Graph Sparsification for Derandomizing Massively Parallel Computation with Low Space[*]

ARTUR CZUMAJ, University of Warwick
PETER DAVIES, IST Austria
MERAV PARTER, Weizmann Institute of Science

The Massively Parallel Computation (MPC) model is an emerging model which distills core aspects of distributed and parallel computation, developed as a tool to solve combinatorial (typically graph) problems in systems of many machines with limited space.

Recent work has focused on the regime in which machines have sublinear (in n, the number of nodes in the input graph) space, with randomized algorithms presented for the fundamental problems of Maximal Matching and Maximal Independent Set. However, there have been no prior corresponding *deterministic* algorithms.

A major challenge underlying the sublinear space setting is that the local space of each machine might be too small to store all edges incident to a single node. This poses a considerable obstacle compared to classical models in which each node is assumed to know and have easy access to its incident edges. To overcome this barrier we introduce a new *graph sparsification* technique that *deterministically* computes a low-degree subgraph, with the additional property that solving the problem on this subgraph provides significant progress towards solving the problem for the original input graph.

Using this framework to derandomize the well-known algorithm of Luby [SICOMP'86], we obtain $O(\log\Delta + \log\log n)$-round *deterministic* MPC algorithms for solving the problems of *Maximal Matching* and *Maximal Independent Set* with $O(n^{\varepsilon})$ space on each machine for any constant $\varepsilon > 0$. These algorithms also run in $O(\log\Delta)$ rounds in the closely related model of CONGESTED CLIQUE, improving upon the state-of-the-art bound of $O(\log^2\Delta)$ rounds by Censor-Hillel et al. [DISC'17].

CCS Concepts: • **Computing methodologies → Distributed algorithms**; • **Mathematics of computing → Graph algorithms**; • **Theory of computation → Pseudorandomness and derandomization**.

Additional Key Words and Phrases: Massively Parallel Computation, Sparsification, Derandomization, Maximal Independent Set, Maximal Matching

ACM Reference Format:
Artur Czumaj, Peter Davies, and Merav Parter. 2021. Graph Sparsification for Derandomizing Massively Parallel Computation with Low Space. *ACM Trans. Algor.* 1, 1 (May 2021), 28 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

[*]A preliminary version of this paper appeared in *Proceedings of the 32nd Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 175–185, virtual event, USA, July 15–17, 2020.

Authors' addresses: A. Czumaj, a.czumaj@warwick.ac.uk, Computer Science Department and Centre for Discrete Mathematics and its Applications (DIMAP), University of Warwick, Coventry CV4 7AL, UK; P. Davies, peter.davies@ist.ac.at, IST Austria, 3400 Klosterneuburg, Austria; M. Parter, merav.parter@weizmann.ac.il, Weizmann Institute of Science, Rehovot 7610001, Israel.

1 INTRODUCTION

The last few years have seen an increasing interest in the design of parallel algorithms. This has been largely caused by the successes of a number of massively parallel computation frameworks, such as MapReduce [19, 20], Hadoop [53], Dryad [35], or Spark [54], which resulted in the need of active research for understanding the computational power of such systems. The *Massively Parallel Computation (MPC)* model, first introduced by Karloff et al. [37] has become the standard theoretical model of algorithmic study, as it provides a clean abstraction of these frameworks.

The MPC model shares many similarities to earlier models of parallel computation, for example with the PRAM model; indeed, it was quickly observed that it is easy to simulate a single step of PRAM in a constant number of rounds on MPC [31, 37], implying that a vast body of work on PRAM algorithms naturally translates to the MPC model. However, the fact that local computation in the MPC model is essentially "free" enabled it to capture a more "coarse-grained" and meaningful aspect of parallelism. Recent works have brought a number of new algorithms for fundamental graph combinatorial and optimization problems that demonstrated that in many situations the MPC model can be significantly more powerful than PRAM. And so, for example, in a sequence of papers we have seen that the fundamental problems of connectivity (see, e.g., [3, 6]), matching, maximal independent set, vertex cover, coloring, etc (see, e.g., [4, 5, 9, 11–13, 16, 18, 23, 28, 40, 41]) can be solved in the MPC model significantly faster than on PRAM. However, the common feature of most of these results is that they were relying on randomized algorithms, and very little research has been done to study deterministic algorithms.

The main theme of this paper is to *explore the power of the MPC model in the context of deterministic algorithms*. In particular, we want to understand whether the MPC model allows faster deterministic algorithms than in PRAM-like models, in a similar way as it has demonstrated to do in the setting of randomized computation.

We consider two corner-stone problems of local computation: *maximal matching* and *maximal independent set (MIS)*. These problems are arguably among the most fundamental graph problems in parallel and distributed computing with numerous applications. The study of these problems can be traced back to PRAM algorithms of the 1980s [1, 36, 38, 45] and they have been considered as benchmark problems in various computational models since then. In particular, these problems have been central in our understanding of derandomization techniques. Luby [45], and independently Alon et al. [1], have been the first to present a generic transformation of parallel algorithms for maximal matching and MIS, to obtain efficient deterministic algorithms for these problems in the PRAM model. For example, Luby [45] showed that his randomized MIS $O(\log n)$-time algorithm can be derandomized on PRAM in $O(\log^3 n \log\log n)$ time. The bound was later improved to $O(\log^3 n)$ time [29], $O(\log^{2.5} n)$ time [32], and then $\widetilde{O}(\log^2 n)$ time [34].

The MPC model. The *Massively Parallel Computation (MPC)* model was first introduced by Karloff et al. [37] and later refined in [2, 8, 31]. In the MPC model, there are M machines and each of them has S words of space. Initially, each machine receives its share of the input. In our case, the input is a collection V of nodes and E of edges and each machine receives approximately $\frac{n+m}{M}$ of them (divided arbitrarily), where $|V| = n$ and $|E| = m$.

The computation proceeds in synchronous *rounds* in which each machine performs some local computation on its data without communicating with other machines. Some works in the literature define the model to require this computation to be polynomial in n, while others make no restriction on it; in any case, our algorithms will require only polynomial computation.

At the end of each round, machines exchange messages. Each message is sent only to a single machine specified by the machine that is sending the message. All messages sent and received by each machine in each round have to fit into the machine's local space. Hence, their total length is bounded by S. This, in particular, implies that the total communication is bounded by $M \cdot S$ in each round. The messages are processed by recipients in the next round. At the end of the computation, machines collectively output the solution. The data output by each machine has to fit in its local space. Hence again, each machine can output at most S words.

Local space, range of values for S and M, and fully scalable algorithms. If the input is of size N, one usually wants S to be sublinear in N, and the total space across all the machines to be at least N (in order for the input to fit onto the machines) and ideally not much larger. Formally, one usually considers $S = \Theta(N^{\varepsilon})$, for some $\varepsilon > 0$. Optimally, one would want to design *fully scalable algorithms*, which work for *any* positive value of ε (see, e.g., [3, 28, 46]), though most of the earlier works have been focusing on graph algorithms whose space S has been close to the number of edges of the graph (see, e.g., [40], where $S = \Theta\left(n^{1+\varepsilon}\right)$), or close to the number of nodes of the graph (see, e.g., [5, 18, 23], where $S = \widetilde{\Theta}(n)$).

In this paper, the focus is on graph algorithms. If n is the number of nodes in the graph, the input size can be as large as $\Theta\left(n^2\right)$. Our deterministic parallel algorithms are fully scalable, i.e., for any constant $\varepsilon > 0$ require $S = \Theta(n^{\varepsilon})$ space per machine.

Known bounds. For many graph problems, including MIS and maximal matching, fully scalable *randomized* $O(\log n)$ round $n^{\Omega(1)}$ space MPC algorithms can be achieved by simulating PRAM algorithms [1, 36, 45]. These bounds (still in the randomized case) have been improved only very recently and only in some settings. For fully scalable algorithms, we know only of a *randomized* algorithm due to Ghaffari and Uitto [28] working in $\widetilde{O}(\sqrt{\log \Delta})$ rounds for maximal matching and MIS, where Δ is the maximum degree. For the special case of bounded arboricity graphs, Ghaffari et al. [24] give an $O(\log\log n)$-round algorithm for the problems. Better bounds are known for maximal matching algorithms using significantly more space: Lattanzi et al. [40] gave an $O(1/\varepsilon)$ rounds randomized algorithm using $O(n^{1+\varepsilon})$ space per machine, and Behnezhad et al. [13] presented an $O(\log\log \Delta + \log\log\log n)$-round randomized algorithm in $n/2^{\Omega(\sqrt{\log n})}$ space. Further, Ghaffari et al. [27] gave conditional evidence that no $o(\log\log n)$ round fully scalable MPC algorithm can find a maximal matching, or MIS.

Unfortunately, much less is known about *deterministic* MPC algorithms for maximal matching and MIS. Except some parts of the early work in [31] (cf. Lemma 4), we are not aware of any previous algorithms designed specifically for the MPC model. One can use a simulation of PRAM algorithms to obtain fully scalable deterministic algorithms for maximal matching and MIS on MPC, and their number of rounds would be asymptotically the same; to our knowledge, the fastest deterministic PRAM algorithms require $O(\log^{2.5} n)$ [32] rounds for maximal matching, and $\widetilde{O}(\log^2 n)$ rounds for MIS [34]. If one can use linear space per machine, $S = O(n)$, then the recent deterministic CONGESTED CLIQUE algorithms for MIS by Censor-Hillel et al. [15], directly give an $O(\log n \log \Delta)$-round deterministic MPC algorithm for MIS. In concurrent work to ours, Czumaj et al. [17] give a constant-round deterministic CONGESTED CLIQUE algorithm for $(\Delta + 1)$-list coloring, and an $O(\log \Delta + \log\log n)$-round low-space MPC algorithm for the same problem which uses a reduction to our MIS algorithm here.

There have been some related works on derandomization in the LOCAL, CONGEST, and CONGESTED CLIQUE distributed models (cf. [7, 17, 21, 25, 26, 33, 47, 48]). The MPC setting in the low space regime brings along crucial challenges that distinguishes it from the previous computational models in which derandomization has been studied. The inability of a machine to view the entire neighborhood of a node requires novel derandomization paradigms, which is why we developed deterministic graph sparsification. We note that graph sparsification has been shown to be useful before in the context of low-space *randomized* MPC algorithms (e.g., [28]).

1.1 New results

We demonstrate the power of the deterministic algorithms in the MPC model on the example of two fundamental optimization problems: finding a maximal matching and finding an MIS.

Theorem 1. *For any constant $\varepsilon > 0$, **maximal matching** and **MIS** can be found deterministically in the MPC model in $O(\log \Delta + \log \log n)$ rounds, using $O(n^{\varepsilon})$ space per machine and $O(m + n^{1+\varepsilon})$ total space.*

While the additive $O(\log \log n)$ term in the bound in Theorem 1 looks undesirable, it is most likely necessary. Indeed, as mentioned earlier, Ghaffari et al. [27] provided an $\Omega(\log \log n)$ conditional hardness result for maximal matching and MIS, even for *randomized* fully scalable MPC algorithms. They proved that unless there is an $o(\log n)$-round (randomized) MPC algorithm for connectivity with local space $S = n^{\varepsilon}$ for a constant $0 < \varepsilon < 1$ and poly(n) global space (see [50] for strong arguments about the hardness of that problem), there is no component-stable[1] randomized MPC algorithm with local space $S = n^{\varepsilon}$ and poly(n) global space that computes a maximal matching or an MIS in $o(\log \log n)$ rounds (see [27] for a more precise claim and explanations of the notation). For maximal matching, the lower bound holds even on trees. We note, however, that our algorithms are *not* component-stable, since they involve global agreement on hash functions.

1.2 Our approach

Our approach to the problems of MIS and maximal matching in low-space MPC considers two regimes separately: the case where maximum degree Δ is above $n^{\frac{\varepsilon}{2}}$, and the case where it is below. These regimes present different difficulties and require different techniques, which we outline below:

1.2.1 High-degree case. When degree is high, the most prominent limitation is that we cannot store nodes' neighborhoods on single machine. This immediately rules out most standard derandomization techniques, which rely on being able to check neighborhoods to ensure that the solution is progressing as expected. To tackle this problem, we develop our technique of *deterministic graph sparsification*, a method of deterministically reducing the number of edges and/or vertices in a graph while preserving some problem-specific crucial properties.

Deterministic graph sparsification. For the problems of MIS and maximal matching, our eventual aim will be to derandomize a variant of Luby's MIS algorithm [45], which repeatedly finds an independent set, such that removing the independent set along with its neighborhood reduces the number of edges in the graph by a constant factor (a similar approach for maximal matching finds a matching with an analogous property). The property we must preserve during sparsification is therefore that we can still find such a set in the sparsified graph. To do so, we:

(1) provide a *randomized* sampling procedure which requires only bounded independence, and
(2) show that we can check whether the property has been preserved only by summing functions computable by individual low-space machines.

With these two properties, we show that we can apply an implementation of the *method of conditional expectations* to *derandomize* the sampling process, and yield a deterministic means of sparsification. Our general framework can be applied to any problem where the above two points can be satisfied, and so we hope that it might prove useful elsewhere.

We use this procedure to reduce vertices' degrees to the point that we can collect 2-hop neighborhoods onto single machines; at this point we can apply a more standard derandomization to our variant of Luby's algorithm to find the independent set we seek. We can then remove this set,

[1]An algorithm is *component-stable* if the outputs in one connected component are independent of other components [50].

reducing the number of edges in the graph by a constant factor; after $O(\log n)$ steps, we will have removed all edges from the graph and found a maximal independent set. We show that each step can be implemented in only $O(1)$ MPC rounds, yielding an $O(\log n)$ total round complexity. Since in the high-degree regime we have $\Delta = n^{\Omega(1)}$, this is also $O(\log \Delta)$ rounds.

Sections 3 and 4 present this approach in detail for maximal matching and MIS respectively, giving an $O(\log n)$-round MPC algorithm for each.

1.2.2 Low-degree case. When $\Delta \le n^{\frac{\varepsilon}{2}}$, we already have the ability to collect two-hop neighborhoods onto single machines, and therefore we need not perform deterministic graph sparsification: we can apply a more standard approach to derandomize Luby's algorithm. However, this would, as above, give an $O(\log n)$-round algorithm. For the low degree regime, we show that we can do better, obtaining an $O(\log \Delta + \log\log n)$-round algorithm. Since in the high-degree case $O(\log n) = O(\log \Delta)$, this implies an $O(\log \Delta + \log\log n)$ round complexity overall.

