Derandomizing Local Distributed Algorithms under Bandwidth Restrictions*†

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

1 **Department of Computer Science, Technion, Haifa, Israel**
ckeren@cs.technion.ac.il
2 **MIT, CSAIL, Cambridge, USA**
parter@mit.edu
3 **Department of Computer Science, Technion, Haifa, Israel**
gregory.schwartzman@gmail.com

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. We combine bounded independence, which we show to be sufficient for some algorithms, with the method of conditional expectations and with additional machinery, to obtain the following results.

First, we show that in the *Congested Clique* model, which allows all-to-all communication, there is a deterministic maximal independent set (MIS) 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 $O(\log \Delta)$.

Adapting the above to the CONGEST model gives an $O(D \log^2 n)$-round deterministic MIS algorithm, where D is the diameter of the graph. Apart from a previous unproven claim of a $O(D \log^3 n)$-round algorithm, the only known deterministic solutions for the CONGEST model are a coloring-based $O(\Delta + \log^* n)$-round algorithm, where Δ is the maximal degree in the graph, and a $2^{O(\sqrt{\log n \log\log n})}$-round algorithm, which is super-polylogarithmic in n.

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. For comparison, in the more stringent CONGEST model, where the communication graph is identical to the input graph, the best deterministic algorithm for constructing a $(2k-1)$-spanner with $O(kn^{1+1/k})$ edges runs in $O(n^{1-1/k})$ rounds.

1998 ACM Subject Classification G.2.2 Graph Algorithms

Keywords and phrases Local problems, congested clique, derandomization

Digital Object Identifier 10.4230/LIPIcs.DISC.2017.11

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

* This research is partially supported by the Israel Science Foundation (grant 1696/14).
† A full version of the paper is available at https://arxiv.org/abs/1608.01689.

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. [14] 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. [10], 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 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.

The curious phenomenon that shows up here is that the derandomization toolbox that was developed for sequential algorithms does not seem to lend itself for the LOCAL model, but it can be used in the congested clique model. This allows us to obtain deterministic algorithms for local problems in the congested clique model, whose complexities roughly match the complexities of their randomized counterparts in the LOCAL model. This can be contrasted with the exponential in Δ or near-exponential in n gaps between the deterministic and randomized complexities of these problems in the LOCAL model alone.

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, in this algorithm, nodes choose to mark themselves with probabilities that evolve depending on the previous probabilities of neighbors. In particular, if the sum of marking probabilities of a vertex's neighbors is large (resp., small) – the vertex reduces (resp. , increases) its own marking probability in the next round. 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 graph consists of a convenient decomposition into small clusters for which the problem can be solved fast. This is called the shattering phenomena (see e.g., [7]).

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 [27]) to $O(\log \Delta)$ rounds.

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

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

For the derandomization, we use the method of conditional expectations (see e.g., [44, Chapter 6.3]). In our context, this shows the existence of an assignment to the random choices made by the nodes that attains the desired property of removing a sufficiently large part of the graph in each iteration, where removal is due to a node already having an output (whether the vertex is in the MIS or not). As in many uses of this method, we need to reduce the number of random choices that are made in order to be able to efficiently *compute* the above assignment.

However, we need to overcome several obstacles. First, we need to reduce the search space of a good assignment to the random choices of the nodes, by showing that pairwise independence (see, e.g., [44, Chapter 13]) is sufficient for the algorithm to work. Unfortunately, this does not hold directly in the original algorithm.

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 this removal-probability guarantee holds also with pairwise independence upon our slight adaptation of the constants used by the algorithm.

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 [38]. 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 $O(\log n)$. 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.

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. In fact, the algorithm works in the more restricted *Broadcast Congested Clique* model, in which a node must send the same $O(\log n)$-bit message to all other nodes in any single round. Carefully placing all the pieces of the toolbox we develop, gives the following.

▶ **Theorem 2.** *There is a deterministic MIS algorithm for the broadcast 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. The proof of the following is deferred to the full version [12].

