick, U. Deterministic constructions of approximate distance oracles and spanners. In: ICALP, pp. 261–272 (2005)

57. Schmidt, J.P., Siegel, A., Srinivasan, A.: Chernoff-hoeffding bounds for applications with limited independence. SIAM J. Discrete Math. **8**(2), 223–250 (1995)
58. Srivastav, A., Kliemann, L.: Parallel algorithms via the probabilistic method. In: Rajasekaran, S. (ed.) Handbook of Parallel Computing: Models. Algorithms and Applications. Chapman and Hall/CRC, Boca Raton (2007)
59. Suomela, J.: Survey of local algorithms. ACM Comput. Surv. **45**(2), 24 (2013)
60. Vadhan, S.P.: Pseudorandomness. Found. Trends Theor. Comput. Sci. **7**(1–3), 1–336 (2012)