



Massively Parallel Ruling Set Made Deterministic

Jeff Giliberti  

University of Maryland, College Park, MD, USA

Zahra Parsaeian  

University of Freiburg, Germany

Abstract

We study the *deterministic complexity* of the 2-Ruling Set problem in the model of Massively Parallel Computation (MPC) with linear and strongly sublinear local memory.

Linear MPC: We present a constant-round deterministic algorithm for the 2-Ruling Set problem that matches the randomized round complexity recently settled by Cambus, Kuhn, Pai, and Uitto [DISC'23], and improves upon the deterministic $O(\log \log n)$ -round algorithm by Pai and Pemmaraju [PODC'22]. Our main ingredient is a simpler analysis of CKPU's algorithm based solely on bounded independence, which makes its efficient derandomization possible.

Sublinear MPC: We present a deterministic algorithm that computes a 2-Ruling Set in $\tilde{O}(\sqrt{\log n})$ rounds deterministically. Notably, this is the first deterministic ruling set algorithm with sublogarithmic round complexity, improving on the $O(\log \Delta + \log \log^* n)$ -round complexity that stems from the deterministic MIS algorithm of Czumaj, Davies, and Parter [TALG'21]. Our result is based on a simple and fast randomness-efficient construction that achieves the same sparsification as that of the randomized $\tilde{O}(\sqrt{\log n})$ -round LOCAL algorithm by Kothapalli and Pemmaraju [FSTTCS'12].

2012 ACM Subject Classification Theory of computation \rightarrow MapReduce algorithms

Keywords and phrases deterministic algorithms, distributed computing, massively parallel computation, graph algorithms, derandomization

Digital Object Identifier 10.4230/LIPIcs.DISC.2024.29

Related Version *Full Version:* <https://arxiv.org/abs/2406.12727>

Funding *Jeff Giliberti:* Financial support in part by the Fulbright U.S. Graduate Student Program, sponsored by the U.S. Department of State and the Italian-American Fulbright Commission. The content does not necessarily represent the views of the Program.

Acknowledgements We are grateful to Christoph Grunau and Manuela Fischer for valuable discussions. We would also like to thank the anonymous reviewers for their helpful feedback, and Yannic Maus for his shepherding of the paper.

1 Introduction

In this paper, we present faster deterministic parallel algorithms for finding 2-ruling sets. Given an n -vertex m -edge graph $G = (V, E)$ and an integer $\beta \geq 1$, the more general problem of β -ruling sets consists of finding a subset $S \subseteq V$ of non-adjacent vertices such that each vertex $v \in V \setminus S$ is at most β hops away from some vertex in S . Thus, a β -ruling set is also a $\beta + 1$ ruling set. This concept serves as a natural generalization of one of the most central and well-studied problems in distributed graph algorithms, known as *Maximal Independent Set* (MIS), which corresponds to a 1-ruling set. Generally, for $\beta \geq 1$, the complexity of a β -ruling set reduces as the value of β increases.

We design 2-ruling set algorithms for the model of Massively Parallel Computation (MPC) in the strongly sublinear and linear memory regimes. The study of 2-ruling sets is motivated by its close relationship with MIS, while still permitting the development of considerably faster algorithms. Additionally, it is known that for problems utilizing MIS as a subroutine, a β -ruling set may serve as an alternative for some $\beta > 1$ [4].



© Jeff Giliberti and Zahra Parsaeian;

licensed under Creative Commons License CC-BY 4.0

38th International Symposium on Distributed Computing (DISC 2024).

Editor: Dan Alistarh; Article No. 29; pp. 29:1–29:21



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

MPC Model. Initially introduced by [32] and later refined in [2, 7, 30], this model is characterized by a set of M machines each with memory S . The input is distributed across machines and the computation proceeds in synchronous rounds. Each round machines perform arbitrary local computation and all-to-all communication, sending and receiving up to S words. The main goal is to minimize the number of communication rounds required by the algorithm. A second goal is to minimize the global space needed to solve the problem, i.e., the number of machines times the local memory per machine, which is $\Omega(n + m)$ for graph problems. In the linear regime of MPC each machine is assigned local memory $S = O(n)$, while in the (strongly) sublinear regime of MPC the local memory is $O(n^\alpha)$, for constant $0 < \alpha < 1$.

Linear MPC. In the linear model of MPC, a series of works showed that several fundamental problems such as $(\Delta + 1)$ -coloring [13, 16] and minimum-spanning tree [42] admit constant-round deterministic algorithms. Surprisingly, a recent work of [11] provides a randomized 2-ruling set algorithm with constant-round complexity improving on the $O(\log \log \log n)$ time algorithm by [31] and the $O(\log \log \Delta)$ time bound that stems from the MIS algorithm by [26]. On the deterministic side, [43] gave an algorithm that computes a 2-ruling set in $O(\log \log n)$ time, which improved on the $O(\log \Delta + \log \log^* n)$ round complexity due to the deterministic MIS algorithm of [18, 17, 20]. Key challenges in this domain lie in determining the existence of deterministic algorithms achieving constant-round complexity for 2-ruling sets and sublogarithmic-round complexity for MIS.

Sublinear MPC. In the sublinear model of MPC, the above $O(\log \Delta + \log \log^* n)$ -round algorithm by [18, 17] is the fastest known for both MIS and 2-ruling set. On the randomized side, [28] show that MIS can be solved in $\tilde{O}(\sqrt{\log \Delta} + \log \log n)$ rounds and [43] show that 2-ruling set can be solved in $\tilde{O}(\log^{1/6} \Delta + \log \log n)$, where the $\tilde{O}(\cdot)$ notation hides poly $\log(\cdot)$ factors. It may be worth noting that if we limit the global space to $\tilde{O}(n + m)$, then the fastest 2-ruling set algorithm has $\tilde{O}(\log^{1/4} n + \log \log n)$ randomized complexity [43] and $O(\log \Delta \log \log n)$ deterministic complexity [18, 23].

Other Related Work. There is a large body of work studying ruling sets in the LOCAL model [24, 9, 31, 45, 10, 6]. The most relevant to ours is the randomized LOCAL algorithm of [35] for computing 2-ruling sets that combined with [25] yields a LOCAL round complexity of $\tilde{O}(\sqrt{\log n})$. On the hardness side, in the LOCAL model, there is a lower bound for 2-ruling set of $\Omega(\min\{\sqrt{\Delta}, \log_\Delta n\})$ deterministic rounds and of $\Omega(\min\{\sqrt{\Delta}, \log_\Delta \log n\})$ randomized rounds by [5, 4], which, in terms of its proportion to n , are $\Omega(\frac{\log n}{\log \log n})$, and $\Omega(\frac{\log \log n}{\log \log \log n})$, respectively. For MIS and maximal matching (MM), the best known deterministic lower bound is $\Omega(\min\{\Delta, \log_\Delta n\})$ by [3], and the best known randomized lower bounds are $\Omega(\min\{\Delta, \log_\Delta \log n\})$ by [3] and $\Omega(\min\{\frac{\log \Delta}{\log \log \Delta}, \log_\Delta n\})$ by [36], which, in terms of its proportion to n , are $\Omega(\frac{\log n}{\log \log n})$, $\Omega(\frac{\log \log n}{\log \log \log n})$, and $\Omega(\sqrt{\frac{\log n}{\log \log n}})$, respectively. Via the MPC conditional lower-bound framework by [27, 17], these results give the following component-stable lower bounds for sublinear MPC algorithms:

- $\Omega(\log \log n)$ for deterministic 2-ruling set, deterministic and randomized MIS and MM.
- $\Omega(\log \log \log n)$ for randomized 2-ruling set.

1.1 Our Contribution

We design improved deterministic algorithms for the problem of 2-ruling set in the MPC setting with linear and sublinear local memory.

Linear MPC Regime. We develop a deterministic algorithm that matches the constant-round complexity of [11] and even its optimal global space usage.

► **Theorem 1.** *There is a $O(1)$ -round linear MPC algorithm that computes a 2-ruling set deterministically using linear global space.*

Prior to our work, the best known deterministic complexity was $O(\log \log n)$ by a result of [43]. Our algorithm (Section 3) is obtained by derandomizing the $O(1)$ -round algorithm of [11]. While the derandomization framework of our algorithm has been applied successfully to numerous MPC graph problems [12, 16, 19, 18, 15, 22, 23, 43], the main challenge lies in analyzing (a slight variation of) [11]’s algorithm under limited independence, as we overview later in Section 1.2.1.

Sublinear MPC Regime. We design the first deterministic sublogarithmic algorithm for finding a 2-ruling set when the memory per machine is strictly sublinear.

► **Theorem 2.** *There is a deterministic sublinear MPC algorithm that finds a 2-ruling set in $O(\sqrt{\log \Delta} \cdot \log \log \Delta + \log \log^* n)$ rounds using $O(n^{1+\varepsilon} + m)$ global space, for any constant $\varepsilon > 0$. Moreover, the same algorithm runs in $O(\sqrt{\log \Delta} \cdot \log \log n)$ using global space $O(n + m)$.*

For $\Delta \gg \log^* n$, our algorithm gives an almost quadratic improvement over the runtime obtained using the MIS algorithm of [20], and gets closer to the $\tilde{O}(\log^{1/6} \Delta + \log \log n)$ randomized complexity of [33]. It is worth noting that it matches the conditionally-optimal runtime of $\Omega(\log \log n)$ when $\Delta = O(2^{\log^2 \log n / \log \log \log n})$, even though, being it not component-stable, the lower bound does not apply.