▶ **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 $\log\Delta$. We conclude that 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. The state of the art for that setting is $O(2^{\sqrt{\log n\log\log n}})$ round algorithm, using the network decomposition of [5] (see Cor. 5.4 there). Wes then 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 4.** *There is a deterministic MIS algorithm for the CONGEST model that completes in* $\min\{O(D\log^2 n), O(2^{\sqrt{\log n\log\log n}})\}$ *rounds.*

The significance of the latter is that it is the first deterministic MIS algorithm in CONGEST to have only a polylogarithmic gap compared to its randomized counterpart when D is polylogarithmic. Notice that this logarithmic complexity is the best that is known even in the LOCAL model. In [49] it is shown that an MIS can be computed deterministically in $2^{O(\sqrt{\log n})}$ rounds via network decomposition, which is super-polylogarithmic in n. Moreover, the algorithm requires large messages and hence is unsuitable for CONGEST. Focusing on deterministic algorithms in CONGEST, the only known non-trivial solution 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, e.g., [7], but the reduction has to pay for the number of colors anyhow). Our $O(D\log^2 n)$-round MIS algorithm is therefore unique in its parameters.

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 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 5.** *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.*

As in the MIS algorithm, 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. The entire spanner construction is deferred to the full version [12].

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 [42], Alon et al. [1] and Israeli and Itai [35]. [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.

The best deterministic algorithm is by Panconesi and Srinivasan [48], and completes in $2^{O(\sqrt{\log n})}$ rounds. On the lower bound side, Linial [40] gave an $\Omega(\log^* n)$ lower bounds for 3-coloring the ring, which also applies to finding an MIS. Kuhn et al. [37] 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 [55].

Distributed constructions of spanners. The construction of spanners in the distribute setting has been studied extensively both in the randomized and deterministic setting [8, 15, 16, 17, 18, 52]. We emphasize that the construction of [18] 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 [53]. We emphasize that we need entirely different techniques to derandomize the Baswana-Sen algorithm compared with the centralized derandomization of [53].

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 decomposition. The state of the art is due to Derbel et al [17]. 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. [19] 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. [41], 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 [21], faster MST algorithms [27, 30], distance computation [33, 34, 46], subgraph detection [20], algebraic computations [11, 25], and routing and sorting [38, 39, 51]. Local problems were addressed in [32] who study ruling sets. Connections to the MapReduce model is given in [31].

Derandomization in the parallel setting. Derandomization of local algorithms has attracted much attention in the *parallel* setting [1, 9, 13, 28, 29, 35, 36, 45, 50, 54]. Luby [43] 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, 35]. Berger and Rompel [9] 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.

Derandomization in the distributed setting. Naor and Stockmeyer [47] showed that constant-round randomized algorithms for problems that are locally checkable can be derandomized without an asymptotic overhead, extended by [14, 24] for larger time complexities and for a wider range of problems. Awerbuch et al. [2] claim to use the derandomized MIS algorithm of Luby [43] to obtain a deterministic CONGEST MIS algorithm. This claim is, however, not supported in their paper and is also late stated as open in [22].

2 Preliminaries and Notation

Our derandomization approach consists of first reducing 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 [56].

▶ **Definition 6** ([56, 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 $x_1, x_2, ..., x_d \in [N]$, the random variables $H(x_1), ..., H(x_d)$ are independent and uniformly distributed in $[M]$ for a randomly chosen H in $\mathcal{H}$.

In [56] an explicit construction of $\mathcal{H}$ is presented, with parameters as stated next.

▶ **Lemma 7** ([56, Corollary 3.34])**.** *For every $\gamma, \beta, d \in \mathbb{N}$, there is a family of d-wise independent functions $\mathcal{H}_{\gamma,\beta} = \{h : \{0,1\}^\gamma \to \{0,1\}^\beta\}$ 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 7 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 $t = d \cdot \max\{\gamma, \beta\}$ 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 look at Y as a vector of length $t/\log n$ where each entry takes values in $\{1, \ldots, \lceil \log n \rceil\}$. 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 6 it holds that $Pr[h(ID(v)) = 0] = p$, as needed.