The key idea here is to perform deterministic *round compression* on Luby's algorithm. Round compression is a technique which has been used in some *randomized* results in MPC and CONGESTED CLIQUE (e.g., [18, 23, 24], though only the former uses the term), which works by gathering enough information onto machines to simulate multiple steps of a LOCAL or CONGEST algorithm at once.

To perform round compression *deterministically*, we first note that we have the space budget to collect $O\left(\frac{\log n}{\log \Delta}\right)$-hop neighborhoods onto machines, and can do so in $O(\log\log n)$ rounds via graph exponentiation. Next, we prove that a step of Luby's algorithm can be performed correctly using only $O(\log \Delta)$ random bits. Finally, we demonstrate that, using the method of conditional expectations, we can therefore derandomize $O\left(\frac{\log n}{\log \Delta}\right)$ steps of Luby's algorithm at once. The $O(\log n)$ rounds of Luby's algorithm are therefore compressed into only $O(\log \Delta)$ MPC rounds, which, along with the rounds required to collect neighborhoods, gives us our final $O(\log \Delta + \log\log n)$ round complexity. This is detailed in Section 5.

1.3 Implications to CONGESTED CLIQUE

As recently observed (cf. [10]), the MPC model is very closely related to the CONGESTED CLIQUE model from distributed computing. The nowadays classical distributed CONGESTED CLIQUE model (see, e.g., [44, 49]) is a variant of the classical LOCAL model of distributed computation (where in each round network nodes can send through all incident links messages of unrestricted size) with limited communication bandwidth. The distributed system is represented as a complete network (undirected graph) G, where network nodes execute distributed algorithms in synchronous rounds. In any single round, all nodes can perform an unlimited amount of local computation, send a possibly different $O(\log n)$-bit message to each other node, and receive all messages sent by them. We measure the complexity of algorithms by the number of synchronous rounds required. Note that whereas in the LOCAL and CONGEST models [49], only neighboring nodes in G can communicate directly, in the CONGESTED CLIQUE model *every* pair of nodes are allowed to exchange $O(\log n)$ bit in each round.

It is not difficult to see (see, e.g., [10]) that any r-round CONGESTED CLIQUE algorithm can be simulated in $O(r)$ rounds in the MPC model with n machines and $S = O(rn)$. Furthermore, Behnezhad et al. [10] showed that by using the routing scheme of Lenzen [42], MPC algorithms with $S = O(n)$ are adaptable to the CONGESTED CLIQUE model. These results immediately imply that the recent deterministic CONGESTED CLIQUE algorithm due to Censor-Hillel et al. [15] to find MIS in $O(\log n \log \Delta)$ rounds can be extended to be run in the MPC model with $S = \tilde{O}(n)$. (When $\Delta = O(n^{1/3})$, the bound improves to $O(\log \Delta)$.) Notice though, that in contrast to our work, the derandomization algorithm from [15] relies on a derandomization of Ghaffari's MIS algorithm [22], whereas our derandomization is based on Luby's MIS algorithm.

These simulations imply also that our new deterministic MPC algorithms for maximal matching and MIS can be implemented to run in the CONGESTED CLIQUE model using $O(\log \Delta)$ rounds. By combining Theorem 3, for the regime $\Delta = \omega(n^{1/3})$, with the $O(\log \Delta)$-round MIS algorithm of [15] for the regime $\Delta = O(n^{1/3})$, we get an $O(\log \Delta)$-round algorithm for MIS. We further note that, in the $\Delta = O(n^{1/3})$ regime, one can collect 2-hop neighborhoods onto single machines, and thus find a maximal matching by simulating MIS on the line graph of the input graph. So, combining Theorem 7 with the MIS algorithm of [15] yields the following:

Corollary 2. *One can deterministically find MIS and maximal matching in $O(\log \Delta)$ rounds in the CONGESTED CLIQUE model.*

2 PRELIMINARIES

An *independent set* in a graph $G = (V, E)$ is any subset of nodes $\mathcal{I} \subseteq V$ such that no two nodes in $\mathcal{I}$ share an edge. An independent set $\mathcal{I}$ is called a *maximal independent set (MIS)* if it is not possible to add any other node of G to $\mathcal{I}$ and obtain an independent set.

A *matching* of a graph $G = (V, E)$ is any independent subset of edges $M \subseteq E$ (i.e., no two edges in M share an endpoint). A matching M of a graph G is a *maximal matching* if it is not possible to add any other edge of G to M and obtain a matching.

For a node $v \in V$, the neighborhood $N(v)$ is the set of nodes u with $\{u, v\} \in E$; for any $U \subseteq V$, we define $N(U) = \bigcup_{v \in U} N(v)$.

In any graph G we denote the degree of a node v or an edge e (the degree of an edge is the number of other edges sharing an endpoint to it) by $d(v)$ and $d(e)$, respectively. If we have a subset of nodes $U \subseteq V$ or edges $E' \subseteq E$, we will denote $d_U(v)$ to be the number of nodes $u \in U$ such that $\{u, v\} \in E$, and $d_{E'}(v)$ to be the number of edges $e \in E'$ such that $v \in e$. We define the degree $d_{E'}(e)$, of an edge e, to be the number of edges in E' which are adjacent to e. We will use $u \sim v$ to denote adjacency between nodes (or edges), with the underlying graph as a subscript where it is otherwise ambiguous.

Throughout the paper, we use $[\ell]$ to denote the set $\{1, \ldots, \ell\}$.

2.1 Luby's MIS algorithm

Our algorithms will be based on Luby's algorithm [45] for MIS:

Algorithm 1 Luby's MIS algorithm

```
while $|E(G)| > 0$ do
    Each node $v$ generates a random value $z_v \in [0, 1]$
    Node $v$ joins independent set $\mathcal{I}$ iff $z_v < z_u$ for all $u \sim v$
    Add $\mathcal{I}$ to output independent set
    Remove $\mathcal{I}$ and $N(\mathcal{I})$ from the graph $G$
end while
```

The central idea in the analysis is to define an appropriate subset of nodes and show that it is adjacent to a constant fraction of edges in the graph G. Let X be the set of all nodes v that have at least $\frac{d(v)}{3}$ neighbors u with $d(u) \leq d(v)$. Then the following lemma is shown, for example, in Lemma 8.1 of [52].

Lemma 3. *Let X be the set of all nodes v that have at least $\frac{d(v)}{3}$ neighbors u with $d(u) \leq d(v)$. Then $\sum_{v \in X} d(v) \geq \frac{1}{2}|E|$.*

Next, one can then show that every node $v \in X$ has a constant probability of being removed from G, and therefore, in expectation, a constant fraction of G's edges are removed.

This approach gives an $O(\log n)$-round *randomized* algorithm for MIS (with $S = n^{\varepsilon}$). Luby showed, also in [45], that the analysis requires only pairwise independent random choices, and that the algorithm can thus be efficiently *derandomized* (in $O(\log^3 n \log\log n)$ parallel time). However, doing so directly requires many machines ($O(mn^2) = O(n^4)$ in [45]), which would generally be considered a prohibitively high total space bound in MPC.

The approach used in Luby's MIS algorithm can be also extended to find maximal matching, since a maximal matching in G is an MIS in the line graph of G, and in many settings one can simulate Luby's algorithm on this line graph. The resulting algorithm is very similar to Algorithm 1: the difference is that it is now *edges* e that generate values z_e, and join a matching M (an independent set of edges) if $z_e < z_{e'}$ for all adjacent edges $e' \sim e$. The analysis is also similar: one can use the same property about the node set X, and again show that every node $v \in X$ has a constant probability of being removed (when nodes which are matched in M are removed from G).

2.2 Communication in low-space MPC

Low-space MPC is in some ways a restrictive model, and even fully scalable algorithms for routing and communication therein are highly non-trivial. Fortunately, prior work on MapReduce and earlier models of parallel computation have provided black-box tools which will permit all of the types of communication we require for our algorithms. We will not go into the details of those tools, but instead refer the reader to the following summary:

Lemma 4 ([31]). *For any positive constant δ, sorting and computing prefix sums of n numbers can be performed deterministically in MapReduce (and therefore in the MPC model) in a constant number of rounds using $S = n^{\delta}$ space per machine, $O(n)$ total space, and $poly(n)$ local computation.*

The computation of prefix sums here means the following: each machine $m \in [M]$ holds an input value x_m, and outputs $\sum_{i=1}^{m} x_i$.

Proof. The result for sorting follows from applying Theorem 2 of [31] to the BSP sorting algorithm of [30]. The prefix sums result comes from Lemma 1 of [31]. □

This result essentially allows us to perform all of the communication we will need to do in a constant number of rounds. For example, if for each edge $\{u, v\}$ we create two entries (u, v) and (v, u) in memory, and then sort these lexicographically, we can ensure that the neighborhoods of all nodes are stored on contiguous blocks of machines. Then, by computing prefix sums, we can compute sums of values among a node's neighborhood, or indeed over the whole graph. This allows us to, e.g., compute objective functions for the method of conditional expectations (see Section 2.4). When 2-hop neighborhoods fit in the space of a single machine, we can collect them by sorting edges to collect 1-hop neighborhoods onto machines, and then having each such machine send requests for the neighborhoods of all the nodes it stores.

An important point to note is that since Lemma 4 uses $S = n^{\delta}$ for *any* positive constant δ, by setting δ sufficiently smaller than our space parameter ε, we can perform $n^{\Omega(1)}$ *simultaneous* sorting or prefix sum procedures. This will be especially useful to us for efficiently performing the method of conditional expectations.

2.3 Bounded-independence hash functions

Our derandomization is based on a classic recipe: we first show that a randomized process using a *small random seed* produces good results, by using our random seed to select a hash function from a k-wise independent family. Then, we search the space of random seeds to find a good one, using the *method of conditional expectations.*

The families of hash functions we require are specified as follows:

Definition 5. *For $N, L, k \in \mathbb{N}$ such that $k \le N$, a family of functions $\mathcal{H} = \{h : [N] \to [L]\}$ is* k-wise independent *if for all distinct $x_1, \ldots, x_k \in [N]$, the random variables $h(x_1), \ldots, h(x_k)$ are independent and uniformly distributed in $[L]$ when h is chosen uniformly at random from $\mathcal{H}$.*

We will use the following well-known lemma (cf. [51, Corollary 3.34]).

Lemma 6. *For every a, b, k, there is a family of k-wise independent hash functions $\mathcal{H} = \{h : \{0,1\}^a \to \{0,1\}^b\}$ such that choosing a random function from $\mathcal{H}$ takes $k \cdot \max\{a, b\}$ random bits, and evaluating a function from $\mathcal{H}$ takes time $poly(a, b, k)$.*

For all of our purposes (except when extending to low degree inputs, in Section 5), when we require a family of hash functions, we will use a family of c-wise independent hash functions $\mathcal{H} = \{h : [n^3] \to [n^3]\}$, for sufficiently large constant c (we can assume that n^3 is a power of 2 without affecting asymptotic results). We choose n^3 to ensure that our functions have (more than) large enough domain and range to provide the random choices for all nodes and edges in our algorithms. By Lemma 6, a random function can be chosen from $\mathcal{H}$ using $O(\log n)$ random bits (defining the *seeds*).

2.4 Method of conditional expectations

Another central tool in derandomization of algorithms we use is the classical *method of conditional expectations.* In our context, we will show that, over the choice of a random hash function $h \in \mathcal{H}$, the expectation of some objective function (which is a sum of functions q_x calculable by individual machines) is at least some value Q. That is,

$$\mathbf{E}_{h \in \mathcal{H}}\Big[q(h) := \sum_{\text{machines } x} q_x(h)\Big] \ge Q \ .$$

Since, by the probabilistic method, this implies the existence of a hash function $h^* \in \mathcal{H}$ for which $q(h^*) \ge Q$, then our goal is to find one such $h^* \in \mathcal{H}$ in $O(1)$ MPC rounds.

We will find the sought hash function h^* by fixing the $O(\log n)$-bit seed defining it (cf. Lemma 6), by having all machines agree gradually on chunks of $\Theta(\log n)$ bits at a time. That is, we iteratively extend a fixed prefix of the seed until we have fixed the entire seed. For each chunk, and for each i, $1 \le i \le n^{\Omega(1)}$, each machine calculates $\mathbf{E}_{h \in \mathcal{H}}\left[q_x(h) | \Xi_i\right]$, where Ξ_i is the event that the random seed specifying h is prefixed by the current fixed prefix, and then followed by i (since there are $poly(n)$ total seeds, and evaluating each hash function requires only polynomial-time local computation, this calculation of conditional expectations is also polynomial-time). We then sum these values over all machines for each i, using Lemma 4 (recall that we have sufficient space to conduct $n^{\Omega(1)}$ concurrent applications), obtaining $\mathbf{E}_h\left[q(h) | \Xi_i\right]$. By the probabilistic method, at least one of these values is at least Q. We fix i to be such that this is the case, and continue.

After $O(1)$ iterations, we find the entire seed to define $h^* \in \mathcal{H}$ such that $q(h^*) \ge Q$. Since each iteration requires only a constant number of MPC rounds, this process takes only $O(1)$ rounds.

3 MAXIMAL MATCHING IN $O(\log n)$ MPC ROUNDS

In this section we present a deterministic fully scalable $O(\log n)$-rounds MPC algorithm for the maximal matching problem. Later, in Section 5, we will extend this algorithm to obtain a round complexity $O(\log \Delta + \log\log n)$, as promised in Theorem 1; this improves the bound from this section for $\Delta = n^{o(1)}$.

Theorem 7. *For any constant $\varepsilon > 0$, maximal matching can be found deterministically in the MPC model in $O(\log n)$ rounds, using $O(n^{\varepsilon})$ space per machine and $O(m + n^{1+\varepsilon})$ total space.*

The main idea is to derandomize a variant of a maximal matching algorithm due to Luby (cf. Section 2.1), which in $O(\log n)$ rounds finds a maximal matching. In each round of Luby's algorithm one selects some matching M and then removes all nodes in M (and hence all edges adjacent to M). It is easy to see that after sufficiently many rounds the algorithm finds maximal matching. The central feature of the randomized algorithm is that in expectation, in each single round one will remove a constant fraction of the edges, and hence $O(\log n)$ rounds will suffice in expectation. This is achieved in two steps. One first selects an appropriated subset of nodes and show that it is adjacent to a constant fraction of edges in the graph G (cf. Lemma 3). Then, one shows that every node $v \in X$ has a constant probability of being removed from G (by being incident to the matching M found in a given round), and therefore, in expectation, a constant fraction of G's edges are removed.