This algorithm (Section 4) is obtained by derandomizing the sparsification developed by [35] for solving 2-ruling sets in the LOCAL model. Specifically, we show that a randomized $O(1)$ -LOCAL downsampling step can be carried out in only $O(\log \log \Delta)$ rounds deterministically in MPC with strongly sublinear space per machine and optimal global space. To achieve that, we combine several well-established derandomization tools such as limited independence, the method of conditional expectation, and coloring for reducing seed length, as we discuss in Section 1.2.2.

We also note that our techniques may be more general and apply to β -ruling sets for $\beta > 2$. Concretely, one may combine our result with the framework of [10] to obtain faster MPC β -ruling sets algorithms. This direction is left for future work.

1.2 2-Ruling Sets: Technical Overview

We present the main intuition behind the recent constant-round randomized algorithm by [11] in the linear regime of MPC and the randomized $\tilde{O}(\sqrt{\log n})$ -round LOCAL algorithm by [35], which is also closely followed by subsequent works [31, 33, 43]. Then, we provide an overview of our deterministic algorithms and the main ideas that lead to randomness-efficient analyses.

1.2.1 Linear Memory Regime

Randomized Constant-Round Algorithm. The constant-round 2-ruling set algorithm by [11] relies on computing an MIS iteratively on subgraphs of linear size, which can be solved locally on a single machine. Their algorithm samples each vertex v from V and includes it in V_{samp} independently with probability $1/\sqrt{\deg(v)}$. This sampling primitive is shown to give two useful structural properties, with high probability. First, the induced subgraph $G[V_{\text{samp}}]$ has a linear number of edges. Second, a certain MIS computation on $G[V_{\text{samp}}]$ returns an independent set that is at distance at most two from all but at most n/\sqrt{d} vertices with degree $[d, 2d)$ in the original graph G , for each $d \in \{2^{\lfloor \log \Delta \rfloor}, 2^{\lfloor \log \Delta \rfloor - 1}, \dots, \Omega(1)\}$. Then, it is shown that, after two repetitions, the number of remaining edges for each degree class d is at most $n/\text{poly}(d)$, which sums up to $O(n)$ over all d 's.

Their analysis of the above sampling process relies on full independence in the sense that random decisions of any node influence its neighbors at distance at most three. Then, each node influences only up to $n^{3\alpha}$ many nodes by assuming that any node has degree at most n^α , for constant $\alpha > 0$. This property is exploited to union bound over large sets of independent nodes in G^7 , since nodes at distance 8 are enough far apart not to influence one another. Clearly, this property breaks apart under our constraint of limited independence and requires to analyze the sampling process differently.

Constant-Round Derandomization. In a nutshell, we show that the same asymptotic guarantees as that provided by the above randomized algorithm can be achieved deterministically. While it is easy to show that their initial sampling step gives a subgraph with a linear number of edges in expectation, even under pairwise independence, the main challenge is to prove that only $n/d^{\Omega(1)}$ nodes survive across all $O(\log \Delta)$ d -degree classes, simultaneously. Establishing the same polynomial decrease (in $d^{\Omega(1)}$) of the size of each d -degree class ensures the same constant-round complexity.

Our key modification to [11]'s analysis is to increase the threshold for a node to be called good. We say that a node of degree d is good if it has at least $d^{\Omega(1)}$ neighbors in $G[V_{\text{samp}}]$, as opposed to the $\Theta(\log n)$ requirement of [11]. This leads to the following two properties.

First, in the sampling step, we prove that each good node of degree d is covered with probability $1 - 1/\text{poly}(d)$ and that suffices. In fact, through the method of conditional expectation, non-covered nodes will induce at most $O(n)$ edges.

Second, in the MIS step, we prove that remaining “bad” nodes are at most $n/d^{\Omega(1)}$ for each degree class, without any assumption on the maximum degree. To achieve that, we combine a pairwise independent MIS algorithm (similar to that of [23]) with a pessimistic estimator that notably expresses the progress made over *all* degree classes as a *single* expectation. This expectation can then be obtained by means of standard derandomization tools.

1.2.2 Strongly Sublinear Memory Regime

Randomized 2-Ruling Set Sparsification. The central step of the 2-ruling set algorithms by [34, 33] is a sparsification procedure that returns a subgraph G' of sufficiently small maximum degree. Then, computing a maximal independent set on G' has time proportional to its maximum degree, and yields a 2-ruling set that covers all vertices in G which have a neighbor in G' .

They construct a subgraph G' of maximum degree $O(f \cdot \log n)$ such that any (high-degree) node with a degree in $[\Delta, \Delta/f]$ in G has a neighbor in G' , for some parameter $f \geq \log n$. It is easy to see that sampling each vertex $v \in V$ with probability $f \cdot \log n / \Delta$ independently ensures that every vertex with degree at least Δ/f will have a sampled vertex in its neighborhood with high probability.

We just focused solely on covering vertices with degrees in $[\Delta, \Delta/f]$. It turns out that, by each time removing the subgraph G' and its neighbors, the same sampling step can be repeated $O(\log_f \Delta)$ times, where in the j -th step nodes with degrees in $[f^{\log_f \Delta - (j-1)}, f^{\log_f \Delta - j}]$ are covered, with $j \in [\log_f \Delta]$. This simple process leads to a randomized round complexity of $O(\log f + \log_f \Delta + \text{poly log log } n)$ by applying any MIS algorithm that runs in $O(\log \Delta + \text{poly log log } n)$ rounds [25, 28] on the union of all subgraphs, which have no conflicts by construction. Then, $f = 2\sqrt{\log \Delta}$ is chosen to achieve a runtime of $O(\sqrt{\log \Delta} + \text{poly log log } n)$.

Deterministic 2-Ruling Set Sparsification. Our goal is to replace the above randomized sampling with a deterministic sampling that returns a subgraph G' with the same properties as those returned by the above construction [34, 33]. We slightly alter the sampling guarantees to allow for a relaxed maximum degree in G' of up to $\text{poly}(f)$ instead of $O(f \log n)$. Instead of sampling each vertex with probability $f \cdot \log n / \Delta$ randomly and independently in a single round, we sample them in a deterministic manner in $O(\log \log \Delta)$ rounds. The way in which we design this deterministic sampling step is explained next.

The standard approach is to limit the randomness by sampling vertices using a carefully selected k -wise independent hash function. A naive implementation that samples vertices with probability $\frac{\text{poly}(f)}{\Delta}$ would need a family of k -wise independent hash functions with $k = \Omega(\log_f n)$, since each vertex has $\text{poly}(f)$ expected sampled neighbors. The need for $\Omega(\log_f n)$ -wise independence results in a seed of length $\Omega(\log_f n \cdot \log \Delta)$. Since in $O(1)$ MPC rounds only $O(\log n)$ bits can be fixed, this one-step process appears to require $\Omega(\frac{\log \Delta}{\log f})$ many rounds¹, which is very far from being sublogarithmic.

Our approach to make this construction randomness-efficient relies on breaking down the sampling process into $O(\log \log \Delta)$ sub-sampling processes, each of which has weaker guarantees but requires only $O(1)$ rounds. In particular, the basis of our process is a simple, deterministic, constant-round routine that decreases the maximum degree by a $O(\sqrt{\Delta})$ -factor, while ensuring that the maximum-to-minimum degree ratio of $O(f)$ is maintained, i.e., each vertex v has degree roughly $|N_G(v)|/\sqrt{\Delta}$ in G' .

Then, we repeatedly apply this degree-reduction routine to sparsify the neighborhoods of high-degree vertices until their degree drops to $2^{O(\log f)}$. It is easy to see that this requires at most $O(\log \log \Delta)$ repetitions. However, in each iteration, some downsampled neighborhoods may deviate from their expectation, say by an ϵ -factor. Such deviation is amplified each time, resulting in a potential error of $\epsilon^{O(\log \log \Delta)}$. Nevertheless, through a suitable f and ϵ , we can minimize the error and show that the subgraph G' has $\text{poly}(f)$ maximum degree. Therefore, we can iterate through the $O(\log_f \Delta)$ degree classes (as in the randomized case) and apply our deterministic degree reduction to achieve the same result, up to a $O(\log \log \Delta)$ factor.

Further Comparison. Several sparsifications for MIS and 2-ruling sets in LOCAL and low-memory MPC have been studied. We include a brief comparison with the works of [18, 39, 33].