The method of conditional expectations. Next, we consider the method of conditional expectations. Let $\phi : A^\ell \to \mathbb{R}$, and let $X = (X_1, ..., X_\ell)$ be a vector of random variables taking values in A. If $E[\phi(X)] \geq \alpha$ then there is an assignment of values $Z = (z_1, ..., z_\ell)$ 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 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 from above 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 [38], 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 Deterministic MIS

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.

We define a slight modification to the first part of Ghaffari's MIS Algorithm: Set $p_0(v) = 1/4$. Define $p_{t+1}(v) = 1/2 \cdot p_t(v)$, if $d_t(v) \geq 1/2$ and $p_{t+1}(v) = \min\{2p_t(v), 1/4\}$, otherwise. Here $d_t(v) = \sum_{u \in N(v)} p_t(u)$ 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 $O(\log \Delta)$ 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 [38], the second part can be completed in $O(1)$ rounds w.h.p. We note that the proof that $O(n)$ edges remain 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 in the following section new machinery. The full proof Thm. 1 appears in the full version [12].

3.2 Derandomizing the modified MIS algorithm

3.2.1 Ghaffari's algorithm with pairwise independence

We review the main terminology and notation from [26], up to our modification of constants. Changing the constants is important as we are using pairwise independence and not complete independence as in the original algorithm of Ghaffari. A node v is called *light* if $d_t(v) < 1/4$. We define two types of *golden phases* for a node v. This is a modification of the corresponding definitions in [26].

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. The proof of the following appears in the full version [12].

▶ **Lemma 8** (golden nodes with pairwise independence)**.** *Consider the modified Ghaffari's algorithm with pairwise independent coin flips.*

(1) *If t is a type-1 golden phase for a node v, then v joins the MIS in phase t with probability at least $1/8$.*

(2) *If t is a type-2 golden phase for a node v then v is removed in phase t with probability at least $\alpha = 1/160$.*

As a result, the following holds in the pairwise independence setting:

▶ **Lemma 9.** *Within $O(\log \Delta)$ phases, every node remains with probability at most $1/\Delta$.*

Recall that the proof from Subsection 3.1 that $O(n)$ edges remain 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.

3.2.2 $O(\log n \log \Delta)$-round deterministic MIS in the congested clique

Using derandomization we show there is a deterministic MIS algorithm for the broadcast congested clique model that completes in $O(\log \Delta \log n)$ rounds, as stated in Theorem 2.

3.2.2.1 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. Towards derandomizing the algorithm, for each node, we define the corresponding random variables $x_1, \ldots, x_{n'}$ indicating whether v_i is marked in phase t. Let $X_i = (x_1 = b_1, \ldots, x_i = b_i)$ define a partial assignment for $v_1, \ldots, v_i$ (i.e., whether or not they are in the MIS in phase t). Let $X_0 = \emptyset$ denote the case where none of the decisions is fixed.

For a golden node v (in phase t), 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 Lemma 8, there is a constant c such that $\mathbb{E}(R_t) \geq c \cdot |V_t|$. Potentially, we could aim for the following: Given the partial assignment X_i, compute the two expectations of the number of removed golden nodes conditioned on the two possible assignments for x_{i+1}, $\mathbb{E}(R_t \mid X_i, x_{i+1} = 0)$ and $\mathbb{E}(R_t \mid X_i, x_{i+1} = 1)$, and choose x_{i+1} according to the larger expectation.

However, towards the above goal we face the following main challenges. (C1) The value of R_t cannot be easily computed, since when using probabilities of neighboring nodes we might be double-counting: a node might be removed while having more than a single neighbor that joins the MIS. (C2) The search space of size 2^n is too large and in particular, the conditional expectation computation consists of n steps. (C3) Even when using pairwise independence to enjoy an $O(\log n)$-bit seed, searching a good seed point in a space of size $O(\text{poly } n)$ in a brute force manner cannot be done efficiently in the congested clique. (C4) Despite our proof that golden nodes are removed with constant probability even with pairwise independence, it is still not clear how to implement the second part of the MIS algorithm, because showing that only $O(n/\Delta)$ nodes survive cannot be done with pairwise independence. That is, the proof from Subsection 3.1 that $O(n)$ edges remain inherently needs full independence.