In order to derandomize such algorithm, we will show that each single round can be implemented deterministically in a constant number of rounds in the MPC model so that the same property will be maintained deterministically: in a constant number of rounds one will remove a constant fraction of the edges, and hence $O(\log n)$ rounds will suffice. This is achieved in three steps:

- select a set of good nodes $B \subseteq X$ which are adjacent to a constant fraction of the edges,
- then sparsify to $\mathcal{E}^*$ the set of edges incident to B to ensure that each node has degree $O(n^{\varepsilon/2})$ in $\mathcal{E}^*$, and hence a single machine can store its entire 2-hop neighborhood, and
- then find a matching $M \subseteq \mathcal{E}^*$ such that removal of all nodes in M (i.e., removal of M and all edges adjacent to M) reduces the number of edges by a constant factor.

This outline is broadly similar to that described in Section 2.1; the main difference is that rather than analyzing the entire set X of good nodes, we instead find a subset $B \subset X$ that is still incident to a (smaller) constant fraction of edges, and also has a property that admits sparsification (which is that its node degrees are all within a small polynomial range). Then we can sparsify the graph to $\mathcal{E}^*$ while ensure that nodes in B remain good. Finally, we can collect 2-hop neighborhoods and perform the step of Luby's algorithm, by finding an appropriate matching M to remove.

Good nodes. We start with a corollary of Lemma 3, which specifies a set of *good nodes* which are nodes with similar degrees that are adjacent to a constant fraction of edges in the graph.

Let δ be a positive constant, with $1/\delta \in \mathbb{N}$ (we will later show that we require $n^{O(\delta)}$ space per machine, and thereby meet our n^{ε} space bound by fixing δ sufficiently smaller than ε). We will proceed in a constant (dependent on δ) number of stages, sparsifying the graph induced by the edges incident to good nodes by derandomizing the sampling of edges with probability $n^{-\delta}$ in each stage. In order for this to work, we want our good nodes to be within a degree range of at most a n^{δ} factor, for their behavior to be similar.

Let us recall (cf. Section 2.1) that X is the set of all nodes v which have at least $\frac{d(v)}{3}$ neighbors u with $d(u) \le d(v)$. Partition nodes into sets C^i, $1 \le i \le 1/\delta$, such that $C^i = \{v : n^{(i-1)\delta} \le d(v) < n^{i\delta}\}$. Let $B^i = C^i \cap X$. The following is a simple corollary of Lemma 3.

Corollary 8. *There is $i \le 1/\delta$, such that $\sum_{v \in B_i} d(v) \ge \frac{\delta}{2}|E|$.*

Proof. By Lemma 3, $\sum_{v \in X} d(v) \geq |E|$. Since the sets $B^1, \dots, B^{1/\delta}$ form a partition of X into $1/\delta$ subsets, at least one of them must contribute a δ-fraction of the sum $\sum_{v \in X} d(v) \geq \frac{1}{2}|E|$. □

From now on, let us fix some i which satisfies Corollary 8. Denote $B := B^i$, and for each node $v \in B$, let $X(v) := \{\{u, v\} \in E : d(u) \leq d(v)\}$. Note that the definition of set X yields $|X(v)| \geq \frac{d(v)}{3}$. Denote $\mathcal{E}_0 = \bigcup_{v \in B} X(v)$. $\mathcal{E}_0$ is the set of edges we will be sub-sampling to eventually find a matching, and B is the set of good nodes which we want to match and remove from the graph, in order to significantly reduce the number of edges.

The outline of our maximal matching algorithm is as follows:

Algorithm 2 Maximal matching algorithm outline

```
while $|E(G)| > 0$ do
    Compute $i$, $B$ and $\mathcal{E}_0$
    Select a set $\mathcal{E}^* \subseteq \mathcal{E}_0$ that induces a low degree subgraph
    Find matching $M \subseteq \mathcal{E}^*$ with $\sum_{\text{matched nodes } v} d(v) = \Omega(|E(G)|)$
    Add $M$ to output matching, remove matched nodes from $G$
end while
```

As long as each iteration reduces the number of edges in G by a constant fraction, we get only $O(\log n)$ iterations to find a maximal matching. We will show that the iterations require a constant number of rounds each, so $O(\log n)$ rounds are required overall.

3.1 Computing i, B, and $\mathcal{E}_0$

As discussed in Section 2.2, a straightforward application of Lemma 4 allows all nodes to determine their degrees, and therefore their membership of sets C^i, in a constant number of rounds. A second application allows nodes to determine whether they are a member of X, and therefore B^i, and also provides nodes $v \in X$ with $X(v)$. Finally, a third application allows the computation of the values $\sum_{v \in B^i} d(v)$ for all i. Upon completing, all nodes know which i yields the highest value for this sum, and that is the value for i which will be fixed for the remainder of the algorithm.

3.2 Deterministically selecting $\mathcal{E}^*$ that induces a low degree subgraph

We will show now how to deterministically, in $O(1)$ stages, select a subset $\mathcal{E}^*$ of $\mathcal{E}_0$ that induces a low degree subgraph, as required in our MPC algorithm. For that, our main goal is to ensure that every node has degree $O(n^{4\delta})$ in $\mathcal{E}^*$ (to guarantee that its 2-hop neighborhood will fit a single MPC machine with $S = O(n^{8\delta})$), and that one can then locally find a matching $M \subseteq \mathcal{E}^*$ that will cover a linear number of edges.

We first consider the easy case when $i \leq 4$, in which case we set directly $\mathcal{E}^* = \mathcal{E}_0$. Notice that in that case, by definitions of X and $B = C^i \cap X$, we have (i) $d_{\mathcal{E}^*}(v) = d_{\mathcal{E}_0}(v) \leq n^{4\delta}$ for all nodes v, and (ii) $|X(v) \cap \mathcal{E}^*| = |X(v)| \geq \frac{d(v)}{3}$ for all nodes $v \in B$, which is what yields the requirements for $\mathcal{E}^*$ (cf. the Invariant below) needed in our analysis in Section 3.3.

Next, for the rest of the analysis, let us assume that $i \geq 5$. We proceed in $i - 4$ stages, starting with $\mathcal{E}_0$ and sparsifying it by sub-sampling a new edge set $\mathcal{E}_j$ in each stage j, $j = 1, 2, \dots, i - 4$. Note that for any node v we have $d_{\mathcal{E}_0}(v) \leq n^{i\delta}$, since nodes in B have maximum degree $n^{i\delta}$ and since v only has adjacent edges in $\mathcal{E}_0$ if $v \in B$ or $\exists u \in B : d(v) \leq d(u)$.

Invariant: In our construction of sets $\mathcal{E}_0, \mathcal{E}_1, \dots, \mathcal{E}_{i-4}$, in order to find a good matching in the resulting sub-sampled graph $\mathcal{E}^*$, we will maintain the following invariant for every j:

(i) for all nodes v: $d_{\mathcal{E}_j}(v) \leq (1+o(1))n^{-j\delta}d_{\mathcal{E}_0}(v) + n^{3\delta}$,
(ii) for all nodes $v \in B$: $|X(v) \cap \mathcal{E}_j| \geq (1-o(1))n^{-j\delta}|X(v)|$.

The intuition behind this invariant is that nodes' degrees decrease roughly as expected in the sub-sampled graph, and nodes $v \in B$ do not lose too many edges to their neighbors in $X(v)$ (to ensure that many of them can be matched in the sub-sampled graph).

One can see that the invariant holds for $j = 0$ trivially, by definition of sets $\mathcal{E}_0$ and B.

Distributing edges and nodes among the machines. In order to implement our scheme in the MPC model, we first allocate the nodes and the edges of the graph among the machines.

- Each node v distributes its adjacent edges in $\mathcal{E}_{j-1}$ across a group of *type A* machines, with $n^{4\delta}$ edges on all but at most one machine (which holds any remaining edges).
- Each node $v \in B$ also distributes its adjacent edges in $X(v) \cap \mathcal{E}_{j-1}$ across a group of *type B* machines in the same fashion.

Type A machines will be used to ensure that the first point of the invariant holds, and type B machines will ensure the second.

In order to sparsify $\mathcal{E}_{j-1}$ to define $\mathcal{E}_j$, we proceed with derandomization of a sub-sampled graph. We will fix a seed specifying a hash function from $\mathcal{H}$ (recall that $\mathcal{H} = \{h : [n^3] \to [n^3]\}$ is a c-independent family for sufficiently large constant c). Each hash function h induces a set $\mathcal{E}_h$ in which each edge in $\mathcal{E}_{j-1}$ is sampled with probability $n^{-\delta}$, by placing $e \in \mathcal{E}_h$ iff $h(e) \leq n^{3-\delta}$.

Good machines. We will call a machine *good* for a hash function $h \in \mathcal{H}$ if the effect of h on the edges it stores looks like it will preserve the invariant. We will then show that if all machines are good for a hash function h, the invariant is indeed preserved.

Formally, consider a machine (of either type) x that receives $\mathcal{E}(x) \subseteq \mathcal{E}_{j-1}$ and let $\xi_x := |\mathcal{E}(x)|$. For hash function $h \in \mathcal{H}$, we call x *good* if $\xi_x n^{-\delta} - n^{0.1\delta}\sqrt{\xi_x} \leq |\mathcal{E}(x) \cap \mathcal{E}_h| \leq \xi_x n^{-\delta} + n^{0.1\delta}\sqrt{\xi_x}$.

Our aim is to use the following concentration bound to show that a machine is good with high probability:

Lemma 9 (Lemma 2.2 of [14]). *Let $c \geq 4$ be an even integer. Let $Z_1, \ldots, Z_t$ be c-wise independent random variables taking values in $[0,1]$, $Z = Z_1 + \cdots + Z_t$ and $\mu = \mathbf{E}[Z]$. Let $\lambda > 0$. Then,*

$$\mathbf{Pr}[|Z-\mu| \geq \lambda] \leq 2\left(\frac{ct}{\lambda^2}\right)^{c/2} .$$

We will take Z to be the sum of the indicator variables $\mathbf{1}_{\{e \in \mathcal{E}_h\}}$ for $e \in \mathcal{E}(x)$ (i.e., $Z = |\mathcal{E}(x) \cap \mathcal{E}_h|$). These indicator variables $\mathbf{1}_{\{e \in \mathcal{E}_h\}}$ are c-wise independent, and each has expectation $n^{-\delta}$. Using that c is a sufficiently large constant, we apply Lemma 9 and get that

$$\mathbf{Pr}\left[|Z-\mu| \geq n^{0.1\delta}\sqrt{\xi_x}\right] \leq 2\left(\frac{c\xi_x}{n^{0.2\delta}\xi_x}\right)^{c/2} = 2\left(cn^{-0.2\delta}\right)^{c/2} \leq n^{-5} .$$

This means that with high probability, $\xi_x n^{-\delta} - n^{0.1\delta}\sqrt{\xi_x} \leq |\mathcal{E}(x) \cap \mathcal{E}_h| \leq \xi_x n^{-\delta} + n^{0.1\delta}\sqrt{\xi_x}$, and x is good.

By the method of conditional expectations, as described in Section 2.4 using objective function $q_x(h) = \mathbf{1}_{x \text{ is good for } h}$, we can find a function h which makes all machines good, in a constant number of rounds. We then set $\mathcal{E}_j = \mathcal{E}_h$.

3.2.1 *Properties of $\mathcal{E}_j$: satisfying the invariant.*

Having fixed a sub-sampled graph for the stage, we need to show that since all machines were good, we satisfy our invariant for the stage.

Lemma 10 (Invariant (i)). *All nodes v satisfy*

$$d_{\mathcal{E}_j}(v) \le (1+o(1))n^{-j\delta} d_{\mathcal{E}_0}(v) + n^{3\delta} \ .$$

Proof. Node v's adjacent edges in $\mathcal{E}_{j-1}$ were divided among $\lfloor \frac{d_{\mathcal{E}_{j-1}}(v)}{n^{3\delta}} \rfloor$ type A machines containing $n^{4\delta}$ neighbors, and one machine containing the remaining $d_{\mathcal{E}_{j-1}}(v) - n^{4\delta}\lfloor \frac{d_{\mathcal{E}_{j-1}}(v)}{n^{4\delta}} \rfloor \le d_{\mathcal{E}_{j-1}}(v)$ neighbors. Therefore we obtain:

$$\begin{aligned} d_{\mathcal{E}_j}(v) &\le \sum_{\substack{v\text{'s type } A \\ \text{machines } x}} v_x n^{-\delta} + n^{0.1\delta}\sqrt{v_x} \\ &\le n^{-\delta} d_{\mathcal{E}_{j-1}}(v) + \left\lfloor \frac{d_{\mathcal{E}_{j-1}}(v)}{n^{4\delta}} \right\rfloor n^{0.1\delta}\sqrt{n^{4\delta}} + n^{0.1\delta}\sqrt{d_{\mathcal{E}_{j-1}}(v)} \\ &\le n^{-\delta} d_{\mathcal{E}_{j-1}}(v) + \frac{n^{0.1\delta}}{n^{2\delta}} d_{\mathcal{E}_{j-1}}(v) + n^{0.1\delta}\sqrt{d_{\mathcal{E}_{j-1}}(v)} \ . \end{aligned}$$

If $d_{\mathcal{E}_{j-1}}(v) \ge n^{3\delta}$, we have

$$\begin{aligned} d_{\mathcal{E}_j}(v) &\le n^{-\delta} d_{\mathcal{E}_{j-1}}(v) + \frac{n^{0.1\delta}}{n^{2\delta}} d_{\mathcal{E}_{j-1}}(v) + n^{0.1\delta}\sqrt{d_{\mathcal{E}_{j-1}}(v)} \\ &\le n^{-\delta} d_{\mathcal{E}_{j-1}}(v) + n^{-1.9\delta} d_{\mathcal{E}_{j-1}}(v) + n^{-1.4\delta} d_{\mathcal{E}_{j-1}}(v) \\ &= (1+o(1))n^{-\delta} d_{\mathcal{E}_{j-1}}(v) \\ &\le (1+o(1))n^{-j\delta} d_{\mathcal{E}_0}(v) \ . \end{aligned}$$