A deterministic $O(1)$ -round sampling process appeared in the MIS algorithm of [18]. There, the goal is to reduce the maximum degree to at most n^ϵ while ensuring that the resulting subgraph maintains enough edges and the distribution of degrees is still representative of the original graph. They decrease the maximum degree by an $n^{\Omega(1)}$ -factor for $O(1)$ times, until the desired bound is achieved. Since the expected new maximum degree is still on the order

¹ Here, shortening the seed length using a family of ϵ -approximate k -wise independent hash functions still requires $\omega(1)$ MPC rounds.

of $n^{\Omega(1)}$, concentration around the expectation can be achieved with $O(1)$ -wise independence, and thus derandomized in $O(1)$ rounds. In contrast, in 2-ruling set, the main challenge is to subsample the neighborhoods of nodes with degree $d \ll n^{\Omega(1)}$. In fact, applying a similar subsampling method would require $\Omega(\log_d n)$ -wise independence and $\Omega(\frac{\log \Delta}{\log f})$ rounds, as explained in the paragraph above. Thus, while the method in [18] is effective for high-degree nodes with $d = n^{\Omega(1)}$, handling smaller degrees requires a different approach.

The ruling set algorithm of [39] introduces a CONGEST sparsification that runs in $O(\log^2 n)$ rounds and deals with $O(\log \Delta)$ degree classes. There, a single sampling step requires a seed of length $O(\log^2 n)$ as they require guarantees stricter than ours. Specifically, their sparsification must maintain a low diameter and ensure proper coverage. Although their derandomization is CONGEST-efficient, it would require $O(\log n)$ MPC rounds, making it unsuitable to our setting.

Finally, we note that the faster randomized 2-ruling set algorithm of [33] relies on (informally) performing graph exponentiation on a sparsified subgraph. This approach relies on fixing the randomness of future iterations in advance, which simplifies the process of speeding up algorithms in LOCAL. The main challenge in adapting this approach to a deterministic setting is that existing techniques are generally effective at derandomizing only $O(1)$ steps of an algorithm. They do not easily extend to derandomize algorithms that simulate $\Omega(1)$ randomized rounds locally on each single machine via graph exponentiation. Consequently, achieving the same speed up deterministically appears to require a novel approach.

2 Preliminaries

In our analyses, we will use the notation $\text{poly}(\cdot)$ to refer to $(\cdot)^c$, for a constant $c > 0$ at the exponent that can be made arbitrarily large without affecting asymptotic bounds.

Primitives in MPC. We recall that basic computations can be performed in the MPC model with strongly sublinear local memory in $O(1)$ rounds deterministically [29, 30].

Therefore, tasks such as computing the degree of each vertex, ensuring neighborhoods of all vertices are stored on single machines, and collecting certain subgraphs onto a single machine will be used as black-box tools.

Derandomization Framework. A rich and successful line of research has studied the derandomization of algorithms in the parallel and distributed setting. In the MPC model, classic derandomization schemes using limited independence and the method of conditional expectation [38, 41], can be augmented with the power of local computation and global communication to achieve the expected result in $O(1)$ rounds.

We will often use the concepts of k -wise independence and family of k -wise independent hash functions (see, e.g., [40, 44]). Given a randomized process that works under k -wise independence, it is known how to construct a k -wise independent family of hash functions.

► **Lemma 3** ([1, 14, 21]). *For every $N, k, \ell \in \mathbb{N}$, there is a family of k -wise independent hash functions $\mathcal{H} = \{h : [N] \rightarrow \{0, 1\}^\ell\}$ such that choosing a uniformly random function h from \mathcal{H} takes at most $k(\ell + \log N) + O(1)$ random bits, and evaluating a function from \mathcal{H} takes time $\text{poly}(\ell, \log N)$ time.*

Moreover, to show concentration around the expected value under k -wise independence, we will use the following tail bound.

► **Lemma 4** (Lemma 2.3 of [8]). *Let $k \geq 4$ be an even integer. Let X_1, \dots, X_n be random variables taking values in $[0, 1]$. Let $X = X_1 + \dots + X_n$ denote their sum and let $\mu \leq \mathbb{E}[X]$ satisfying $\mu \geq k$. Then, for any $\epsilon > 0$, we have*

$$\Pr [|X - \mathbb{E}[X]| \geq \epsilon \cdot \mathbb{E}[X]] \leq 8 \left(\frac{2k}{\epsilon^2 \mu} \right)^{k/2}.$$

We consider randomized algorithms that succeed in expectation when their random choices are made using a family of k -wise independent hash functions \mathcal{H} . Once our algorithm (randomly) picks a hash function h , then all choices are made deterministically according to h . Thus, our problem is that of deterministically finding a hash function that achieves a result as good as the expectation.

The by-now standard MPC derandomization process can be broken down into two parts: (i) show that the family of hash functions \mathcal{H} has size $\text{poly}(n)$ and produces the desired result in expectation, and (ii) find one good hash function by applying the method of conditional expectation in a distributed fashion. We will focus on establishing (i), since (ii) can then be achieved by known MPC derandomization methods introduced by earlier works [12, 15, 18] to which we refer for further details. It is worth mentioning that for step (ii) to be solved using earlier tools as a black-box, the aimed expectation should be expressed as a sum of locally computable quantities by each individual machine, i.e., the individual expectation of each node that a machine stores.

3 Deterministic 2-Ruling Set in Linear MPC

We first introduce the reader to several sets of nodes that play a crucial role in our algorithm. These sets of nodes are defined to reflect how a node will be handled by our algorithm. Specifically, the core of the algorithm is a downsampling procedure that outputs a sufficiently small subgraph on which we will compute a maximal independent set with the goal of *ruling* a large fraction of nodes in the original graph.

Observe that if a node has a neighbor in the downsampled graph, then it will have some node in the maximal independent set at distance at most two. This means that if a node is likely to have a sampled neighbor, then it is likely to be ruled, and we call such a node *good*. In the following, our definitions and algorithm are parameterized by a constant $\epsilon = 1/40$, which has not been optimized.

► **Definition 5** (Good Node). *A node $v \in G$ is good if it satisfies $\sum_{u \in N(v)} \frac{1}{\sqrt{\deg(u)}} \geq \deg(v)^\epsilon$.*

If a node v is not good, i.e., $\sum_{u \in N(v)} \frac{1}{\sqrt{\deg(u)}} < \deg(v)^\epsilon$, then we say that v is a *bad* node. Bad nodes are split into $O(\log \Delta)$ degree classes as follows. Let d_0 be a sufficiently large constant and $d_{\max} = \lceil \log \Delta \rceil$.

► **Definition 6** (Bad Node Classes). *For $d \in \{2^{d_0}, 2^{d_0+1}, \dots, 2^{d_{\max}}\}$, the set B_d includes all bad nodes with degree in $[d, 2d)$.*

Therefore, bad nodes are likely to have few sampled nodes. This fact motivates the following observation. If a (bad) node has *many* bad nodes within its 2-hop neighborhood, then it is likely that at least one of such bad ones is in the maximal independent set. If that is the case, we call such nodes *lucky* bad nodes, as specified in the following definition.

► **Definition 7** (Lucky Bad Nodes). *For $d \in \{2^{d_0}, 2^{d_0+1}, \dots, 2^{d_{\max}}\}$, the set $\bar{B}_d \subseteq B_d$ includes each node $u \in B_d$ such that u has a neighbor w with $|N(w) \cap B_d| \geq 6d^{0.6}$. If there are multiple such w 's, pick one arbitrarily and let S_u be an arbitrarily chosen subset of $N(w) \cap B_d$ such that $|S_u| = 6d^{0.6}$.*

With these definitions in mind, we are now ready to present our deterministic constant-round 2-Ruling Set algorithm in the linear regime of MPC.

The algorithm operates in three simple steps: Sampling, Gathering, and MIS Computation. The first step of the algorithm samples each node v with probability $\deg^{-1/2}(v)$. The sampling probability is chosen to ensure that the downsampled graph has a linear number of edges. Moreover, we will slightly alter the downsampled graph to include all nodes that do not satisfy certain requirements, without affecting the asymptotic size of this subgraph. Therefore, in the second step, we will be able to collect such subgraph onto a single machine. Then, the MIS computation begins by running one iteration of Luby's MIS on (part of) the subgraph from the previous step and continues by extending such independent set to a maximal one locally.

We will prove several desirable properties about the three-step algorithm above that lead to a reduction of a $d^{\Omega(1)}$ -factor for each degree class d . Therefore, by repeating this three-step algorithm $O(1)$ times, the number of edges over all degree classes converges to $O(n)$ and thus can be collected and solved locally, completing the proof of Theorem 1.

Next, we present the algorithm in more detail and then proceed to analyzing its three steps with a particular focus on randomness efficiency. In fact, such randomness-efficient analyses will allow for a simple derandomization.

3.1 The Algorithm

Sampling Step. Let $G = (V, E)$ be the input graph with n vertices and m edges. Let V_{samp} denote the set of sampled vertices. We include each vertex $v \in V$ in V_{samp} with probability $p_v = \frac{1}{\sqrt{\deg(v)}}$, according to a family of k -wise independent random variables with $k = O(1)$.

Gathering Step. We gather several subsets of nodes whose (combined) induced subgraph will be shown to have a linear number of edges. Gathered nodes are those either sampled in the previous step or not satisfying certain properties as formally defined below. Let V^* denote the union of the following node subsets, which are being gathered locally onto a single machine:

1. The set of sampled nodes V_{samp} ;
2. Every good node that is *not sampled* and has *no sampled neighbors*;
3. For each d , every lucky bad node $u \in \overline{B_d}$ that has either less than $d^{0.1}$ sampled nodes in S_u or one of the sampled nodes in S_u has more than $d^{2\epsilon}$ sampled neighbors; as formalized in Lemma 10.