Addressing (C4) requires 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 tools and then provide our algorithm.

3.2.2.2 Derandomization tools

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 [42, 43], 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 if 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. This allows coping with challenge (C1).

Let $m_{v,t}$ be the random variable indicating the event that v is *marked*. Let $m_{v,u,t}$ indicate the event that both u and v are marked. Define $\psi_{v,t} = m_{v,t} - \sum_{u \in N(v)} m_{v,u,t}$ if v is of type-1, and $\psi_{v,t} = \sum_{u \in N(v)}(m_{u,t} - \sum_{w \in N(u)} m_{u,w,t} - \sum_{w' \in N(v) \setminus \{u\}} m_{u,w',t})$, if v is of type-2. 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 in the MIS, we have[2]

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

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 8). By Lemma 8, 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.

[2] For ease of presentation, here we condition on a n-length vector. However, our algorithm will use the pairwise independence – discussed in the following paragraph – to condition on a seed of length $O(\log n)$.

Tool 2: Pairwise independence. We now combine the method of conditional expectations with a small search space. We use Lemma 7 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 7, 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.[3] 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. This standard approach takes care of challenge (C2).

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 $Y_i = (y_1 = b_1, \ldots, y_i = b_i)$ 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 Equation (1), but we condition on the small seed Y_i rather than on X_i. This addresses challenge (C3).

Tool 3: An age-based weighted adaptation. To handle challenge (C4), 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 $age(v)$ 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 $\psi'_{v,t} = (1/(1-\alpha))^{age(v)} \cdot \psi_{v,t}$, 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.

3.2.2.3 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.

[3] 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 $y \in [1, 2^{b-i}]$.

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. 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 Equation (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 Lemma 8. 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 full version [12] 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 8. 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 $W(v)$.[4] 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 Equation (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. A pseudocode appears in the full version [12].

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 Lemma 15 in the full paper attached, which justifies the definition of the expectation given in Equation (2). 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 [38]. 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] 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.

3.3 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 [48] 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 4** (restated). *There is a deterministic MIS algorithm for the CONGEST model that completes in* $\min\{O(D \log^2 n), O(2^{\sqrt{\log n \log\log n}})\}$ *rounds.*

Proof. The bound of $2^{O(\sqrt{\log n \log\log n})}$ follows by the network decomposition of [5]. To get a bound of $O(D \log^2 n)$ rounds, we use a similar algorithm to 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 eariler 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 $O(D \log^2 n)$. For the correctness, we assume towards contradiction that after $\Omega(\log n)$ rounds, at least one node remains undecided. Then, we show that every node that survives can charge $\Omega(n^c)$ nodes that are removed, which is a contradiction as there only n nodes. ◀

4 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 $O(\log^* n)$ rounds [27], but the best deterministic bound is $O(\log \log n)$ rounds [41]. Derandomization of such global tasks might require different techniques.

The importance of randomness in *local* computation lies in the fact that recent developments [14] 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.

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

References

1 Noga Alon, László Babai, and Alon Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of algorithms*, 7(4):567–583, 1986.

2 Baruch Awerbuch, Andrew V. Goldberg, Michael Luby, and Serge A. Plotkin. Network decomposition and locality in distributed computation. In *FOCS*, pages 364–369, 1989.

3 Leonid Barenboim. Deterministic $(\Delta + 1)$-coloring in sublinear (in Δ) time in static, dynamic, and faulty networks. *J. ACM*, 63(5):47:1–47:22, 2016.

4 Leonid Barenboim and Michael Elkin. Distributed graph coloring: Fundamentals and recent developments. *Synthesis Lectures on Distributed Computing Theory*, 4(1):1–171, 2013.

5 Leonid Barenboim, Michael Elkin, and Cyril Gavoille. A fast network-decomposition algorithm and its applications to constant-time distributed computation. In *SIROCCO*, pages 209–223, 2015.

6 Leonid Barenboim, Michael Elkin, and Fabian Kuhn. Distributed (delta+1)-coloring in linear (in delta) time. *SIAM J. Comput.*, 43(1):72–95, 2014.