Otherwise, $d_{\mathcal{E}_j}(v) \le d_{\mathcal{E}_{j-1}}(v) \le n^{3\delta}$. In either case, we satisfy the invariant for stage j. □

Lemma 11 (Invariant (ii)). *All nodes $v \in B$ satisfy*

$$|X(v) \cap \mathcal{E}_j| \ge (1-o(1))n^{-j\delta}|X(v)| \ .$$

Proof. The edges in $X(v) \cap \mathcal{E}_{j-1}$ were divided among $\lfloor \frac{|X(v)\cap\mathcal{E}_{j-1}|}{n^{3\delta}} \rfloor$ type B machines containing $n^{4\delta}$ neighbors, and one machine containing the remaining $|X(v) \cap \mathcal{E}_{j-1}| - n^{4\delta}\lfloor \frac{|X(v)\cap\mathcal{E}_{j-1}|}{n^{4\delta}} \rfloor \le |X(v) \cap \mathcal{E}_{j-1}|$ neighbors.

$$\begin{aligned} |X(v)\cap\mathcal{E}_j| &\ge \sum_{\substack{v\text{'s type } B \\ \text{machines } x}} v_x n^{-\delta} - n^{0.1\delta}\sqrt{v_x} \\ &= n^{-\delta}|X(v)\cap\mathcal{E}_{j-1}| - \left\lfloor \frac{|X(v)\cap\mathcal{E}_{j-1}|}{n^{4\delta}} \right\rfloor n^{0.1\delta}\sqrt{n^{4\delta}} - n^{0.1\delta}\sqrt{|X(v)\cap\mathcal{E}_{j-1}|} \\ &\ge n^{-\delta}|X(v)\cap\mathcal{E}_{j-1}| - \frac{n^{0.1\delta}}{n^{2\delta}}|X(v)\cap\mathcal{E}_{j-1}| - n^{0.1\delta}\sqrt{|X(v)\cap\mathcal{E}_{j-1}|} \ . \end{aligned}$$

Since, by the invariant for stage $j-1$, we have $|X(v)\cap\mathcal{E}_{j-1}| \ge n^{4\delta}$, we can continue the lower bound from above to obtain,

$$\begin{aligned} |X(v)\cap\mathcal{E}_j| &\ge n^{-\delta}|X(v)\cap\mathcal{E}_{j-1}| - \frac{n^{0.1\delta}}{n^{2\delta}}|X(v)\cap\mathcal{E}_{j-1}| - n^{0.1\delta}\sqrt{|X(v)\cap\mathcal{E}_{j-1}|} \\ &\ge n^{-\delta}|X(v)\cap\mathcal{E}_{j-1}| - n^{3\delta} \\ &= (1-o(1))n^{-\delta}|X(v)\cap\mathcal{E}_{j-1}| \\ &\ge (1-o(1))n^{-j\delta}|X(v)| \ , \end{aligned}$$

where the last inequality follows directly from Invariant (ii). □

We have proven that our invariant is preserved in every stage, and therefore holds in our final sub-sampled edge set $\mathcal{E}^* := \mathcal{E}_{i-4}$.

3.3 Finding a matching $M \subseteq \mathcal{E}^*$

The construction in Section 3.2 ensures that either $i \leq 4$, in which case $\mathcal{E}^* = \mathcal{E}_0$, or $i \geq 5$ and after $i-4$ stages, we now have a set of edges $\mathcal{E}^* = \mathcal{E}_{i-4}$ with the following properties:

(i) all nodes v have

$$d_{\mathcal{E}^*}(v) \leq (1+o(1))n^{(4-i)\delta} d_{\mathcal{E}_0}(v) + n^{3\delta} \leq 2n^{4\delta} \quad ,$$

(ii) all nodes $v \in B$ have

$$|X(v) \cap \mathcal{E}^*| \geq (1-o(1))n^{(4-i)\delta}|X(v)| \quad .$$

We can show a property analogous to Lemma 3 in $\mathcal{E}^*$[2].

Lemma 12. *Every node $v \in B$ either satisfies $\sum_{\{u,v\}\in\mathcal{E}^*} \frac{1}{d_{\mathcal{E}^*}(\{u,v\})} \geq \frac{1}{27}$, or is incident to an edge $\{u,v\} \in \mathcal{E}^*$ whose degree in $\mathcal{E}^*$ is 0.*

Proof. Fix $v \in B$. We begin with the case $i \geq 5$, and then proceed to the simpler case $i \leq 4$.

Case $i \geq 5$: If v does not have an incident edge $\{u,v\} \in \mathcal{E}^*$ whose degree in $\mathcal{E}^*$ is 0, then:

$$\begin{aligned}
\sum_{\{u,v\}\in\mathcal{E}^*} \frac{1}{d_{\mathcal{E}^*}(\{u,v\})} &\geq \sum_{\{u,v\}\in\mathcal{E}^*\cap X(v)} \frac{1}{d_{\mathcal{E}^*}(\{u,v\})} \geq \sum_{\{u,v\}\in\mathcal{E}^*\cap X(v)} \frac{1}{d_{\mathcal{E}^*}(u) + d_{\mathcal{E}^*}(v)} \\
&\geq \sum_{\{u,v\}\in\mathcal{E}^*\cap X(v)} \frac{1}{2n^{(4-i)\delta}(d(u)+d(v)) + 2n^{3\delta}} \\
&\geq \sum_{\{u,v\}\in\mathcal{E}^*\cap X(v)} \frac{1}{2n^{(4-i)\delta}(2d(v) + n^{(i-1)\delta})} \\
&= \frac{|\mathcal{E}^* \cap X(v)|}{2n^{(4-i)\delta}(2d(v)+n^{(i-1)\delta})} \geq \frac{n^{(4-i)\delta}|X(v)|}{3n^{(4-i)\delta}(2d(v)+n^{(i-1)\delta})} \\
&\geq \frac{|X(v)|}{9d(v)} \geq \frac{1}{27} \quad .
\end{aligned}$$

Here, the 3rd inequality follows from Invariant *(i)*, the 4th inequality follows from the fact that $\{u,v\} \in X(v)$ yields $d(u) \leq d(v)$, the 5th inequality follows from Invariant *(ii)*, the 6th inequality follows from the fact that $v \in C^i$ and hence $d(v) \geq n^{(i-1)\delta}$, and the last inequality follows from the fact that $v \in B$ yields $v \in X$ and hence $|X(v)| \geq \frac{d(v)}{3}$.

Case $1 \leq i \leq 4$: The proof for the case $1 \leq i \leq 4$ uses the fact $\mathcal{E}^* = \mathcal{E}_0$, and hence *(i)* $d_{\mathcal{E}^*}(v) = d_{\mathcal{E}_0}(v)$ for all nodes v, and *(ii)* $|X(v) \cap \mathcal{E}^*| = |X(v)| \geq \frac{d(v)}{3}$ for all nodes $v \in B$. For a fixed $v \in B$, if

[2]Recall (cf. Section 2) that a degree of an edge e in $\mathcal{E}^*$, $d_{\mathcal{E}^*}(e)$, is the number of edges in $\mathcal{E}^*$adjacent to e.

v does not have an incident edge $\{u, v\} \in \mathcal{E}^*$ whose degree in $\mathcal{E}^*$ is 0, then:

$$\begin{aligned}
\sum_{\{u,v\}\in\mathcal{E}^*} \frac{1}{d_{\mathcal{E}^*}(\{u,v\})} &= \sum_{\{u,v\}\in\mathcal{E}^*\cap X(v)} \frac{1}{d_{\mathcal{E}^*}(\{u,v\})} \geq \sum_{\{u,v\}\in\mathcal{E}^*\cap X(v)} \frac{1}{d_{\mathcal{E}^*}(u) + d_{\mathcal{E}^*}(v)} \\
&\geq \sum_{\{u,v\}\in\mathcal{E}^*\cap X(v)} \frac{1}{d(u)+d(v)} \geq \sum_{\{u,v\}\in\mathcal{E}^*\cap X(v)} \frac{1}{2d(v)} \\
&= \sum_{\{u,v\}\in X(v)} \frac{1}{2d(v)} = \frac{|X(v)|}{2d(v)} \geq \frac{1}{6} \enspace .
\end{aligned}$$

Here, in the third inequality we use the fact that $\{u, v\} \in X(v)$ yields $d(u) \leq d(v)$, and the last inequality follows from the fact that $v \in B$ yields $v \in X$ and hence $|X(v)| \geq \frac{d(v)}{3}$. □

Now we are ready to present our deterministic MPC algorithm that for a given subset of edges $\mathcal{E}^*$ satisfying the invariant, in $O(1)$ rounds constructs a matching $M \subseteq \mathcal{E}^*$ such that the removal of M and all edges adjacent to M removes $\Omega(\delta|E|)$ edges from the graph.

First, each node $v \in B$ is assigned a machine x_v which gathers its 2-hop neighborhood in $\mathcal{E}^*$. Since for every node u we have $d_{\mathcal{E}^*}(u) \leq 2n^{4\delta}$ by Invariant *(i)* (or by the definition of B and $\mathcal{E}_0 = \mathcal{E}^*$, when $1 \leq i \leq 4$), this requires at most $2n^{4\delta} \cdot 2n^{4\delta} = O(n^{8\delta})$ space per machine. Altogether, since $|B| \leq n$, this is $O(n^{1+8\delta})$ total space.

We will fix a seed specifying a hash function h from $\mathcal{H}$. This hash function h will be used to map each edge e in $\mathcal{E}^*$ to a value $z_e \in [n^3]$. Then, e joins the *candidate matching* $\mathcal{E}_h$ iff $z_e < z_{e'}$ for all $e' \sim e$. Further, since each node $v \in B$ is assigned a machine which gathers its 2-hop neighborhood in $\mathcal{E}^*$, in a single MPC round, every node $v \in B$ can determine its degree $d_{\mathcal{E}_h}(v)$.

Clearly $\mathcal{E}_h$ is indeed a matching for every $h \in \mathcal{H}$, but we require that removing $\mathcal{E}_h \cup N(\mathcal{E}_h)$ from the graph reduces the number of edges by a constant fraction. We will show that $|\mathcal{E}_h \cup N(\mathcal{E}_h)| = \Omega(\delta|E|)$ in expectation, and therefore by the method of conditional expectations (cf. Section 2.4) we will be able to find a seed $h^* \in \mathcal{H}$ for which $|\mathcal{E}_{h^*} \cup N(\mathcal{E}_{h^*})| = \Omega(\delta|E|)$.

LEMMA 13. *For any machine x_v holding the 2-hop neighborhood of v in $\mathcal{E}^*$, the probability that $d_{\mathcal{E}_h}(v) = 1$, for a random hash function $h \in \mathcal{H}$, is at least $\frac{1}{109}$.*

PROOF. If v has an adjacent edge e with degree 0, then e will join $\mathcal{E}_h$ and we are done.

Otherwise, for any edge $\{u, v\} \in \mathcal{E}^*$ it holds that,

$$\frac{1}{2d_{\mathcal{E}^*}(\{u,v\})} \geq \mathbf{Pr}\left[z_{\{u,v\}} < \frac{n^3}{2d_{\mathcal{E}^*}(\{u,v\})}\right] \geq \frac{1}{2d_{\mathcal{E}^*}(\{u,v\})} - \frac{1}{n^3} \enspace .$$

Conditioned on $z_{\{u,v\}} < \frac{n^3}{2d_{\mathcal{E}^*}(\{u,v\})}$, the probability that $\{u,v\} \in \mathcal{E}_h$ is at least

$$\begin{aligned}
\mathbf{Pr}\left[\{u,v\} \in \mathcal{E}_h \middle| z_{\{u,v\}} < \tfrac{n^3}{2d_{\mathcal{E}^*}(\{u,v\})}\right] &= \mathbf{Pr}\left[\forall_{e\in\mathcal{E}^*\sim\{u,v\}}\ z_{\{u,v\}} < z_e \middle| z_{\{u,v\}} < \tfrac{n^3}{2d_{\mathcal{E}^*}(\{u,v\})}\right]\\
&= 1 - \mathbf{Pr}\left[\exists_{e\in\mathcal{E}^*\sim\{u,v\}}\ z_e \le z_{\{u,v\}} \middle| z_{\{u,v\}} < \tfrac{n^3}{2d_{\mathcal{E}^*}(\{u,v\})}\right]\\
&\ge 1 - \sum_{e\in\mathcal{E}^*\sim\{u,v\}} \mathbf{Pr}\left[z_e \le z_{\{u,v\}} \middle| z_{\{u,v\}} < \tfrac{n^3}{2d_{\mathcal{E}^*}(\{u,v\})}\right]\\
&\ge 1 - \sum_{e\in\mathcal{E}^*\sim\{u,v\}} \mathbf{Pr}\left[z_e < \tfrac{n^3}{2d_{\mathcal{E}^*}(\{u,v\})}\right]\\
&\ge 1 - d_{\mathcal{E}^*}(\{u,v\}) \cdot \frac{1}{2d_{\mathcal{E}^*}(\{u,v\})}\\
&= \frac{1}{2}\ ,
\end{aligned}$$

by pairwise independence of the family of hash functions $\mathcal{H}$.