MIS Computation. Our goal is now to compute a maximal independent set on the locally gathered subgraph $G[V^*]$ to rule all but roughly at most a $\Delta^{\Omega(1)}$ -fraction of nodes in G . We achieve this by first computing a partial MIS on the sampled bad vertices, i.e., $\bigcup_d B_d \cap V_{\text{samp}}$, using a variation of Luby's algorithm as detailed in the proof of Lemma 12. Afterward, we can simply compute an MIS locally (and thus sequentially) on the remaining vertices, which are not incident to the partial MIS computed earlier.

Output Properties. We expect that the output given by the *derandomization* of the above three-step process satisfies the following properties. We will later use these properties to achieve a deterministic constant-round complexity. Observe that we can ignore constant-degree nodes since they can be gathered and dealt with locally at last.

- **Good nodes:** All good nodes in G are ruled after the MIS step.

- **Uncovered lucky bad nodes:** For each d , after the computation of a partial MIS, only a $d^{\Omega(1)}$ -fraction of lucky bad nodes remains uncovered.
- **Uncovered bad nodes:** For each d , the number of bad nodes in $B_d \setminus \bar{B}_d$ is only a $d^{\Omega(1)}$ -fraction of all nodes with initial degree at least d in G .

3.2 Analysis

We first establish that good nodes are likely to have a neighbor in V_{samp} . Since we will compute an MIS on $V^* \supseteq V_{\text{samp}}$, such good nodes will be at distance at most 2 from a node in the MIS. Moreover, good nodes that have no sampled neighbor will be shown to be incident to a linear number of edges, allowing us to gather them as part of V^* .

► **Lemma 8.** *Every good vertex v has a neighbor in V_{samp} with probability at least $1 - \frac{1}{\text{poly}(\deg(v))}$.*

Proof. For any vertex u , let X_u be the indicator random variable for the event $u \in V_{\text{samp}}$, and X be the random number of neighbors of v in V_{samp} . Further, let $\mu := \mathbb{E}[X] = \sum_{u \in N(v)} \mathbb{E}[X_u] = \sum_{u \in N(v)} \Pr[X_u = 1] \geq \deg(v)^\varepsilon \gg k$, since nodes of constant degree can be ignored and dealt with separately at last by collecting them onto a single machine. By applying Lemma 4, we have

$$\Pr[X = 0] \leq \Pr[|X - \mu| \geq \mu] \leq 8 \cdot \left(\frac{k\mu + k^2}{\mu^2} \right)^{k/2} \leq 8 \cdot \left(\frac{2k}{\mu} \right)^{k/2} = \frac{1}{\text{poly}(\deg(v))},$$

which proves the lemma. ◀

Toward the goal of ruling lucky bad nodes, we next show that bad nodes are likely to have few sampled neighbors. This means that sampled bad nodes, by having a low degree in the sampled graph, will have higher chances of being in the partial MIS computed later.

► **Lemma 9.** *Any node $u \in B_d$ has at most $d^{2\varepsilon}$ sampled neighbors with probability at least $1 - \frac{1}{\text{poly}(d)}$.*

Proof. Recall that for any $u \in B_d$, it holds that $\sum_{w \in N(u)} \frac{1}{\sqrt{\deg(w)}} < \deg(u)^\varepsilon$. We will use this fact to prove that the number of sampled neighbors does not deviate by more than $O(d^{2\varepsilon})$ with probability at least $1 - \frac{1}{\text{poly}(d)}$. Let X_w be the indicator random variable for the event $w \in V_{\text{samp}}$, and X be the random number of neighbors of u in V_{samp} . Let $\mu = \mathbb{E}[X] = \sum_{w \in N(u)} \mathbb{E}[X_w] = \sum_{w \in N(u)} \Pr[X_w = 1] < \deg(u)^\varepsilon < 2d^\varepsilon$. By applying Lemma 4, we get

$$\Pr[|X - \mu| \geq d^{2\varepsilon} - \mu] \leq 8 \cdot \left(\frac{k^2 + k\mu}{(d^{2\varepsilon} - \mu)^2} \right)^{k/2} \leq 8 \cdot \left(\frac{2k^2}{d^\varepsilon} \right)^{k/2} = \frac{1}{\text{poly}(d)}.$$

Note that for small values of d , our constant d_0 can be chosen such that $2^{d_0 \cdot \varepsilon} = \Omega(k^2)$. ◀

The next lemma proves that each lucky bad node u has a large number of nodes sampled out of its set S_u . Specifically, we need to show that the number of sampled nodes in S_u is higher than the degree of such nodes in the sampled graph. This fact will be used to ensure that lucky bad nodes have a vertex, within their 2-hop neighborhoods, in the MIS, thereby, ensuring their coverage.

► **Lemma 10.** *For any lucky bad node u , its set $S_u \subseteq B_d$ of cardinality $6d^{0.6}$ contains at least $d^{0.1}$ sampled nodes and each sampled node in S_u has at most $d^{2\varepsilon}$ sampled neighbors with probability at least $1 - \frac{1}{\text{poly}(d)}$.*

29:10 Massively Parallel Ruling Set Made Deterministic

Proof. By Lemma 9 and a union bound over the set S_u of $6d^{0.6}$ nodes, none of them has more than $d^{2\varepsilon}$ sampled neighbors with probability at least $1 - \frac{1}{\text{poly}(d)}$. Our goal is now to prove that the number of sampled vertices within S_u is less than $d^{0.1}$ with probability at most $\frac{1}{\text{poly}(\deg(u))} = \frac{1}{\text{poly}(d)}$.

Let X be the random number of sampled vertices in S_u , and let $\mu = \mathbb{E}[X] \geq 3d^{0.1}$, since each vertex in B_d is sampled with probability at least $1/\sqrt{2d}$. By applying Lemma 4, the probability of X deviating by more than $d^{0.1}$ from its expected value is

$$\Pr[|X - \mu| \geq \mu - d^{0.1}] \leq 8 \cdot \left(\frac{2k\mu}{(\mu - d^{0.1})^2} \right)^{k/2} \leq 8 \cdot \left(\frac{2k}{d^{0.1}} \right)^{k/2} = \frac{1}{\text{poly}(\deg(u))}. \quad \blacktriangleleft$$

We now use the above lemmas, together with a bound on the number of edges induced by the sampling step, to prove that our gathering step effectively collects $O(n)$ edges.

► **Lemma 11.** *The subgraph induced by $G[V^*]$ has $O(n)$ edges in expectation.*

Proof. Our goal is to prove that the expected sum of the original degrees of nodes in V^* is $O(n)$, which clearly upper bounds the number of edges in the induced subgraph. To do so, we analyze each subset individually.

We first analyze the expected number of edges induced by V_{samp} . Let X denote the random number of edges within the subgraph $G[V_{\text{samp}}]$. Let Y_e be an indicator random variable for the event that edge e is in $G[V_{\text{samp}}]$. To aid our analysis, we orient each edge in the graph from the endpoint with lower degree to the endpoint with higher degree. Now, consider an edge $e = (u, v)$ with $\deg(u) \leq \deg(v)$. Vertices u and v are each sampled with probability at most $\frac{1}{\sqrt{\deg(u)}}$. By pairwise independence, the probability of edge e being in $G[V_{\text{samp}}]$ is bounded by $\frac{1}{\deg(u)}$. Consequently, the expected number of edges is $\mathbb{E}[X] = \sum_{v \in V} \sum_{e \in \text{out}(v)} \mathbb{E}[Y_e] \leq \sum_{v \in V} \sum_{e \in \text{out}(v)} \frac{1}{\deg(u)} = O(n)$.