7 Leonid Barenboim, Michael Elkin, Seth Pettie, and Johannes Schneider. The locality of distributed symmetry breaking. *J. ACM*, 63(3):20:1–20:45, 2016.

8 Surender Baswana and Sandeep Sen. A simple and linear time randomized algorithm for computing sparse spanners in weighted graphs. *Random Structures & Algorithms*, 30(4):532–563, 2007.

9 Bonnie Berger and John Rompel. Simulating (log cn)-wise independence in nc. *Journal of the ACM (JACM)*, 38(4):1026–1046, 1991.

10 Sebastian Brandt, Orr Fischer, Juho Hirvonen, Barbara Keller, Tuomo Lempiäinen, Joel Rybicki, Jukka Suomela, and Jara Uitto. A lower bound for the distributed lovász local lemma. In *STOC*, pages 479–488, 2016.

11 Keren Censor-Hillel, Petteri Kaski, Janne H. Korhonen, Christoph Lenzen, Ami Paz, and Jukka Suomela. Algebraic methods in the congested clique. In *PODC*, pages 143–152, 2015.

12 Keren Censor-Hillel, Merav Parter, and Gregory Schwartzman. Derandomizing local distributed algorithms under bandwidth restrictions. *arXiv preprint arXiv:1608.01689*, 2016. URL: `https://arxiv.org/abs/1608.01689`.

13 Karthekeyan Chandrasekaran, Navin Goyal, and Bernhard Haeupler. Deterministic algorithms for the lovász local lemma. *SIAM Journal on Computing*, 42(6):2132–2155, 2013.

14 Yi-Jun Chang, Tsvi Kopelowitz, and Seth Pettie. An exponential separation between randomized and deterministic complexity in the LOCAL model. In *FOCS*, pages 615–624, 2016.

15 Bilel Derbel and Cyril Gavoille. Fast deterministic distributed algorithms for sparse spanners. *Theor. Comput. Sci.*, 399(1-2):83–100, 2008.

16 Bilel Derbel, Cyril Gavoille, and David Peleg. Deterministic distributed construction of linear stretch spanners in polylogarithmic time. In *DISC*, pages 179–192, 2007.

17 Bilel Derbel, Cyril Gavoille, David Peleg, and Laurent Viennot. On the locality of distributed sparse spanner construction. In *PODC*, pages 273–282, 2008.

18 Bilel Derbel, Cyril Gavoille, David Peleg, and Laurent Viennot. Local computation of nearly additive spanners. In *DISC*, pages 176–190, 2009.

19 Bilel Derbel, Mohamed Mosbah, and Akka Zemmari. Sublinear fully distributed partition with applications. *Theory of Computing Systems*, 47(2):368–404, 2010.

20 Danny Dolev, Christoph Lenzen, and Shir Peled. "tri, tri again": Finding triangles and small subgraphs in a distributed setting - (extended abstract). In *DISC*, pages 195–209, 2012.

21 Andrew Drucker, Fabian Kuhn, and Rotem Oshman. On the power of the congested clique model. In *PODC*, pages 367–376, 2014.

22 Michael Elkin. A faster distributed protocol for constructing a minimum spanning tree. *J. Comput. Syst. Sci.*, 72(8):1282–1308, 2006.

23 Michael Elkin. A near-optimal distributed fully dynamic algorithm for maintaining sparse spanners. In *PODC*, pages 185–194, 2007.

24 Laurent Feuilloley and Pierre Fraigniaud. Randomized local network computing. In *SPAA*, pages 340–349, 2015.

25 François Le Gall. Further algebraic algorithms in the congested clique model and applications to graph-theoretic problems. In *DISC*, pages 57–70, 2016.

26 Mohsen Ghaffari. An improved distributed algorithm for maximal independent set. In *SODA*, pages 270–277, 2016.

27 Mohsen Ghaffari and Merav Parter. Mst in log-star rounds of congested clique. In *PODC*, pages 19–28, 2016.

28 Mark Goldberg and Thomas Spencer. A new parallel algorithm for the maximal independent set problem. *SIAM Journal on Computing*, 18(2):419–427, 1989.