Therefore, the probability that $d_{\mathcal{E}_h}(v) = 1$ is at least:

$$\begin{aligned}
\mathbf{Pr}\left[d_{\mathcal{E}_h}(v) = 1\right] &= \sum_{u\sim_{\mathcal{E}^*}v} \mathbf{Pr}\left[\{u,v\} \in \mathcal{E}_h\right] \ge \sum_{u\sim_{\mathcal{E}^*}v} \mathbf{Pr}\left[\{u,v\} \in \mathcal{E}_h \wedge z_{\{u,v\}} < \tfrac{n^3}{2d_{\mathcal{E}^*}(\{u,v\})}\right]\\
&\ge \sum_{u\sim_{\mathcal{E}^*}v} \mathbf{Pr}\left[\{u,v\} \in \mathcal{E}_h \middle| z_{\{u,v\}} < \tfrac{n^3}{2d_{\mathcal{E}^*}(\{u,v\})}\right] \cdot \mathbf{Pr}\left[z_{\{u,v\}} < \tfrac{n^3}{2d_{\mathcal{E}^*}(\{u,v\})}\right]\\
&\ge \sum_{u\sim_{\mathcal{E}^*}v} \frac{1}{2} \cdot \left(\frac{1}{2d_{\mathcal{E}^*}(\{u,v\})} - \frac{1}{n^3}\right) \ge \frac{1}{4} \cdot \sum_{u\sim_{\mathcal{E}^*}v} \frac{1}{d_{\mathcal{E}^*}(\{u,v\})} - \frac{1}{2n^2} \ge \frac{1}{108} - \frac{1}{2n^2}\ ,
\end{aligned}$$

where the first identity follows since the events are mutually exclusive, and the last one follows by Lemma 12. The claim now follows from $\frac{1}{108} - \frac{1}{2n^2} \ge \frac{1}{109}$ for large enough n. □

We will denote $N_h := \{v \in B : d_{\mathcal{E}_h}(v) = 1\}$, i.e., the set of nodes in the matching induced by hash function h. We want to study the number of edges incident to N_h. By Lemma 13,

$$\mathbf{E}\left[\sum_{v\in N_h} d(v)\right] \ge \sum_{v\in B} d(v) \cdot \mathbf{Pr}\left[v \in N_h\right] \ge \frac{1}{109}\sum_{v\in B} d(v) \ge \frac{\delta|E|}{218}\ .$$

By the method of conditional expectations (cf. Section 2.4), using objective function $q_{x_v}(h) = d(v)\mathbf{1}_{v\in N_h}$, we can select a hash function h with $\sum_{v\in N_h} d(v) \ge \frac{1}{218}\delta|E|$. We then add the matching $M := \mathcal{E}_h$ to our output, and remove matched nodes from the graph. In doing so, we remove at least $\frac{\delta|E|}{436}$ edges from the graph.

3.4 Completing the proof of Theorem 7: finding a maximal matching

Now we are ready to complete the proof of Theorem 7, that a maximal matching can be found deterministically in the MPC model in $O(\log n)$ rounds, with $S = O(n^\varepsilon)$, and $O(m + n^{1+\varepsilon})$ total space.

Our algorithm returns a maximal matching in $\log_{\frac{1}{1-\delta/536}} |E| = O(\log n)$ iterations, each requiring $O(1)$ MPC rounds. The space required is dominated by storing the input graph G ($O(m)$ total space) and collecting 2-hop neighborhoods when finding an matching ($O(n^{8\delta})$ space per machine, $O(n^{1+8\delta})$ total space). Setting $\delta = \frac{\varepsilon}{8}$ allows us to conclude Theorem 7, that for any constant $\varepsilon > 0$, maximal matching can be found in the MPC model in $O(\log n)$ rounds, using $O(n^\varepsilon)$ space per machine and $O(m + n^{1+\varepsilon})$ total space. □

4 MAXIMAL INDEPENDENT SET IN $O(\log n)$ MPC ROUNDS

In this section we modify the approach from Section 3 for the maximal independent set problem and prove the following.

Theorem 14. *For any constant $\varepsilon > 0$, MIS can be found deterministically in MPC in $O(\log n)$ rounds, using $O(n^{\varepsilon})$ space per machine and $O(m + n^{1+\varepsilon})$ total space.*

Later, in Section 5, we will extend this algorithm to obtain a round complexity $O(\log \Delta + \log\log n)$; this will improve the bound from Theorem 14 when $\Delta = n^{o(1)}$.

4.1 Outline

The approach to find an MIS in $O(\log n)$ MPC rounds is similar to the algorithm for maximal matching. However, the main difference is that for MIS, instead of the edges, as for the matching, we have to collect the nodes, which happen to require some changes in our analysis and makes some of its part slightly more complex.

Let $\mathcal{A}$ be the set of all nodes v such that $\sum_{u\sim v} \frac{1}{d(u)} \geq \frac{1}{3}$. Our analysis again relies on a corollary to the analysis of Luby's algorithm (cf. Lemma 3) that follows from the fact that $X \subseteq \mathcal{A}$.

Corollary 15. $\sum_{v\in\mathcal{A}} d(v) \geq \frac{1}{2}|E|$.

Proof. Nodes v in X (cf. Lemma 3) satisfy $\sum_{u\sim v} \frac{1}{d(u)} \geq \frac{1}{3}$. □

We will again partition nodes into classes of similar degree. Let δ be an arbitrarily small constant and assume $1/\delta \in \mathbb{N}$. As in Section 3, partition nodes into sets C^i, $1 \leq i \leq 1/\delta$, with $C^i = \{v : n^{(i-1)\delta} \leq d(v) < n^{i\delta}\}$. For any $1 \leq i \leq 1/\delta$, let $\mathcal{B}_i$ be the set of all nodes v satisfying $\sum_{u\in C^i : u\sim v} \frac{1}{d(u)} \geq \frac{\delta}{3}$. We can easily prove the following.

Corollary 16. *There is $i \leq 1/\delta$, such that $\sum_{v\in\mathcal{B}_i} d(v) \geq \frac{\delta}{2}|E|$.*

Proof. Each element $v \in \mathcal{A}$ must be a member of at least one of the sets $\mathcal{B}_i$, since

$$\sum_{1\leq i\leq 1/\delta} \sum_{u\in C^i \sim v} \frac{1}{d(u)} = \sum_{u\sim v} \frac{1}{d(u)} \geq \frac{1}{3} \quad .$$

Therefore, there is at least one set $\mathcal{B}_i$ that contributes at least a δ-fraction of the sum $\sum_{v\in\mathcal{A}} d(v)$, i.e., $\sum_{v\in\mathcal{B}_i} d(v) \geq \frac{\delta}{2}|E|$. □

Henceforth we will fix i to be a value satisfying Corollary 16, and let $\mathcal{B} := \mathcal{B}_i$ and $Q_0 := C^i$. With this notation, we are now ready to present the outline of our algorithm:

Algorithm 3 MIS algorithm outline

```
while $|E(G)| > 0$ do
    Add all isolated nodes to MIS; remove them from $G$
    Compute $i$, $\mathcal{B}$ and $Q_0$
    Select a set $Q' \subseteq Q_0$ that induces a low degree subgraph
    Find independent set $\mathcal{I} \subseteq Q'$ with $\sum_{v\in N(\mathcal{I})} d(v) = \Omega(|E(G)|)$
    Add $\mathcal{I}$ to MIS; remove $\mathcal{I}$ and $N(\mathcal{I})$ from $G$
end while
```

Notice that since in each round we find an independent set $\mathcal{I}$ with $\sum_{v\in N(\mathcal{I})} d(v) = \Omega(|E(G)|)$, it is easy to see that Algorithm 3 finds an MIS in $O(\log n)$ rounds. Hence our goal is to find an independent set $\mathcal{I}$ with $\sum_{v\in N(\mathcal{I})} d(v) = \Omega(|E(G)|)$ in $O(1)$ MPC rounds.

As one can see, Algorithm 3 is very similar to Algorithm 2, and the major difference is that in Algorithm 3 we sub-sample nodes instead of edges, since we cannot afford to have removed any edges between nodes we are considering for our independent set $\mathcal{I}$.

Similarly to matching (cf. Section 3.1), computing i, $\mathcal{B}$ and Q_0 can be completed in $O(1)$ MPC rounds using several applications of Lemma 4. Therefore in the following Sections 4.2–4.3 we will first show how to deterministically construct in $O(1)$ MPC rounds an appropriated set $Q' \subseteq Q_0$ that induces a low degree subgraph and then how to deterministically find in $O(1)$ MPC rounds an independent set $\mathcal{I} \subseteq Q'$ such that $\sum_{v \in N(\mathcal{I})} d(v) = \Omega(|E(G)|)$.

4.2 Deterministically selecting $Q' \subseteq Q_0$ that induces a low degree subgraph

We will show now how to deterministically, in $O(1)$ stages, find a subset Q' of Q_0 that induces a low degree subgraph, as required in our MPC algorithm for MIS. For that, our main goal is to ensure that every node has degree $O(n^{4\delta})$ in Q' (to guarantee that its 2-hop neighborhood fits a single MPC machine with $S = O(n^{8\delta})$), and that one can then locally find an independent $\mathcal{I} \subseteq Q'$ that covers a linear number of edges.

We again proceed in $i - 4$ stages (if $i \leq 4$, then similarly to Algorithm 2, we will use $Q' = Q_0$), starting with Q_0 and sampling a new set Q_j ($Q_j \subseteq Q_{j-1}$) in each stage $j = 1, 2, \ldots, i - 4$. The *invariant* we will maintain is that, after every stage j, $0 \leq j \leq i - 4$:

(i) all nodes $v \in Q_j$ have $d_{Q_j}(v) \leq (1 + o(1))n^{-j\delta} d(v)$, and
(ii) all nodes $v \in \mathcal{B}$ have $\sum_{u \in Q_j \sim v} \frac{1}{d(u)} \geq \frac{\delta - o(1)}{3n^{j\delta}}$.

It is easy to see that the invariant holds for $j = 0$ trivially, by definition of Q_0 and $\mathcal{B}$. In what follows, we will show how, for a given set Q_{j-1} satisfying the invariant, to construct in $O(1)$ rounds a new set $Q_j \subseteq Q_{j-1}$ that satisfies the invariant too.

Distributing edges and nodes among the machines. In order to implement our scheme in the MPC model, we first allocate the nodes and the edges of the graph among the machines.

- Each node v in Q_{j-1} distributes its adjacent edges to nodes in Q_{j-1} across a group of machines (*type Q machines*), with at most one machine having fewer than $n^{4\delta}$ edges and all other machines having exactly $n^{4\delta}$ edges.
- Each node v in $\mathcal{B}$ distributes its adjacent edges to nodes in Q_{j-1} across a group of machines (*type $\mathcal{B}$ machines*), with at most one machine having fewer than $n^{4\delta}$ edges and all other machines having exactly $n^{4\delta}$ edges.

Note that nodes may be in both Q_{j-1} and $\mathcal{B}$ and need only one group of machines, but for the ease of analysis we treat the groups of machines separately. Similarly to Section 3.2, type Q machines will ensure Invariant (i) and type $\mathcal{B}$ machines will ensure Invariant (ii).

In order to select $Q_{j-1} \subseteq Q_j$, we will first fix a seed specifying a hash function from a $\mathcal{H}$. Each hash function h induces a candidate set Q_h into which node in Q_{j-1} is "sampled with probability $n^{-\delta}$", by placing v into Q_h iff $h(v) \leq n^{3-\delta}$.

Type Q machines. Consider a type Q machine x that gets allocated edges $V(x) \subseteq Q_{j-1}$ and let $v_x := |V(x)|$. For hash function $h \in \mathcal{H}$, we say x is *good* if $|V(x) \cap Q_h| \leq v_x n^{-\delta} + n^{0.1\delta}\sqrt{v_x}$.

Each of the indicator random variables $\mathbf{1}_{\{v \in Q_h\}}$ is c-wise independent, and has expectation $n^{-\delta}$. Therefore we can apply to these random variable Lemma 9: taking Z to be the sum of the indicator variables for $V(x)$ (i.e., $Z = |V(x) \cap Q_h|$), and choosing a sufficiently large constant c, Lemma 9 implies that $\mathbf{Pr}\left[|Z - \mu| \geq n^{0.1\delta}\sqrt{v_x}\right] \leq n^{-5}$. This means that with high probability, $|V(x) \cap Q_h| \leq v_x n^{-\delta} + n^{0.1\delta}\sqrt{v_x}$, and x is good.

Type $\mathcal{B}$ machines. Consider a type $\mathcal{B}$ machine x that gets allocated edges $V(x) \subseteq Q_{j-1}$; let $v_x := |V(x)|$. For $h \in \mathcal{H}$, we call x *good* if $\sum_{v \in V(x) \cap Q_h} \frac{1}{d(v)} \geq n^{-\delta} \sum_{v \in V(x)} \frac{1}{d(v)} - n^{(0.9-i)\delta} \sqrt{v_x}$.

As before, we will apply Lemma 9, setting $Z_v = \frac{n^{(i-1)\delta}}{d(v)} \mathbf{1}_{\{v \in Q_h\}}$ and $Z = \sum_{v \in V(x)} Z_v$. Since $V(x) \subseteq Q$, each $d(v)$ is at least $n^{(i-1)\delta}$, and so the variables Z_v take values in $[0,1]$. They have expectation $\mathbf{E}[Z_v] = \frac{n^{(i-2)\delta}}{d(v)}$, and as before, they are c-wise independent. Hence, we can apply Lemma 9 with sufficiently large c to find that $\mathbf{Pr}\left[|Z-\mu| \geq n^{0.1\delta}\sqrt{v_x}\right] \leq n^{-5}$. Hence, with high probability,

$$n^{(i-1)\delta} \sum_{v \in V(x) \cap Q_h} \frac{1}{d(v)} \geq n^{(i-2)\delta} \sum_{v \in V(x)} \frac{1}{d(v)} - n^{0.1\delta}\sqrt{v_x} \ ,$$

and therefore $\sum_{v \in V(x) \cap Q_h} \frac{1}{d(v)} \geq n^{-\delta} \sum_{v \in V(x)} \frac{1}{d(v)} - n^{(0.9-i)\delta}\sqrt{v_x}$, so x is good.

Since there are at most $\frac{2n^2}{S} + 2n \leq n^2$ machines, by a union bound the probability that a particular hash function $h \in \mathcal{H}$ makes all machines good is at least $1 - n^{-3}$. The expected number of machines which are not good for a random choice of function is therefore less than 1. So, by the method of conditional expectations (cf. Section 2.4), using objective $q_x(h) = \mathbf{1}_{x \text{ is good for } h}$, in a constant number of MPC rounds we can find a hash function $h \in \mathcal{H}$ which makes all machines good. We then use such hash function h to set $Q_j = Q_h$.

4.2.1 Properties of Q_j: satisfying the invariants. Having fixed a sub-sampled set of nodes Q_j for the stage, we need to show that since all machines were good, we satisfy our invariants for the stage.