Next, let \bar{V}_{good} denote the set of good nodes that have no sampled neighbor and Y the random number of edges incident to \bar{V}_{good} in G . By Lemma 8, each good node v is in \bar{V}_{good} with probability at most $1/\text{poly}(\deg(v))$. Thus,

$$\mathbb{E}[Y] \leq \sum_{v \in V} \deg(v) \cdot \Pr[v \in \bar{V}_{\text{good}}] \leq \sum_{v \in V} \frac{\deg(v)}{\text{poly}(\deg(v))} = O(n).$$

Finally, let the set $B'_d \subseteq \bar{B}_d$ include each unlucky bad node u such that either less than $d^{0.1}$ vertices in S_u are sampled or any sampled node in S_u has more than $2d^\varepsilon$ sampled neighbors. By Lemma 10, each node u is in B'_d with probability at most $1/\text{poly}(d)$. Let Z be the random number of edges incident to B'_d . We have

$$\mathbb{E}[Z] \leq \sum_{i=d_0}^{d_{\max}} \sum_{u \in B_{2^i}} \deg(u) \cdot \Pr[u \in B'_{2^i}] \leq \sum_{i=d_0}^{d_{\max}} \sum_{u \in B_{2^i}} \frac{2d}{\text{poly}(d)} \leq \sum_{i=d_0}^{d_{\max}} |B_{2^i}| = O(n). \quad \blacktriangleleft$$

Derandomize Sampling and Gathering Steps. We are now ready to discuss how the above Sampling and Gathering steps can be turned into a deterministic linear MPC algorithm. Recall that each vertex is sampled according to a family of k -wise independent random variables with $k = O(1)$. A family \mathcal{H} of k -wise independent hash functions such that $h \in \mathcal{H} : [n] \rightarrow [n^3]$ can be specified using a random seed of length $O(\log n)$, meaning that $|\mathcal{H}| = \text{poly}(n)$. Each h maps the n vertex IDs (assumed to be from 1 up to n) to an integer in $[n^3]$. Then, each vertex is sampled and belongs to V_{samp} iff its ID is mapped to an

integer that is at most $\lfloor n^3/\sqrt{\deg(v)} \rfloor$ with respect to h , where the floor affects results only asymptotically. Each vertex can now locally check whether it will be included in V^* for a specified hash function h . In fact, the machine that v is assigned to stores all v 's neighbors and the set S_v if v is a lucky bad node. Therefore, it is easy to see that each node can compute the objective function $|E(G[V^*])|$ locally, and we can thus apply the distributed method of conditional expectation. Since $|\mathcal{H}| = \text{poly}(n)$, after a constant number of rounds we will find a h that ensures $|E(G[V^*])| = O(n)$.

We now turn to analyzing the MIS step. Recall that we first compute a partial MIS on the sampled bad nodes in order to rule all but a small fraction of lucky bad nodes. The next lemma explains how such an independent set is being computed.

► **Lemma 12.** *Let \hat{B}_d include each node $u \in \overline{B}_d$ that satisfies the property of Lemma 10. After the partial MIS computation, each node $u \in \hat{B}_d$ will be ruled with probability at least $1 - \frac{45}{d^\varepsilon}$ for all $d \in [d_0, d_{\max}]$. This result depends only on the randomness used in the MIS computation.*

The proof of Lemma 12 is provided in Appendix A.

The above lemma turns out not to be sufficient to derandomize our MIS step. In fact, we need to show that all degree classes of lucky bad nodes have a high enough chance of being ruled *simultaneously*. This is due to the fact that in the derandomization process, we can control only *one* objective function and not $O(\log \Delta)$ as the number of degree classes would appear to require. In the next lemma, we show how to define a pessimistic estimator that solves this issue.

► **Lemma 13.** *After the partial MIS computation, all but at most $\frac{|\overline{B}_d|}{d^{\Omega(1)}}$ nodes will be ruled in expectation, for all d simultaneously.*

Proof. Let us first reason about a fixed d and then about all d 's simultaneously.

Recall that \hat{B}_d include each node $u \in \overline{B}_d$ that satisfies the property of Lemma 10. There are at most $\frac{|\overline{B}_d|}{\text{poly}(d)}$ vertices in $\overline{B}_d \setminus \hat{B}_d$ by Lemma 10. Then, any vertex in \hat{B}_d is ruled with probability at least $1 - \frac{45}{d^\varepsilon}$ by Lemma 12. Therefore, by linearity of expectation, the number of non-ruled vertices in is at most $45|\overline{B}_d|/d^\varepsilon$.

Our goal is now to define a *single* objective function whose expected value ensures that the same asymptotic result holds for *all* d simultaneously. Let X_d be the random number of unruled nodes in \overline{B}_d , for each d . We define our objective function Q , which will serve as a ‘‘pessimistic estimator’’, as a weighted sum of the X_d 's as follows.

$$Q = \sum_{i=d_0}^{d_{\max}} X_{2^i} \cdot \frac{2^{i \cdot \frac{\varepsilon}{2}}}{|B_{2^i}|},$$

so that we get

$$\mathbb{E}[Q] = \sum_{i=d_0}^{d_{\max}} \mathbb{E}[X_{2^i}] \cdot \frac{2^{i \cdot \frac{\varepsilon}{2}}}{|B_{2^i}|} \leq \sum_{i=d_0}^{d_{\max}} \frac{45|\overline{B}_{2^i}|}{2^{i\varepsilon}} \cdot \frac{2^{i \cdot \frac{\varepsilon}{2}}}{|B_{2^i}|} = \sum_{i=d_0}^{d_{\max}} \frac{45}{2^{i\varepsilon/2}} = O(1),$$

where the convergency follows from choosing a sufficiently large constant $d_0 = O(\varepsilon^{-1})$. Observe that the expected value of Q ensures that, for each set \overline{B}_d , the number of nodes which are not ruled after running our Luby's step is $X_d \leq \mathbb{E}[Q] \cdot \frac{|\overline{B}_d|}{d^{\varepsilon/2}} = \frac{|\overline{B}_d|}{d^{\Omega(1)}}$. ◀

29:12 Massively Parallel Ruling Set Made Deterministic

Deterministic MIS Step. We now present an efficient derandomization of the above partial MIS computation in the linear MPC regime. As discussed in Lemma 12, our family \mathcal{H} of pairwise independent hash functions has size $|\mathcal{H}| = \text{poly}(n)$. Note that each lucky bad node u can store in a single machine its set S_u and all of their sampled neighbors since $|S_u| \cdot d^{2\varepsilon} = O(d) = O(\text{deg}(u))$. Then, each vertex u can check whether it will be ruled under a specified hash function h . Therefore, we can compute u 's contribution to $Q(h)$ locally, where $Q(h)$ is the objective function of Lemma 13 under a specified hash function h . This allows us to apply the distributed method of conditional expectation with objective Q to find a good hash function with $Q(h) = O(1)$ in a constant number of rounds.

Counting the bad nodes. Let $V_{\geq d}$ denote the set of all nodes in G with initial degree at least d , and let the set $B_d^* \stackrel{\text{def}}{=} B_d \setminus \overline{B}_d$. It remains to prove that the set B_d^* contains only a small fraction of nodes. The next lemma is equivalent to Lemma 9 of [11] up to some parameters change.

► **Lemma 14.** *For any degree $d \in [2^{d_0}, 2^{d_{\max}}]$, we have that $|B_d^*| \leq 12|V_{\geq d}|/d^{0.4}$.*

Proof. For a bad node v , it is easy to see by contradiction that at least $d/2$ of v 's neighbors have degree at least $d^{2(1-\varepsilon)}/4$ (see also Lemma 8 of [11]). Let $d' = \frac{d^{2(1-\varepsilon)}}{4}$. Therefore, any node $v \in B_d^*$ has at least $d/2$ neighbors in $V_{\geq d'}$. Furthermore, any node in $V_{\geq d'}$ neighboring a node in B_d^* has at most $6d^{0.6}$ edges connecting to nodes in $B_d \supseteq B_d^*$. As a result of these observations, we derive the following inequality:

$$d/2 \cdot |B_d^*| \leq 6|V_{\geq d'}| \cdot d^{0.6},$$

which together with the fact that $d' \geq d$, for d large enough, proves the lemma. ◀

Bounding Total Runtime. In the above paragraphs, we showed how to achieve deterministically the properties required by our three-step algorithm outlined at the beginning of this section. We now prove that repeating this process $O(1)$ times reduces the size of the graph to $O(n/\Delta)$, implying that the remaining nodes can be collected and solved for locally.

► **Lemma 15.** *At the end of the first iteration, the number of remaining uncovered vertices with degree at least d , denoted by $V_{\geq d}^{(1)}$, satisfies*

$$|V_{\geq d}^{(1)}| \leq |V_{\geq d}|/d^{\varepsilon'}.$$

Proof. The remaining uncovered vertices are only bad nodes. An uncovered bad node of degree $[d, 2d)$ can be either in B_d^* (Lemma 14) or remained uncovered after running the deterministic MIS step (Lemma 13). Over all $d, \dots, 2^{d_{\max}}$, this leads to:

$$|V_{\geq d}^{(1)}| \leq \sum_{i=\log d}^{d_{\max}} |B_{2^i}^*| + \frac{|\overline{B}_d|}{2^{\Omega(i)}} \leq \sum_{i=\log d}^{d_{\max}} \frac{12|V_{\geq 2^i}|}{2^{0.4 \cdot i}} + \frac{|\overline{B}_d|}{2^{\Omega(i)}} \leq |V_{\geq d}| \sum_{i=\log d}^{d_{\max}} \frac{1}{2^{\Omega(i)}} = \frac{|V_{\geq d}|}{d^{\Omega(1)}},$$

where the last inequality follows from $|\overline{B}_d| \leq |V_{\geq d}|$, and the final bound is due to the geometric sum being asymptotically dominated by the first term. ◀

Having established, in Lemma 15, the progress made at each iteration by our three-step process, we can now apply a simple induction to show the desired bound on the progress made after several iterations.