29 Yijie Han. A fast derandomization scheme and its applications. *SIAM Journal on Computing*, 25(1):52–82, 1996.

30 James W. Hegeman, Gopal Pandurangan, Sriram V. Pemmaraju, Vivek B. Sardeshmukh, and Michele Scquizzato. Toward optimal bounds in the congested clique: Graph connectivity and MST. In *PODC*, pages 91–100, 2015.

31 James W. Hegeman and Sriram V. Pemmaraju. Lessons from the congested clique applied to mapreduce. In *SIROCCO*, pages 149–164, 2014.

32 James W. Hegeman, Sriram V. Pemmaraju, and Vivek Sardeshmukh. Near-constant-time distributed algorithms on a congested clique. In *DISC*, pages 514–530, 2014.

33 Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. A deterministic almost-tight distributed algorithm for approximating single-source shortest paths. In *STOC*, pages 489–498, 2016.

34 Stephan Holzer and Nathan Pinsker. Approximation of distances and shortest paths in the broadcast congest clique. In *OPODIS*, pages 6:1–6:16, 2015.

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

36 Richard M Karp and Avi Wigderson. A fast parallel algorithm for the maximal independent set problem. In *STOC*, pages 266–272, 1984.

37 Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. Local computation: Lower and upper bounds. *J. ACM*, 63(2):17, 2016.

38 Christoph Lenzen. Optimal deterministic routing and sorting on the congested clique. In *PODC*, pages 42–50, 2013.

39 Christoph Lenzen and Roger Wattenhofer. Tight bounds for parallel randomized load balancing: extended abstract. In *STOC*, pages 11–20, 2011.

40 Nathan Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992.

41 Zvi Lotker, Elan Pavlov, Boaz Patt-Shamir, and David Peleg. MST construction in $O(\log \log n)$ communication rounds. In *SPAA*, pages 94–100, 2003.

42 Michael Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM journal on computing*, 15(4):1036–1053, 1986.

43 Michael Luby. Removing randomness in parallel computation without a processor penalty. *Journal of Computer and System Sciences*, 47(2):250–286, 1993.

44 Michael Mitzenmacher and Eli Upfal. *Probability and computing - randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.

45 Rajeev Motwani, Joseph Naor, and Moni Naor. The probabilistic method yields deterministic parallel algorithms. *J. Comput. Syst. Sci.*, 49(3):478–516, 1994.

46 Danupon Nanongkai. Distributed approximation algorithms for weighted shortest paths. In *STOC*, pages 565–573, 2014.

47 Moni Naor and Larry J. Stockmeyer. What can be computed locally? *SIAM J. Comput.*, 24(6):1259–1277, 1995.

48 Alessandro Panconesi and Aravind Srinivasan. Improved distributed algorithms for coloring and network decomposition problems. In *STOC*, pages 581–592, 1992.

49 Alessandro Panconesi and Aravind Srinivasan. On the complexity of distributed network decomposition. *J. Algorithms*, 20(2):356–374, 1996.

50 Grammati Pantziou, Paul Spirakis, and Christos Zaroliagis. Fast parallel approximations of the maximum weighted cut problem through derandomization. In *FSTTCS*, pages 20–29, 1989.

51 Boaz Patt-Shamir and Marat Teplitsky. The round complexity of distributed sorting: extended abstract. In *PODC*, pages 249–256, 2011.

52 Seth Pettie. Distributed algorithms for ultrasparse spanners and linear size skeletons. *Distributed Computing*, 22(3):147–166, 2010.

53 Liam Roditty, Mikkel Thorup, and Uri Zwick. Deterministic constructions of approximate distance oracles and spanners. In *ICALP*, pages 261–272, 2005.

54 Anand Srivastav and Lasse Kliemann. Parallel algorithms via the probabilistic method. In *Handbook of Parallel Computing - Models, Algorithms and Applications.* Chapman and Hall/CRC, 2007.

55 Jukka Suomela. Survey of local algorithms. *ACM Comput. Surv.*, 45(2):24, 2013.

56 Salil P. Vadhan. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 7(1-3):1–336, 2012.