Lemma 17 (Invariant (i)). *All nodes $v \in Q_j$ satisfy*

$$d_{Q_j}(v) \leq (1+o(1))n^{-j\delta}d(v) \ .$$

Proof. Node v's neighbors in Q_{j-1} were divided among $\lfloor \frac{d_{Q_{j-1}}(v)}{n^{4\delta}} \rfloor$ type Q machines containing $n^{4\delta}$ neighbors, and one type Q machine containing the remaining $d_{Q_{j-1}}(v) - n^{4\delta}\lfloor \frac{d_{Q_{j-1}}(v)}{n^{4\delta}} \rfloor \leq d_{Q_{j-1}}(v)$ neighbors. Therefore we obtain,

$$\begin{aligned} d_{Q_j}(v) &\leq \sum_{v\text{'s machines } x} v_x n^{-\delta} + n^{0.1\delta}\sqrt{v_x} \\ &= n^{-\delta} d_{Q_{j-1}}(v) + \left\lfloor \frac{d_{Q_{j-1}}(v)}{n^{4\delta}} \right\rfloor n^{0.1\delta}\sqrt{n^{4\delta}} + n^{0.1\delta}\sqrt{d_{Q_{j-1}}(v)} \\ &\leq n^{-\delta} d_{Q_{j-1}}(v) + \frac{n^{0.1\delta}}{n^{2\delta}} d_{Q_{j-1}}(v) + n^{0.1\delta}\sqrt{d_{Q_{j-1}}(v)} \ . \end{aligned}$$

If $d_{Q_{j-1}}(v) \geq n^{3\delta}$, we have

$$\begin{aligned} d_{Q_j}(v) &\leq n^{-\delta} d_{Q_{j-1}}(v) + \frac{n^{0.1\delta}}{\sqrt{s}} d_{Q_{j-1}}(v) + n^{0.1\delta}\sqrt{d_{Q_{j-1}}(v)} \\ &\leq n^{-\delta} d_{Q_{j-1}}(v) + n^{-1.9\delta} d_{Q_{j-1}}(v) + n^{-1.4\delta} d_{Q_{j-1}}(v) \\ &= (1+o(1))n^{-\delta} d_{Q_{j-1}}(v) \leq (1+o(1))n^{-j\delta} d(v) \ . \end{aligned}$$

Otherwise, $d_{Q_j}(v) \leq d_{Q_{j-1}}(v) \leq n^{3\delta} \leq n^{-j\delta}d(v)$. In either case, we satisfy the invariant for stage j. □

Lemma 18 (Invariant (ii)). *All nodes $v \in \mathcal{B}$ satisfy*

$$\sum_{u \in Q_j \sim v} \frac{1}{d(u)} \geq \frac{\delta - o(1)}{3n^{j\delta}} \quad .$$

Proof. Node v's neighbors in Q_{j-1} were again divided among $\lfloor \frac{d_{Q_{j-1}}(v)}{n^{4\delta}} \rfloor$ type $\mathcal{B}$ machines containing $n^{4\delta}$ neighbors, and one type $\mathcal{B}$ machine containing the remaining $d_{Q_{j-1}}(v) - n^{4\delta}\lfloor \frac{d_{Q_{j-1}}(v)}{n^{4\delta}} \rfloor \leq d_{Q_{j-1}}(v)$ neighbors. Denote $y := \sum_{u \in Q_{j-1} \sim v} \frac{1}{d(u)}$ for brevity.

$$\begin{aligned}
\sum_{u \in Q_j \sim v} \frac{1}{d(u)} &= \sum_{v\text{'s machines } x} \sum_{u \in V(x) \cap Q_j} \frac{1}{d(u)} \\
&\geq \sum_{v\text{'s machines } x} \left(n^{-\delta} \sum_{u \in V(x)} \frac{1}{d(u)} - n^{(1.9-i)\delta} \sqrt{v_x} \right) \\
&\geq n^{-\delta} \left(\sum_{u \in Q_{j-1} \sim v} \frac{1}{d(u)} - n^{(1.9-i)\delta} \left(\left\lfloor \frac{d_{Q_{j-1}}(v)}{n^{4\delta}} \right\rfloor \sqrt{n^{4\delta}} + \sqrt{d_{Q_{j-1}}(v)} \right) \right) \\
&\geq n^{-\delta} \left(y - n^{(1.9-i)\delta} \left(n^{-2\delta} d_{Q_{j-1}}(v) + \sqrt{d_{Q_{j-1}}(v)} \right) \right) \quad .
\end{aligned}$$

We know that $d_{Q_{j-1}}(v) \leq n^{i\delta} \sum_{u \in Q_{j-1} \sim v} \frac{1}{d(u)} = n^{i\delta} y$, since all nodes $u \in Q_{j-1}$ are in Q and have degree at most $n^{i\delta}$. So,

$$\begin{aligned}
\sum_{u \in Q_j \sim v} \frac{1}{d(u)} &\geq n^{-\delta} \left(y - n^{(1.9-i)\delta} \left(n^{-2\delta} d_{Q_{j-1}}(v) + \sqrt{d_{Q_{j-1}}(v)} \right) \right) \\
&\geq n^{-\delta} \left(y - n^{(1.9-i)\delta} \left(n^{-2\delta} n^{i\delta} y + \sqrt{n^{i\delta} y} \right) \right) \\
&= n^{-\delta} \left(y - n^{-0.1\delta} y - n^{(1.9-0.5i)\delta} \sqrt{y} \right) \\
&= n^{-\delta} \left(y - o(y) - n^{(1.9-0.5i)\delta} \sqrt{y} \right) \quad .
\end{aligned}$$

From our invariant we know that $y \geq \frac{\delta - o(1)}{4n^{\delta(j-1)}}$, and so $\sqrt{y} \leq \frac{y}{\sqrt{\frac{\delta - o(1)}{4n^{\delta(j-1)}}}} \leq \frac{y}{\delta} n^{0.5\delta(j-1)}$. Hence,

$$\begin{aligned}
\sum_{u \in Q_j \sim v} \frac{1}{d(u)} &\geq n^{-\delta} \left(y - o(y) - n^{(1.9-0.5i)\delta} \sqrt{y} \right) \\
&\geq n^{-\delta} \left(y - o(y) - n^{(1.9-0.5i)\delta} \cdot \frac{y}{\delta} n^{0.5\delta(j-1)} \right) \\
&\geq n^{-\delta} \left(y - o(y) - \frac{y}{\delta} \cdot n^{(1.4-0.5i+0.5j)\delta} \right) \\
&\geq n^{-\delta} \left(y - o(y) - \frac{y}{\delta} \cdot n^{(1.4-0.5i+0.5(i-4))\delta} \right) \\
&\geq n^{-\delta} \left(y - o(y) - \frac{y}{\delta} \cdot n^{-0.6\delta} \right) \geq n^{-\delta} \left(y - o(y) \right) \\
&\geq n^{-\delta} \cdot \frac{\delta - o(1)}{3n^{\delta(j-1)}} \geq \frac{\delta - o(1)}{3n^{j\delta}} \quad .
\end{aligned}$$

□

Since our invariants are preserved in every stage, they hold in our final sub-sampled node set $Q' := Q_{i-4}$.

4.3 Finding an independent set $\mathcal{I}$

After $i-4$ stages, we now have a node set $Q' := Q_{i-4}$ with the following properties (cf. Lemmas 17 and 18):

(i) all nodes $v \in Q'$ have $d_{Q'}(v) \leq (1+o(1))n^{(4-i)\delta}d(v) \leq 2n^{4\delta}$;
(ii) all nodes $v \in \mathcal{B}$ have $\sum_{u\in Q'\sim v} \frac{1}{d(u)} \geq \frac{\delta - o(1)}{3n^{(i-4)\delta}}$.

(If $i \leq 4$, we instead have that for $v \in Q'$, $d_{Q'}(v) \leq n^{i\delta}$ and for $v \in B$, $\sum_{u\in Q'\sim v} \frac{1}{d(u)} \geq \frac{\delta}{3}$ from setting $Q' = Q_0$).

We now show a property analogous to Lemma 12 (and hence Lemma 3) for the node set Q'.

Lemma 19. *For each node $v \in \mathcal{B}$, either v has a neighbor $u \in Q'$ with $d_{Q'}(u) = 0$ or v satisfies $\sum_{u\in Q'\sim v} \frac{1}{d_{Q'}(u)} \geq 0.1\delta$.*

Proof. The claim trivially holds for $i \leq 4$. Otherwise, for every $v \in \mathcal{B}$ with no neighbor $u \in Q'$ with $d_{Q'}(u) = 0$, we have the following,

$$\sum_{u\in Q'\sim v} \frac{1}{d_{Q'}(u)} \geq \sum_{u\in Q'\sim v} \frac{1}{2n^{(4-i)\delta}d(u)} = \frac{1}{2n^{(4-i)\delta}} \sum_{u\in Q'\sim v} \frac{1}{d(u)} \geq \frac{1}{2n^{(4-i)\delta}} \cdot \frac{\delta}{5n^{(i-4)\delta}} = 0.1\delta \ ,$$

where the first inequality follows from Invariant (i) (Lemma 17) and the second one from Invariant (ii) (Lemma 18). □

Now we are ready to present our deterministic MPC algorithm that for a given subset of nodes Q' satisfying the invariant, in $O(1)$ rounds constructs an independent set $\mathcal{I} \subseteq Q'$ such that the removal of $\mathcal{I} \cup N(\mathcal{I})$ removes $\Omega(\delta|E|)$ edges from the graph.

Each node $v \in \mathcal{B}$ is assigned a machine x_v which gathers a set N_v of up to $n^{4\delta}$ of v's neighbors in Q' (if v has more than $n^{4\delta}$ neighbors in Q', then take an arbitrary subset of $n^{4\delta}$ of them), along with all of their neighborhoods in Q' (i.e., $N_{Q'}(N_v)$). By Invariant (i), this requires at most $n^{4\delta} \cdot 2n^{4\delta} = O(n^{8\delta})$ space per machine. Since $|\mathcal{B}| \leq n$, this is $O(n^{1+8\delta})$ total space. We prove that these sets N_v preserve the desired property:

Lemma 20. *Each node $v \in \mathcal{B}$ either has a neighbor $u \in Q'$ with $d_{Q'}(u) = 0$ or satisfies $\sum_{u\in N_v} \frac{1}{d_{Q'}(u)} \geq 0.1\delta$.*

Proof. If $d_{Q'}(v) \leq n^{4\delta}$ then $N_v = N_{Q'}(v)$ and so the lemma holds by Lemma 19.
Otherwise we have $|N_v| = n^{4\delta}$, and so by Invariant (i) in the first inequality, we get,

$$\sum_{u\in N_v} \frac{1}{d(u)} \geq \sum_{u\in N_v} \frac{1}{2n^{4\delta}} = \frac{n^{4\delta}}{2n^{4\delta}} = \frac{1}{2} > 0.1\delta \ . \qquad \square$$

We now do one further derandomization step to find an independent set. We will fix a seed specifying a hash function from $\mathcal{H}$. This hash function h will be used to map each node v in Q' to a value $z_v \in [n^3]$. Then, v joins the *candidate independent set* $\mathcal{I}_h$ iff $z_v < z_u$ for all $u \sim v$.

Clearly $\mathcal{I}_h$ is indeed an independent set, but we want to show that removing $\mathcal{I}_h \cup N(\mathcal{I}_h)$ from the graph reduces the number of edges by a constant fraction. We will show that in expectation (over a random choice of $h \in \mathcal{H}$) this is indeed the case, and then we can apply the method of conditional expectations (cf. Section 2.4) to conclude the construction.

Each machine x_v is *good* for a hash function $h \in \mathcal{H}$ if it holds a node $u \in N_v \cap \mathcal{I}_h$. Since x_v holds the neighborhoods in Q' of nodes in N_v, it can determine whether they are members of $\mathcal{I}_h$. We show that with constant probability, x_v is good for a random hash function $h \in \mathcal{H}$.

Lemma 21. *For any machine x_v holding a set N_v and its neighborhood in Q', with probability at least* 0.01δ *(over a choice of a random hash function $h \in \mathcal{H}$) it holds that $|N_v \cap \mathcal{I}_h| \geq 1$.*

Proof. If any node $u \in N_v$ has $d_{Q'}(u) = 0$, it will join $\mathcal{I}_h$ and we are done.

Otherwise, by Lemma 20 we have $\sum_{u \in N_v} \frac{1}{d_{Q'}(u)} \geq 0.1\delta$ and we will consider, for the sake of the analysis, some arbitrary subset $N_v^* \subseteq N_v$ such that $1 \geq \sum_{u \in N_v^*} \frac{1}{d_{Q'}(u)} \geq 0.1\delta$.

For any $u \in N_v^*$ we have

$$\frac{1}{3d_{Q'}(u)} - \frac{1}{n^3} \leq \mathbf{Pr}\left[z_u < \frac{n^3}{3d_{Q'}(u)}\right] \leq \frac{1}{3d_{Q'}(u)} \quad .$$

Let $\mathcal{A}_u$ be the event $\left\{u \in \mathcal{I}_h \wedge z_u < \frac{n^3}{3d_{Q'}(u)}\right\}$. By pairwise independence,

$$\begin{aligned}
\mathbf{Pr}\left[A_u\right] &= \mathbf{Pr}\left[u \in \mathcal{I}_h \wedge z_u < \frac{n^3}{3d_{Q'}(u)}\right] \\
&= \mathbf{Pr}\left[z_u < \frac{n^3}{3d_{Q'}(u)}\right] - \mathbf{Pr}\left[u \notin \mathcal{I}_h \wedge z_u < \frac{n^3}{3d_{Q'}(u)}\right] \\
&= \mathbf{Pr}\left[z_u < \frac{n^3}{3d_{Q'}(u)}\right] - \mathbf{Pr}\left[\bigcup_{w \in Q' \sim u} \left\{z_w \leq z_u < \frac{n^3}{3d_{Q'}(u)}\right\}\right] \\
&\geq \frac{1}{3d_{Q'}(u)} - \frac{1}{n^3} - \sum_{w \in Q' \sim u} \mathbf{Pr}\left[z_w \leq z_u < \frac{n^3}{3d_{Q'}(u)}\right] \\
&\geq \frac{1}{3d_{Q'}(u)} - \frac{1}{n^3} - \sum_{w \in Q' \sim u} \mathbf{Pr}\left[z_w, z_u < \frac{n^3}{3d_{Q'}(u)}\right] \\
&= \frac{1}{3d_{Q'}(u)} - \frac{1}{n^3} - d_{Q'}(u) \cdot \frac{1}{(3d_{Q'}(u))^2} = \frac{2}{9d_{Q'}(u)} - \frac{1}{n^3} \quad .
\end{aligned}$$