► **Lemma 16.** *After $O(1)$ iterations, the graph induced by uncovered nodes has $O(n)$ edges.*

Proof. Let $V_{\geq d}^{(k)}$ denote the number of remaining uncovered vertices with degree at least d at iteration k . Our goal is to prove that after k iterations, it holds that $V_{\geq d}^{(k)} \leq V_{\geq d}/d^{k\varepsilon'}$ so that for $k = O(1/\varepsilon')$, we get $V_{\geq d}^{(k)} \leq V_{\geq d}/d^{1.1}$. The base case for $k = 1$ follows from Lemma 15. Now, let us assume that $V_{\geq d}^{(k-1)} \leq V_{\geq d}/d^{(k-1)\varepsilon'}$. By a straightforward application of Lemma 15, we have that $V_{\geq d}^{(k)} \leq |V_{\geq d}^{(k-1)}|/d^{\varepsilon'} \leq V_{\geq d}/d^{k\varepsilon'}$, as desired. Now, since the number of nodes with degree $[d, 2d)$ is upper bounded by $|V_{\geq d}|$, the total number of edges is at most $\sum_{i=\log d_0}^{\log d_{\max}} V_{\geq d} \cdot 2^{i+1-1.1 \cdot i} = \sum_{i=\log d_0}^{\log d_{\max}} O(n/2^{0.1 \cdot i}) = O(n)$. ◀

4 Deterministic 2-Ruling Set in Sublinear MPC

In this section, we show that for an input graph with maximum degree Δ , a 2-ruling set can be computed deterministically in the strongly sublinear memory regime of MPC in $\tilde{O}(\log^{1/2} n)$ rounds.

We start by introducing a simple, deterministic, constant-round routine that reduces the size of each high-degree neighborhood by a $\sqrt{\Delta}$ -factor, where high-degree refers to node with degree at least $\log(n) \cdot \Delta^{0.6}$. For ease of exposition, assume that high-degree vertices form a set U , and that V is the set of all vertices (including high-degree vertices) that are being downsampled. Therefore, we reason about a bipartite graph $G = (U \sqcup V, E)$, where each node in $u \in U$ is connected to each vertex $v \in N_G(u)$ in the other part. Our goal is to ensure that each vertex u has roughly $|N_G(u)|/\sqrt{\Delta}$ neighbors deterministically. For simplicity, in the next lemma, we make two assumptions: (i) the neighbors of each vertex fit into a single machine, and defer the other case to Lemma 18; (ii) we are given a certain coloring of G that we discuss how to achieve at the end of this section.

► **Lemma 17.** *Let G be a graph with bipartition $V(G) = U \sqcup V$ and Δ be an upper bound on the maximum degree of any node in U such that $\Delta \in O(n^\alpha)$ for some $\alpha < 1$. Furthermore, assume that each node in V is given a color out of a palette of $O(\Delta^6)$ colors, such that any two distinct nodes $v, v' \in V$ that have a common neighbor in U are assigned distinct colors. Then, there exists a deterministic constant-round sublinear MPC algorithm that computes a subset $V^{sub} \subseteq V$ such that for any node $u \in U$ with $\deg_G(u) \geq \log(n) \cdot \Delta^{0.6}$, it holds that $|N_G(u) \cap V^{sub}| \in \left[\frac{1}{3\sqrt{\Delta}} |N_G(u)|, \frac{1}{\sqrt{\Delta}} |N_G(u)| \right]$. The global space usage is linear in the input size.*

Proof. Let us assume that each node $v \in V$ knows its own color c_v of a coloring satisfying the above properties. Then, nodes in V apply a hash function h from a k -wise independent family \mathcal{H} that maps each color to an integer in $[\lceil 3\sqrt{\Delta}/2 \rceil]$. A node v is then sampled under h iff $h(c_v) = 1$, which occurs with probability $1/\lceil 3\sqrt{\Delta}/2 \rceil$, where the ceil affects our results only asymptotically. We choose $k = 4c \log_\Delta n$, for constant $c > 0$, so that the seed length to select a hash function from \mathcal{H} is at most $\ell = O(\log_\Delta n) \cdot \max\{O(\log \Delta^6), O(\log \sqrt{\Delta})\} = O(\log n)$, i.e., the family \mathcal{H} has size $\text{poly}(n)$.

We prove that for each vertex $u \in U$ with degree larger than $\log n \cdot \Delta^{0.6}$, the probability of having between $\frac{1}{3\sqrt{\Delta}} |N(u)|$ and $|N(u)|/\sqrt{\Delta}$ neighbors within V^{sub} is at least $1 - \frac{1}{n^\varepsilon}$, i.e., the count of v 's neighbors in V^{sub} deviates by at most $\frac{1}{3\sqrt{\Delta}} |N(u)|$. For each neighbor v of u , let X_v be an indicator random variable for the event $v \in V^{sub}$. Define $X = \sum_{v \in N(u)} X_v$ as the number of neighbors of u in V^{sub} . Then, $\mu = \mathbb{E}[X] = \frac{2|N(u)|}{3\sqrt{\Delta}} \geq c \log n \Delta^{0.1}$. By applying

29:14 Massively Parallel Ruling Set Made Deterministic

Lemma 4, we have:

$$\begin{aligned} \Pr[|X - \mu| \geq \mu/2] &\leq 8 \left(\frac{4k\mu + 4k^2}{\mu^2} \right)^{k/2} \leq 8 \left(\frac{16c^2 \Delta^{0.1} \log^2 n + 32c^2 \log^2 n}{\Delta^{0.2} c^2 \log^2 n} \right)^{k/2} \\ &\leq 8 \left(\frac{1}{\Delta^{0.1}} \right)^{\frac{4c \cdot \log n}{2 \cdot \log \Delta}} \leq \frac{1}{n^{2c}}. \end{aligned}$$

Therefore, the expected number of high-degree vertices in U whose count of sampled neighbors deviates by more than $\mu/2$ is at most $n^{2c-1} < 1$. This means that we can apply the method of conditional expectation in a distributed fashion with as objective function the number of bad nodes, i.e., those whose sampled neighborhood deviates from the expectation by more than half. Since the memory capacity of each machine is $O(n^\alpha)$, each machine can compute locally the contribution to the objective of all the vertices (and their neighbors) it stores. Therefore, after $O(1)$ rounds, we find a hash function such that *all high-degree vertices* in U have the desired number of sampled neighbors. \blacktriangleleft

Next, we discuss how to extend Lemma 17 to handle the case in which not all neighbors of a vertex in U can be collected onto a single machine. In particular, if $\Delta \gg n^\alpha$, then aiming for a reduction of a $\sqrt{\Delta}$ -factor might not be viable, given the constrained local memory. Due to that, we slightly relax our goal and reduce our high-degree neighborhoods by a n^ε -factor, for some constant $\varepsilon < \alpha$. To achieve that, we split edges into groups so that each machine is assigned $n^{c-\varepsilon}$ edges, for $c > 1$. While we can only control the deviation of each single group of edges, we will be able to bound the overall number of neighbors, i.e., edges per node, using the fact that there are at most $\Delta/n^{c-\varepsilon}$ groups.

► Lemma 18. *Let G be a graph with bipartition $V(G) = U \sqcup V$. Let Δ be an upper bound on the maximum degree of any node in U such that $\Delta \geq n^{10\varepsilon}$, for some constant $\varepsilon > 0$. Then, there exists a deterministic constant-round sublinear MPC algorithm that computes a subset $V^{sub} \subseteq V$ such that for any node $u \in U$ with $\deg_G(u) \geq \log(n) \cdot \Delta^{0.6}$, it holds that $|N_G(u) \cap V^{sub}| \in \left[\frac{1}{2n^\varepsilon} |N_G(u)|, \frac{3}{2n^\varepsilon} |N_G(u)| \right]$. The global space usage is linear in the input size.*

Proof. Consider an arbitrary vertex $u \in U$ with degree at least $\log(n) \cdot \Delta^{0.6}$. The idea is to split edges of u into groups of size at most $n^{4\varepsilon}$, which fits into the memory of one machine. Specifically, each machine holds $n^{4\varepsilon}$ edges except for a single machine that holds any remaining edges, which are at most $n^{4\varepsilon}$. Then, we sample nodes in V with probability $n^{-\varepsilon}$ according to a family of $O(1)$ -wise independent hash function. Using a calculation similar to that of Lemma 17, we can find a hash function such that all groups of $n^{4\varepsilon}$ edges have $n^{3\varepsilon} \pm n^{2\varepsilon}$ sampled edges. Then, the total number of sampled neighbors is at least

$$\sum_{\text{machine } i} n^{3\varepsilon} - n^{2\varepsilon} \geq \left\lfloor \frac{|N_G(u)|}{n^{4\varepsilon}} \right\rfloor \cdot (n^{3\varepsilon} - n^{2\varepsilon}) \geq \frac{|N_G(u)|}{n^\varepsilon} - \frac{|N_G(u)|}{n^{2\varepsilon}} - n^{3\varepsilon} \geq \frac{|N_G(u)|}{2n^\varepsilon},$$

where $n^{3\varepsilon} = o\left(\frac{|N_G(u)|}{2n^\varepsilon}\right)$ since $N_G(u) \geq n^{6\varepsilon}$. An analogous calculation shows that the total number of sampled neighbors for any vertex u is at most $\frac{3|N_G(u)|}{2n^\varepsilon}$. \blacktriangleleft

We are now ready to present our $O(\log \log \Delta)$ sparsification. We show that we can find a subset of nodes incident to all nodes in U such that their induced maximum degree is $2^{O(\log f)}$ for $f = 2\sqrt{\log \Delta}$. This is achieved by repeating the sampling processes of Lemmas 17 and 18 for $O(\log \log \Delta)$ times. Here, one key observation to bound the deviation is that in each run of Lemma 17 only the lower tail may deviate up to a $1/3$ -factor from $\frac{|N_G(u)|}{\sqrt{\Delta}}$. So, the final multiplicative error will be $3^{O(\log \log \Delta)} = \text{poly log } \Delta$.