Furthermore, for any $u \neq u'$, again by pairwise independence of hash functions from $\mathcal{H}$,

$$\begin{aligned}
\mathbf{Pr}\left[A_u \cap A_{u'}\right] &= \mathbf{Pr}\left[u \in \mathcal{I}_h \wedge z_u < \frac{n^3}{3d_{Q'}(u)} \wedge u' \in \mathcal{I}_h \wedge z_{u'} < \frac{n^3}{3d_{Q'}(u')}\right] \\
&\leq \mathbf{Pr}\left[z_u < \frac{n^3}{3d_{Q'}(u)} \wedge z_{u'} < \frac{n^3}{3d_{Q'}(u')}\right] \leq \frac{1}{9 \cdot d_{Q'}(u) \cdot d_{Q'}(u')} \quad .
\end{aligned}$$

So, by the principle of inclusion-exclusion,

$$\begin{aligned}\mathbf{Pr}\left[\bigcup_{u\in N_v^*} A_u\right] &\geq \sum_{u\in N_v^*}\mathbf{Pr}\left[A_u\right] - \sum_{\{u,u'\}\subseteq N_v^*, u\neq u'}\mathbf{Pr}\left[A_u \cap A_{u'}\right]\\
&\geq \sum_{u\in N_v^*}\left(\frac{2}{9d_{Q'}(u)} - \frac{1}{n^3}\right) - \sum_{\{u,u'\}\subseteq N_v^*, u\neq u'}\frac{1}{9d_{Q'}(u)d_{Q'}(u')}\\
&\geq \sum_{u\in N_v^*}\frac{2}{9d_{Q'}(u)} - \frac{1}{18}\left(\sum_{u\in N_v^*}\frac{1}{d_{Q'}(u)}\right)^2 - \frac{1}{n^2}\\
&\geq \frac{2}{9}\sum_{u\in N_v^*}\frac{1}{d_{Q'}(u)} - \frac{1}{18}\sum_{u\in N_v^*}\frac{1}{d_{Q'}(u)} - \frac{1}{n^2}\\
&= \frac{1}{6}\sum_{u\in N_v^*}\frac{1}{d_{Q'}(u)} - \frac{1}{n^2}\\
&\geq 0.016\delta - \frac{1}{n^2}\ .\end{aligned}$$

Here we use that, by our choice of N_v^*, we have $\left(\sum_{u\in N_v^*}\frac{1}{d_{Q'}(u)}\right)^2 \leq \sum_{u\in N_v^*}\frac{1}{d_{Q'}(u)} \leq 1$ and $0.1\delta \leq \sum_{u\in N_v^*}\frac{1}{d_{Q'}(u)}$.

Therefore, for n larger than a sufficiently large constant, we have the following,

$$\mathbf{Pr}\left[N_v \cap \mathcal{I}_h \neq \emptyset\right] \geq \mathbf{Pr}\left[\bigcup_{u\in N_v^*} A_u\right] \geq 0.016\delta - \frac{1}{n^2} \geq 0.01\delta\ . \qquad \square$$

For a hash function $h \in \mathcal{H}$, we will denote $N_h := \{v \in \mathcal{B} : N_v \cap \mathcal{I}_h \neq \emptyset\}$, i.e., the set of nodes to be removed if the independent set induced by hash function h is chosen. By Lemma 21 and by the definition of $\mathcal{B}$ (which ensures $\sum_{v\in\mathcal{B}} d(v) \geq \frac{\delta}{2}|E|$),

$$\mathbf{E}\left[\sum_{v\in N_h} d(v)\right] \geq \sum_{v\in\mathcal{B}} d(v)\cdot\mathbf{Pr}\left[v \in N_h\right] \geq 0.01\delta\sum_{v\in\mathcal{B}} d(v) \geq \frac{\delta^2|E|}{200}\ .$$

By the method of conditional expectations (cf. Section 2.4), using $q_{x_v}(h) = d(v)\mathbf{1}_{x_v \text{ is good for } h}$, we can select a hash function h with $\sum_{v\in N_h} d(v) \geq \frac{\delta^2}{200}\cdot|E|$. We then add the independent set $\mathcal{I} := \mathcal{I}_h$ to our output, and remove $\mathcal{I}$ and $N(\mathcal{I})$ from the graph. In doing so, we remove at least $\frac{1}{2}\sum_{v\in N_h} d(v) \geq \frac{\delta^2|E|}{400}$ edges from the graph.

4.4 Completing the proof of Theorem 14: finding MIS

Now we are ready to complete the proof of Theorem 14. Our algorithm returns an MIS in at most $\log_{\frac{1}{1-\delta^2/400}}|E| = O(\log n)$ stages, each stage of the algorithm, as described above, taking a constant number of rounds in MPC. The space required is dominated by storing the input graph G ($O(m)$ total space) and collecting node neighborhoods when finding an independent set ($O(n^{8\delta})$ space per machine, $O(n^{1+8\delta})$ total space). Setting $\delta = \frac{\varepsilon}{8}$ allows us to conclude Theorem 14 by obtaining that for any constant $\varepsilon > 0$, MIS can be found deterministically in MPC in $O(\log n)$ rounds, using $O(n^{\varepsilon})$ space per machine and $O(m + n^{1+\varepsilon})$ total space. $\square$

5 MIS AND MAXIMAL MATCHING IN $O(\log\Delta + \log\log n)$ MPC ROUNDS

Our efforts in Sections 3–4 were on achieving deterministic MIS and maximal matching algorithms running in $O(\log n)$ MPC rounds; with some additional work we can improve them for graphs with low maximum degree Δ, obtaining deterministic $O(\log\Delta + \log\log n)$-round MPC algorithms.

Let us first observe that it is sufficient to consider the case where $\Delta \le n^\delta$, as otherwise we can use the $O(\log n)$ algorithm from Theorem 14 to achieve an $O(\log\Delta)$-round MPC algorithm. Therefore we no longer need to perform graph sparsification, since we can already fit 2-hop neighborhoods (and, indeed, $O\left(\frac{\log n}{\log\Delta}\right)$-hop neighborhoods) single machines.

We use this observation to perform *round compression* on Luby's algorithm: since we can collect $O\left(\frac{\log n}{\log\Delta}\right)$-hop neighborhoods, machines have all the information they need to perform $O\left(\frac{\log n}{\log\Delta}\right)$ steps of Luby's algorithm *at once*, and therefore the $O(\log n)$ total necessary steps can be performed in only $O(\log\Delta)$ MPC rounds, with $O(\log\log n)$ rounds of pre-processing needed to collect the neighborhoods (by graph exponentiation). We show this round compression technique only for MIS, and use a standard reduction to also obtain a result for maximal matching, but one could similarly compress the rounds of the maximal matching version of Luby's algorithm to reach such a result directly.

Algorithm 4 $O(\log\Delta + \log\log n)$-round MIS algorithm for $\Delta \le n^\delta$

```
Compute a $\Delta^4$-coloring of $G^2$
Collect $\frac{2\delta\log n}{\log\Delta}$-hop neighborhoods in $G$
for $O(\log\Delta)$ iterations: do
    Derandomize $\frac{\delta\log n}{\log\Delta}$ steps of Luby's algorithm
    Update $\frac{2\delta\log n}{\log\Delta}$-hop neighborhoods with removed nodes
end for
```

Algorithm 4 presents an overview of the procedure; we now go into more detail on its constituent parts.

5.1 Precomputation: Δ^4-coloring and neighborhood collection

For purposes of efficient derandomization, as we will discuss in more detail shortly, we require hash functions with domain which is $poly(\Delta)$, rather than $poly(n)$, in order to have only $O(\log\Delta)$ seed length. We can therefore no longer use nodes' identifiers as the hash function domain. However, it is well known that Luby's algorithm only requires independence between nodes that are at most 2-hops apart. This allows us to reduce the seed length from $O(\log n)$ to $O(\log\Delta)$ by assigning every node a new name with only $O(\log\Delta)$ bits, such that every pair of nodes with distance two has distinct names; this task is equivalent to vertex coloring in the graph G^2. For every graph G with maximum degree Δ, Linial [43] showed an $O(\Delta^2)$-coloring using $O(\log^* n)$ rounds in the LOCAL model, and Kuhn [39] extended this algorithm and showed that it can implemented also in the CONGEST model within the same number of rounds.

In our context, since we wish to color the graph G^2, we need to compute a $O(\Delta^4)$-coloring χ on G^2. We can do so by dedicating a machine to each node of G, collecting that node's 2-hop neighborhood onto the machine, and then simulating Kuhn's CONGEST algorithm on G^2 in a straightforward round-by-round fashion (in MPC, machines can communicate directly with those representing nodes in the 2-hop neighborhood, and the CONGEST communication restrictions imply adherence to the MPC space bounds).

Graph exponentiation for neighborhood collection. In order to collect d-hop neighborhoods in $O(\log d)$ rounds for $d = \frac{2\delta \log n}{\log \Delta}$, we apply the standard technique of graph exponentiation: each node is assigned a machine, and we iteratively expand the size of the neighborhood of that node collected onto its machine. In round i, if all machines contain a r_i-hop neighborhood for their nodes, then each send this neighborhood knowledge to the machines for all nodes in the neighborhood. So, machines receive the r_i-hop neighborhood for all nodes within the r_i hops of their node; i.e., they now know a $2r_i$-hop neighborhood. In this way we double the radius of the neighborhood in each MPC round, and so we collect $\frac{2\delta \log n}{\log \Delta}$-hop neighborhoods in $O(\log \log n)$ rounds. Of course, this process is subject to memory constraints, but since maximum degree is Δ, the size of the $\frac{2\delta \log n}{\log \Delta}$-hop neighborhoods is at most $\Delta^{\frac{2\delta \log n}{\log \Delta}} = n^{2\delta}$, and since we choose δ to be a constant sufficiently smaller than ε, this fits in the space of a machine.

5.2 Round-compressed derandomization of Luby's algorithm

Our algorithm consists of $O(\log \Delta)$ phases, each implemented in $O(1)$ MPC rounds. In each phase, the algorithm derandomized $d = \frac{\delta \log n}{\log \Delta}$ steps of Luby's algorithm. Without loss of generality, we focus on the first phase and explain this round-compressed derandomization in details.

We will use hash functions from a pairwise independent family $\mathcal{H}^*$ of functions $h : [\Delta^4] \to [c\Delta^4]$, for sufficiently large constant c (as before we can assume that we round up domain and range to a power of 2). For a single *step*, each node v will use a hash function h from $\mathcal{H}^*$ to map its color in the $O(\Delta^4)$-coloring χ to a value $z_v \in [\Delta^4]$. It then joins the independent set $\mathcal{I}$ if its value z_v is lower than any of its neighbours'. By the same argument as Lemma 21 (and the classic derandomization of Luby [45]), we find that under a uniformly random choice of h, node v joins the MIS with probability $\Omega(\frac{1}{d(v)})$ (note that the error term incurred by the rounding of hash function range is less than $\frac{1}{c\Delta^2}$, and therefore negligible by choice of sufficiently large c).

Then, by Lemma 3, the critical property of the analysis of Luby's algorithm, deleting $\mathcal{I} \cup N(\mathcal{I})$ removes a constant fraction of the edges from the graph in expectation.

Expected value of a sequence of hash functions. We need to analyze $\frac{\delta \log n}{\log \Delta}$ steps, i.e., the effect of using sequences of $\frac{\delta \log n}{\log \Delta}$ independent uniformly random hash functions from $\mathcal{H}^*$. Let E_i denote the number of edges in the graph at the beginning of step i. We know from the above that $\mathbf{E}\left[E_{i+1}|E_i\right] \le pE_i$, for some $p < 1$. By independence, therefore, $\mathbf{E}\left[E_{\frac{\delta \log n}{\log \Delta}}\right] \le p^{\frac{\delta \log n}{\log \Delta}} E_1$.

Derandomizing via the method of conditional expectations. We now perform the method of conditional expectations, as defined previously, to allow all machines to agree on a good *sequence* of hash functions to use for the $\frac{\delta \log n}{\log \Delta}$ steps. Since we are now using c-wise independent hash functions with $[O(\Delta^4)]$ domain and range, by Lemma 6, we need an $O(\log \Delta)$-bit seed to specify each. Therefore the *total* seed length for all $\frac{\delta \log n}{\log \Delta}$ steps is $O(\log n)$, and so we can perform the method of conditional expectations in $O(1)$ rounds. We do so to choose a sequence of hash functions such that $E_{\frac{\delta \log n}{\log \Delta}} \le p^{\frac{\delta \log n}{\log \Delta}} E_1$, i.e. we reduce the number of edges in the graph by an $n^{\Omega(\frac{1}{\log \Delta})}$ factor.

Since, for each node v, we have collected the $\frac{2\delta \log n}{\log \Delta}$-hop neighborhood of v onto a machine, that machine has all necessary information to immediately determine how many of v's adjacent edges remain after applying any hash function sequence. We therefore have the necessary property that the quality of a hash function sequence (number of edges remaining in the graph after application) is the sum of functions computable locally by machines.

5.3 Updating neighborhoods

Once we have performed the method of conditional expectations to simulate $\frac{\delta \log n}{\log \Delta}$ steps of Luby's algorithm and reduce the number of edges in the graph by an $n^{\Omega(\frac{1}{\log \Delta})}$ factor, the final task is to update the $\frac{2\delta \log n}{\log \Delta}$-hop neighborhood held by each machine to reflect the new reduced graph. This can be done simply by having each node v's machine directly inform those of all nodes in its $\frac{2\delta \log n}{\log \Delta}$-hop neighborhood whether v remains in the reduced graph. Thereby all machines are updated of their node's *new* $\frac{2\delta \log n}{\log \Delta}$-hop neighborhood (which is a subset of its old one). This process requires only one MPC round.

5.4 Completing MIS and extending to maximal matching.

We have shown that Algorithm 4, for the case $\Delta \leq n^{\delta}$, has the following properties:

(1) We perform $O(\log \log n)$ rounds of precomputation to compute a Δ^4-coloring of G^2, and gather $\frac{2\delta \log n}{\log \Delta}$-hop neighborhoods.
(2) We then perform iterations which require $O(1)$ rounds each, and reduce the number of edges by an $n^{\Omega(\frac{1}{\log \Delta})}$ factor.

After $O(\log \Delta)$ iterations, therefore, we reduce the number of edges in the graph to 0, and find a maximal independent set. Ensuring that δ is sufficiently smaller than ε, we have used $O(n^{\varepsilon})$ space per machine. We required $O(n)$ machines, so total space is $O(n^{1+\varepsilon})$. Combining this algorithm with Algorithm 3 for the case $\Delta > n^{\delta}$ completes the MIS part of Theorem 1.

Reducing maximal matching to MIS. To obtain the same round complexity for maximal matching, we note that in the $\Delta \leq n^{\delta}$ regime we can apply the the standard reduction of performing MIS on the line graph of the input graph. Since we are collecting $\frac{2\delta \log n}{\log \Delta}$-hop neighborhoods already, we have the necessary information to simulate running on the line graph.

We again use $n^{O(\delta)}$ space per machine; global space is now $m \cdot n^{O(\delta)} \leq n^{1+\delta} \cdot n^{O(\delta)}$, which, setting δ sufficiently smaller than ε, is again $O(n^{\varepsilon})$ and $O(n^{1+\varepsilon})$ local and global space respectively. Combining with the bounds of Algorithm 2 completes the maximal matching part of Theorem 1.

6 CONCLUSIONS

In this paper we study the power of deterministic algorithms on the nowadays classical model of parallel computations — the *Massively Parallel Computation (MPC) model* — on the example of two fundamental graph problems: maximal matching and maximal independent set. We develop a new deterministic method for graph sparsification while maintaining some desired properties and apply it to design the first $O(\log \Delta + \log \log n)$-round fully scalable deterministic MPC algorithms for maximal matching and MIS (Theorem 1). In combination with previous results, this also gives the first deterministic $O(\log \Delta)$-round CONGESTED CLIQUE algorithms for maximal matching and MIS. We expect our method of derandomizing the sampling of a low-degree graph while maintaining good properties will prove useful for derandomizing many more problems in low space or limited bandwidth models (e.g., the CONGEST model).

ACKNOWLEDGMENTS

Artur Czumaj is partially supported by the Centre for Discrete Mathematics and its Applications (DIMAP), by a Weizmann-UK Making Connections Grant, by IBM Faculty Award, and by Engineering and Physical Sciences Research Council (EPSRC) award EP/N011163/1.

Peter Davies is partially supported by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 754411. He partly conducted this work while at the University of Warwick, supported by the Centre for Discrete Mathematics and its Applications (DIMAP), by a Weizmann-UK Making Connections Grant, and by Engineering and Physical Sciences Research Council (EPSRC) award EP/N011163/1.

Merav Parter is partially supported by a Weizmann-UK Making Connections Grant, and the Minerva grant 713238.

REFERENCES

[1] Noga Alon, László Babai, and Alon Itai. 1986. A Fast and Simple Randomized Parallel Algorithm for the Maximal Independent Set Problem. *Journal of Algorithms* 7, 4 (Dec. 1986), 567–583.

[2] Alexandr Andoni, Aleksandar Nikolov, Krzysztof Onak, and Grigory Yaroslavtsev. 2014. Parallel Algorithms for Geometric Graph Problems. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*. 574–583.

[3] Alexandr Andoni, Zhao Song, Clifford Stein, Zhengyu Wang, and Peilin Zhong. 2018. Parallel Graph Connectivity in Log Diameter Rounds. In *Proceedings of the 59th IEEE Symposium on Foundations of Computer Science (FOCS)*. 674–685.

[4] Alexandr Andoni, Clifford Stein, and Peilin Zhong. 2020. Parallel Approximate Undirected Shortest Paths Via Low Hop Emulators. In *Proceedings of the 52nd Annual ACM Symposium on Theory of Computing (STOC)*. 322–335.

[5] Sepehr Assadi, MohammadHossein Bateni, Aaron Bernstein, Vahab Mirrokni, and Cliff Stein. 2019. Coresets Meet EDCS: Algorithms for Matching and Vertex Cover on Massive Graphs. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 1616–1635.

[6] Sepehr Assadi, Xiaorui Sun, and Omri Weinstein. 2019. Massively Parallel Algorithms for Finding Well-Connected Components in Sparse Graphs. In *Proceedings of the 38th ACM Symposium on Principles of Distributed Computing (PODC)*. 461–470.

[7] Philipp Bamberger, Fabian Kuhn, and Yannic Maus. 2020. Efficient Deterministic Distributed Coloring with Small Bandwidth. In *Proceedings of the 39th ACM Symposium on Principles of Distributed Computing (PODC)*. 243–252.

[8] Paul Beame, Paraschos Koutris, and Dan Suciu. 2013. Communication Steps for Parallel Query Processing. In *Proceedings of the 32nd ACM SIGMOD Symposium on Principles of Database Systems (PODS)*. 273–284.

[9] Soheil Behnezhad, Sebastian Brandt, Mahsa Derakhshan, Manuela Fischer, MohammadTaghi Hajiaghayi, Richard M. Karp, and Jara Uitto. 2019. Massively Parallel Computation of Matching and MIS in Sparse Graphs. In *Proceedings of the 38th ACM Symposium on Principles of Distributed Computing (PODC)*. 481–490.

[10] Soheil Behnezhad, Mahsa Derakhshan, and MohammadTaghi Hajiaghayi. 2018. Brief Announcement: Semi-MapReduce Meets Congested Clique. *CoRR abs/1802.10297* (2018).

[11] Soheil Behnezhad, Laxman Dhulipala, Hossein Esfandiari, Jakub Łącki, and Vahab S. Mirrokni. 2019. Near-Optimal Massively Parallel Graph Connectivity. In *Proceedings of the 60th IEEE Symposium on Foundations of Computer Science (FOCS)*. 1615–1636.

[12] Soheil Behnezhad, Laxman Dhulipala, Hossein Esfandiari, Jakub Łącki, Vahab S. Mirrokni, and Warren Schudy. 2019. Massively Parallel Computation via Remote Memory Access. In *Proceedings of the 31st Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*. 59–68.

[13] Soheil Behnezhad, MohammadTaghi Hajiaghayi, and David G. Harris. 2019. Exponentially Faster Massively Parallel Maximal Matching. In *Proceedings of the 60th IEEE Symposium on Foundations of Computer Science (FOCS)*. 1637–1649.

[14] Mihir Bellare and John Rompel. 1994. Randomness-Efficient Oblivious Sampling. In *Proceedings of the 35th IEEE Symposium on Foundations of Computer Science (FOCS)*. 276–287.

[15] Keren Censor-Hillel, Merav Parter, and Gregory Schwartzman. 2017. Derandomizing Local Distributed Algorithms under Bandwidth Restrictions. In *Proceedings of the 31st International Symposium on Distributed Computing (DISC)*. 11:1–11:16.

[16] Yi-Jun Chang, Manuela Fischer, Mohsen Ghaffari, Jara Uitto, and Yufan Zheng. 2019. The Complexity of $(\Delta + 1)$ Coloring in Congested Clique, Massively Parallel Computation, and Centralized Local Computation. In *Proceedings of the 38th ACM Symposium on Principles of Distributed Computing (PODC)*. 471–480.

[17] Artur Czumaj, Peter Davies, and Merav Parter. 2020. Simple, Deterministic, Constant-Round Coloring in the Congested Clique. In *Proceedings of the 39th ACM Symposium on Principles of Distributed Computing (PODC)*. 309–318.

[18] Artur Czumaj, Jakub Łącki, Aleksander Mądry, Slobodan Mitrović, Krzysztof Onak, and Piotr Sankowski. 2018. Round Compression for Parallel Matching Algorithms. In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC)*. 471–484.

[19] Jeffrey Dean and Sanjay Ghemawat. 2004. MapReduce: Simplified Data Processing on Large Clusters. In *Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation (OSDI)*. 10–10.

[20] Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: Simplified Data Processing on Large Clusters. *Commun. ACM* 51, 1 (Jan. 2008), 107–113.

[21] Janosch Deurer, Fabian Kuhn, and Yannic Maus. 2019. Deterministic Distributed Dominating Set Approximation in the CONGEST Model. In *Proceedings of the 38th ACM Symposium on Principles of Distributed Computing (PODC)*. 94–103.

[22] Mohsen Ghaffari. 2016. An Improved Distributed Algorithm for Maximal Independent Set. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 270–277.

[23] Mohsen Ghaffari, Themis Gouleakis, Christian Konrad, Slobodan Mitrović, and Ronitt Rubinfeld. 2018. Improved Massively Parallel Computation Algorithms for MIS, Matching, and Vertex Cover. In *Proceedings of the 37th ACM Symposium on Principles of Distributed Computing (PODC)*. 129–138.

[24] Mohsen Ghaffari, Christoph Grunau, and Ce Jin. 2020. Improved MPC Algorithms for MIS, Matching, and Coloring on Trees and Beyond. 179 (2020), 34:1–34:18. https://doi.org/10.4230/LIPIcs.DISC.2020.34

[25] Mohsen Ghaffari, David G. Harris, and Fabian Kuhn. 2018. On Derandomizing Local Distributed Algorithms. In *Proceedings of the 59th IEEE Symposium on Foundations of Computer Science (FOCS)*. 662–673.

[26] Mohsen Ghaffari and Fabian Kuhn. 2018. Derandomizing Distributed Algorithms with Small Messages: Spanners and Dominating Set. In *Proceedings of the 32nd International Symposium on Distributed Computing (DISC)*. 29:1–29:17.

[27] Mohsen Ghaffari, Fabian Kuhn, and Jara Uitto. 2019. Conditional Hardness Results for Massively Parallel Computation from Distributed Lower Bounds. In *Proceedings of the 60th IEEE Symposium on Foundations of Computer Science (FOCS)*. 1650–1663.

[28] Mohsen Ghaffari and Jara Uitto. 2019. Sparsifying Distributed Algorithms with Ramifications in Massively Parallel Computation and Centralized Local Computation. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 1636–1653.

[29] Mark K. Goldberg and Thomas H. Spencer. 1989. A New Parallel Algorithm for the Maximal Independent Set Problem. *SIAM J. Comput.* 18, 2 (April 1989), 419–427.

[30] Michael T. Goodrich. 1999. Communication-Efficient Parallel Sorting. *SIAM J. Comput.* 29, 2 (Oct. 1999), 416–432.

[31] Michael T. Goodrich, Nodari Sitchinava, and Qin Zhang. 2011. Sorting, Searching, and Simulation in the MapReduce Framework. In *Proceedings of the 22nd International Symposium on Algorithms and Computation (ISAAC)*. 374–383.

[32] Yijie Han. 1996. A Fast Derandomization Scheme and Its Applications. *SIAM J. Comput.* 25, 1 (Jan. 1996), 52–82.

[33] David G. Harris. 2018. Derandomized Concentration Bounds for Polynomials, and Hypergraph Maximal Independent Set. In *Proceedings of the 39th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2161–2180.

[34] David G. Harris. 2019. Deterministic Parallel Algorithms for Bilinear Objective Functions. *Algorithmica* 81, 3 (March 2019), 1288–1318.

[35] Michael Isard, Mihai Budiu, Yuan Yu, Andrew Birrell, and Dennis Fetterly. 2007. Dryad: Distributed Data-parallel Programs from Sequential Building Blocks. *SIGOPS Operating Systems Review* 41, 3 (March 2007), 59–72.

[36] Amos Israeli and Alon Itai. 1986. A Fast and Simple Randomized Parallel Algorithm for Maximal Matching. *Inform. Process. Lett.* 22, 2 (Jan. 1986), 77–80.

[37] Howard J. Karloff, Siddharth Suri, and Sergei Vassilvitskii. 2010. A Model of Computation for MapReduce. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 938–948.

[38] Richard M. Karp and Avi Wigderson. 1985. A Fast Parallel Algorithm for the Maximal Independent Set Problem. *Journal of the ACM* 32, 4 (Oct. 1985), 762–773.

[39] Fabian Kuhn. 2009. Weak Graph Colorings: Distributed Algorithms and Applications. In *Proceedings of the 21st Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*. 138–144.

[40] Silvio Lattanzi, Benjamin Moseley, Siddharth Suri, and Sergei Vassilvitskii. 2011. Filtering: A Method for Solving Graph Problems in MapReduce. In *Proceedings of the 23rd Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*. 85–94.

[41] Jakub Łącki, Slobodan Mitrović, Krzysztof Onak, and Piotr Sankowski. 2020. Walking Randomly, Massively, and Efficiently. In *Proceedings of the 52nd Annual ACM Symposium on Theory of Computing (STOC)*. 364–377.

[42] Christoph Lenzen. 2013. Optimal Deterministic Routing and Sorting on the Congested Clique. In *Proceedings of the 32nd ACM Symposium on Principles of Distributed Computing (PODC)*. 42–50.

[43] Nathan Linial. 1992. Locality in Distributed Graph Algorithms. *SIAM J. Comput.* 21, 1 (Feb. 1992), 193–201.

[44] Zvi Lotker, Elan Pavlov, Boaz Patt-Shamir, and David Peleg. 2003. MST Construction in $O(\log\log n)$ Communication Rounds. In *Proceedings of the 15th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*. 94–100.

[45] Michael Luby. 1986. A Simple Parallel Algorithm for the Maximal Independent Set Problem. *SIAM J. Comput.* 15, 4 (Nov. 1986), 1036–1053.

[46] Krzysztof Onak. 2018. Round Compression for Parallel Graph Algorithms in Strongly Sublinear Space. *CoRR abs/1807.08745* (2018). arXiv:1807.08745

[47] Merav Parter. 2018. $(\Delta + 1)$ Coloring in the Congested Clique Model. In *Proceedings of the 45th Annual International Colloquium on Automata, Languages and Programming (ICALP)*. 160:1–160:14.

[48] Merav Parter and Eylon Yogev. 2018. Congested Clique Algorithms for Graph Spanners. In *Proceedings of the 32nd International Symposium on Distributed Computing (DISC)*. 40:1–40:18.

[49] David Peleg. 2000. *Distributed Computing: A Locality-Sensitive Approach.* SIAM, Philadelphia, PA.

[50] Tim Roughgarden, Sergei Vassilvitski, and Joshua R. Wang. 2018. Shuffles and Circuits (On Lower Bounds for Modern Parallel Computation). *Journal of the ACM* 65, 6 (Nov. 2018), 41:1–41:24.

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

[52] Eric Vigoda. 2006. Lecture Notes for Randomized Algorithms: Luby's Alg. for Maximal Independent Sets using Pairwise Independence. https://www.cc.gatech.edu/~vigoda/RandAlgs/MIS.pdf.

[53] Tom White. 2012. *Hadoop: The Definitive Guide.* O'Reilly Media, Inc.

[54] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. 2010. Spark: Cluster Computing with Working Sets. In *Proceedings of the 2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud).*