► **Lemma 19.** *Let G be a graph with bipartition $V(G) = U \sqcup V$. Let Δ and $\frac{\Delta}{f}$ be an upper bound on the maximum degree and a lower bound on the minimum degree, respectively, of any node in U for any parameter $f \leq \frac{\Delta^{0.4}}{\log n}$ and $f \geq \text{poly}(\log n)$. There exists a sublinear MPC algorithm that computes in $O(\log \log \Delta)$ rounds a subset $V^{sub} \subseteq V$ such that for any node $u \in U$ with $\deg_G(u) \geq \frac{\Delta}{f}$, it holds that $|N_G(u) \cap V^{sub}| \in [1, 2^{O(\log f)}]$. The algorithm global space usage is linear in the input size.*

Proof. Our goal is to find a suitable set V^{sub} by applying the sparsification outlined in Lemma 17. If $\Delta \geq n^\alpha$, we first apply Lemma 18 for $O(1/\varepsilon) = O(1)$ times until the maximum degree in U is within the memory capacity of a single machine $O(n^\alpha)$, which can be achieved by setting $\varepsilon \leq \frac{\alpha}{10}$, i.e., $n^\alpha \geq n^{10\varepsilon}$. Define $\Delta' \leq n^\alpha$ as the maximum degree in U after downsampling vertices in V for $O(1)$ iterations as per Lemma 18. Notice that the minimum degree in U is now $c \cdot \frac{\Delta'}{f}$, for some constant $c > 0$. Then, we run the algorithm of Lemma 17 for $k = O(\log \log \Delta)$ iterations, and stop as soon as the minimum degree in U is within $2^{O(\log f)}$. We prove by induction that after k iterations nodes have degrees in

$$\left[\frac{c}{f \cdot 3^k} (\Delta')^{1/2^k}, (\Delta')^{1/2^k} \right].$$

The base case follows from Lemma 17. The induction step then follows from

$$\left[\frac{c}{f \cdot 3^{(k-1)}} (\Delta')^{1/2^{(k-1)}} \cdot \frac{1}{3(\Delta')^{1/2^k}}, (\Delta')^{1/2^{(k-1)}} \cdot \frac{1}{(\Delta')^{1/2^k}} \right] = \left[\frac{c}{f \cdot 3^k} (\Delta')^{1/2^k}, (\Delta')^{1/2^k} \right].$$

By choosing $k = \lfloor \log \log \Delta' - \log(2 \log(f \cdot \log \Delta')) \rfloor$, one can verify that, for any vertex in U , the minimum degree in the downsampled graph will be at least one, and the maximum degree at most $2^{O(\log(f \cdot \log \Delta))} = 2^{O(\log f)}$. ◀

Our 2-ruling set algorithm is parameterized by $f = 2^{\sqrt{\log \Delta}}$. On a high-level, we mimic the randomized local 2-ruling set algorithm of [35]. In each iteration i , $0 \leq i \leq \lfloor \log f \rfloor$, we address the set of vertices with degree in $(\Delta/f^{i+1}, \Delta/f^i]$. We apply the sparsification of Lemma 19 on each set of high-degree vertices, one set at a time sequentially. Each sparsified subgraph is then put aside and, together with all incident nodes in G , is removed from further consideration before starting the next iteration. At the end, the union of all subgraphs of induced maximum degree $2^{O(\log f)}$ and possibly some remaining low-degree vertices are given in input to an MIS algorithm, whose solution is effectively a 2-ruling set. We detail the algorithm in the following pseudocode and proceed to its analysis below.

■ **Algorithm 1** SUBLINEAR 2-RULING SET.

```

 $f \leftarrow 2^{\sqrt{\log \Delta}}; M \leftarrow \emptyset$ 
for  $i \leftarrow 0, 1, \dots, \lfloor \log f \rfloor$  do
   $U \leftarrow \{v \in V \mid \deg_G(v) \in (\frac{\Delta}{f^{i+1}}, \frac{\Delta}{f^i}]\}; V' \leftarrow V$ 
   $G' \leftarrow (U \sqcup V', E' = \{(u, v) \mid u \in U, v \in V', (u, v) \in E\})$  ▷ Bipartition for sparsification
  for  $j \leftarrow 1, 2, \dots, O(\log \log \Delta)$  do ▷ See also Lemma 19
     $\Delta' \leftarrow$  maximum degree in  $G'$ 
     $V' \leftarrow$  sample  $v \in V'$  with prob.  $\max\{\frac{2}{3\sqrt{\Delta'}}, \frac{1}{n^\varepsilon}\}$ 
   $M \leftarrow M \cup V'$ 
   $V \leftarrow V \setminus (V' \cup N_G(V'))$  ▷ Remove neighbors of sampled set
Return MIS on  $G[M \cup V]$ 

```

The proofs of the next two lemmas are fairly standard and deferred to Appendix B.

► **Lemma 20.** *At the end of iteration i , $1 \leq i \leq \lfloor \log f \rfloor$, all vertices still in V have degree at most $\max\{\frac{\Delta}{f^i}, 2^{O(\log f)}\}$.*

► **Lemma 21.** *After $\lfloor \log f \rfloor$ iterations, the subgraph induced by M together with vertices still in V , i.e., $G[M \cup V]$, has maximum degree $2^{O(\log f)}$.*

Proof of Theorem 2. As proved in Lemma 19, each iteration of the algorithm runs in $O(\log \log \Delta)$ rounds. Since there are $O(\sqrt{\log \Delta})$ iterations for $f = 2^{\sqrt{\log \Delta}}$, the total number of rounds is $O(\sqrt{\log \Delta} \cdot \log \log \Delta)$. From Lemma 21, we see that the sparsified graph given by M together with vertices still in V has degree at most $2^{O(\sqrt{\log \Delta})}$. Therefore, the MIS computation at the end of the algorithm takes $O(\sqrt{\log \Delta} + \log \log^* n)$ by using the deterministic MIS algorithm from Lemma 27 of [20] that runs in $O(\log \Delta' + \log \log^* n)$ on a Δ' -maximum degree graph, provided that the allowed global space is $O(n^{1+\delta} + m)$. Otherwise, we use the variation given in [23] that runs in $O(\sqrt{\log \Delta} \cdot \log \log n)$ and uses linear global space. ◀

Lastly, we need to show how to achieve a $\text{poly}(\Delta)$ coloring of G^2 to fulfill the assumption made in Lemma 17. Due to space constraints, it is deferred to Appendix B.1.

References

- 1 Noga Alon, László Babai, and Alon Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of Algorithms*, 7(4):567–583, 1986. doi:10.1016/0196-6774(86)90019-2.
- 2 Alexandr Andoni, Aleksandar Nikolov, Krzysztof Onak, and Grigory Yaroslavtsev. Parallel algorithms for geometric graph problems. *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, 2013. URL: <https://api.semanticscholar.org/CorpusID:316401>.
- 3 Alkida Balliu, Sebastian Brandt, Juho Hirvonen, Dennis Olivetti, Mikaël Rabie, and Jukka Suomela. Lower bounds for maximal matchings and maximal independent sets. *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 481–497, 2019. URL: <https://api.semanticscholar.org/CorpusID:57721262>.
- 4 Alkida Balliu, Sebastian Brandt, Fabian Kuhn, and Dennis Olivetti. Distributed delta-coloring plays hide-and-seek. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, pages 464–477, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3519935.3520027.
- 5 Alkida Balliu, Sebastian Brandt, and Dennis Olivetti. Distributed lower bounds for ruling sets. *SIAM Journal on Computing*, pages 70–115, 2022. URL: <https://epubs.siam.org/doi/10.1137/20M1381770>, doi:10.1137/20M1381770.
- 6 Leonid Barenboim, Michael Elkin, Seth Pettie, and Johannes Schneider. The locality of distributed symmetry breaking. *J. ACM*, 63(3), June 2016. doi:10.1145/2903137.
- 7 Paul Beame, Paraschos Koutris, and Dan Suciu. Communication steps for parallel query processing. *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGAI symposium on Principles of database systems*, 2013. URL: <https://api.semanticscholar.org/CorpusID:11086753>.
- 8 M. Bellare and J. Rompel. Randomness-efficient oblivious sampling. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 276–287, 1994. doi:10.1109/SFCS.1994.365687.
- 9 Andrew Berns, James Hegeman, and Sriram V. Pemmaraju. Super-fast distributed algorithms for metric facility location. *ArXiv*, abs/1308.2473, 2012. URL: <https://api.semanticscholar.org/CorpusID:124685>, arXiv:1308.2473.
- 10 Tushar Bisht, Kishore Kothapalli, and Sriram V. Pemmaraju. Brief announcement: Super-fast t-ruling sets. *Proceedings of the 2014 ACM symposium on Principles of distributed computing*, 2014. URL: <https://api.semanticscholar.org/CorpusID:12210091>.

- 11 Mélanie Cambus, Fabian Kuhn, Shreyas Pai, and Jara Uitto. Time and Space Optimal Massively Parallel Algorithm for the 2-Ruling Set Problem. In Rotem Oshman, editor, *37th International Symposium on Distributed Computing (DISC 2023)*, volume 281 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 11:1–11:12, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.DISC.2023.11.
- 12 Keren Censor-Hillel, Merav Parter, and Gregory Schwartzman. Derandomizing local distributed algorithms under bandwidth restrictions. *Distributed Computing*, 33(3):349–366, June 2020. doi:10.1007/s00446-020-00376-1.
- 13 Yi-Jun Chang, Manuela Fischer, Mohsen Ghaffari, Jara Uitto, and Yufan Zheng. The Complexity of (Delta+1) Coloring in Congested Clique, Massively Parallel Computation, and Centralized Local Computation. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, PODC '19, pages 471–480, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3293611.3331607.
- 14 Benny Chor and Oded Goldreich. On the power of two-point based sampling. *Journal of Complexity*, 5(1):96–106, 1989. doi:10.1016/0885-064X(89)90015-0.
- 15 Sam Coy and Artur Czumaj. Deterministic massively parallel connectivity. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, pages 162–175, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3519935.3520055.
- 16 Artur Czumaj, Peter Davies, and Merav Parter. Simple, deterministic, constant-round coloring in the congested clique. In *Proceedings of the 39th Symposium on Principles of Distributed Computing*, PODC '20, pages 309–318, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3382734.3405751.
- 17 Artur Czumaj, Peter Davies, and Merav Parter. Component stability in low-space massively parallel computation. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, PODC'21, pages 481–491, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3465084.3467903.
- 18 Artur Czumaj, Peter Davies, and Merav Parter. Graph sparsification for derandomizing massively parallel computation with low space. *ACM Trans. Algorithms*, 17(2), May 2021. doi:10.1145/3451992.
- 19 Artur Czumaj, Peter Davies, and Merav Parter. Improved Deterministic (Delta+1) Coloring in Low-Space MPC. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, PODC'21, pages 469–479, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3465084.3467937.
- 20 Artur Czumaj, Peter Davies-Peck, and Merav Parter. Component stability in low-space massively parallel computation. *Distributed Computing*, 37(1):35–64, March 2024. doi:10.1007/s00446-024-00461-9.
- 21 Guy Even, Oded Goldreich, Michael Luby, Noam Nisan, and Boban Veličković. Efficient approximation of product distributions. *Random Structures & Algorithms*, 13(1):1–16, 1998. doi:10.1002/(SICI)1098-2418(199808)13:1<1::AID-RSA1>3.0.CO;2-W.
- 22 Manuela Fischer, Jeff Giliberti, and Christoph Grunau. Improved Deterministic Connectivity in Massively Parallel Computation. In Christian Scheideler, editor, *36th International Symposium on Distributed Computing (DISC 2022)*, volume 246 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 22:1–22:17, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.DISC.2022.22.
- 23 Manuela Fischer, Jeff Giliberti, and Christoph Grunau. Deterministic massively parallel symmetry breaking for sparse graphs. In *Proceedings of the 35th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '23, pages 89–100, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3558481.3591081.
- 24 Beat Gfeller and Elias Vicari. A randomized distributed algorithm for the maximal independent set problem in growth-bounded graphs. In *ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, 2007. URL: <https://api.semanticscholar.org/CorpusID:13473182>.

- 25 Mohsen Ghaffari. An improved distributed algorithm for maximal independent set. In *Proceedings of the 2016 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 270–277, 2016. doi:10.1137/1.9781611974331.ch20.
- 26 Mohsen Ghaffari, Themis Gouleakis, Christian Konrad, Slobodan Mitrović, and Ronitt Rubinfeld. Improved massively parallel computation algorithms for mis, matching, and vertex cover. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, PODC '18*, pages 129–138, New York, NY, USA, 2018. Association for Computing Machinery. doi:10.1145/3212734.3212743.
- 27 Mohsen Ghaffari, Fabian Kuhn, and Jara Uitto. Conditional hardness results for massively parallel computation from distributed lower bounds. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1650–1663, 2019. doi:10.1109/FOCS.2019.00097.
- 28 Mohsen Ghaffari and Jara Uitto. Sparsifying distributed algorithms with ramifications in massively parallel computation and centralized local computation. In *Proceedings of the 2019 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1636–1653, 2019. doi:10.1137/1.9781611975482.99.
- 29 Michael T. Goodrich. Communication-efficient parallel sorting. *SIAM Journal on Computing*, 29(2):416–432, 1999. doi:10.1137/S0097539795294141.
- 30 Michael T. Goodrich, Nodari Sitchinava, and Qin Zhang. Sorting, searching, and simulation in the mapreduce framework. In Takao Asano, Shin-ichi Nakano, Yoshio Okamoto, and Osamu Watanabe, editors, *Algorithms and Computation*, pages 374–383, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. doi:10.1007/978-3-642-25591-5_39.
- 31 James Hegeman, Sriram V. Pemmaraju, and Vivek Sardeshmukh. Near-constant-time distributed algorithms on a congested clique. In *International Symposium on Distributed Computing*, 2014. URL: <https://api.semanticscholar.org/CorpusID:277941>.
- 32 Howard Karloff, Siddharth Suri, and Sergei Vassilvitskii. A model of computation for mapreduce. In *Proceedings of the 2010 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 938–948, 2010. doi:10.1137/1.9781611973075.76.
- 33 Kishore Kothapalli, Shreyas Pai, and Sriram V. Pemmaraju. Sample-And-Gather: Fast Ruling Set Algorithms in the Low-Memory MPC Model. In Nitin Saxena and Sunil Simon, editors, *40th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2020)*, volume 182 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 28:1–28:18, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.FSTTCS.2020.28.
- 34 Kishore Kothapalli and Sriram Pemmaraju. Distributed graph coloring in a few rounds. In *Proceedings of the 30th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, PODC '11*, pages 31–40, New York, NY, USA, 2011. Association for Computing Machinery. doi:10.1145/1993806.1993812.
- 35 Kishore Kothapalli and Sriram V. Pemmaraju. Super-fast 3-ruling sets. In *Foundations of Software Technology and Theoretical Computer Science*, 2012. URL: <https://api.semanticscholar.org/CorpusID:16038481>.
- 36 Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. Local computation: Lower and upper bounds. *J. ACM*, 63(2), 2016. doi:10.1145/2742012.
- 37 Nathan Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992. doi:10.1137/0221015.
- 38 Michael Luby. Removing randomness in parallel computation without a processor penalty. *Journal of Computer and System Sciences*, 47(2):250–286, 1993. doi:10.1016/0022-0000(93)90033-S.
- 39 Yannic Maus, Saku Peltonen, and Jara Uitto. Distributed symmetry breaking on power graphs via sparsification. In *Proceedings of the 2023 ACM Symposium on Principles of Distributed Computing, PODC '23*, pages 157–167, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3583668.3594579.

- 40 Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Cambridge university press, 1995. doi:10.1017/CB09780511814075.
- 41 Rajeev Motwani, Joseph (Seffi) Naor, and Moni Naor. The probabilistic method yields deterministic parallel algorithms. *Journal of Computer and System Sciences*, 49(3):478–516, 1994. 30th IEEE Conference on Foundations of Computer Science. doi:10.1016/S0022-0000(05)80069-8.
- 42 Krzysztof Nowicki. A deterministic algorithm for the mst problem in constant rounds of congested clique. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1154–1165, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3406325.3451136.
- 43 Shreyas Pai and Sriram V. Pemmaraju. Brief announcement: Deterministic massively parallel algorithms for ruling sets. In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing*, PODC'22, pages 366–368, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3519270.3538472.
- 44 Prabhakar Raghavan. Probabilistic construction of deterministic algorithms: Approximating packing integer programs. *Journal of Computer and System Sciences*, 37(2):130–143, 1988. doi:10.1016/0022-0000(88)90003-7.
- 45 Johannes Schneider, Michael Elkin, and Roger Wattenhofer. Symmetry breaking depending on the chromatic number or the neighborhood growth. *Theor. Comput. Sci.*, 509:40–50, 2013. doi:10.1016/J.TCS.2012.09.004.

A Missing Proofs for Linear MPC Result

Proof of Lemma 12. We analyze one step of (a variation of) Luby’s algorithm that builds an independent set \mathcal{I} on the set of sampled bad vertices $\bigcup_d B_d \cap V_{\text{samp}}$. We will fix a seed specifying a hash function from a pairwise independent family \mathcal{H} . Let $v \in (\bigcup_d B_d \cap V_{\text{samp}})$. An hash function h maps node v to a value $z_v \in [n^3]$. Then, v joins the independent set \mathcal{I} iff $z_v < z_w$ for all $w \sim v$ and $z_v < \frac{n^3}{d^{3\varepsilon}}$, where $w \in N(v) \cap (\bigcup_d B_d \cap V_{\text{samp}})$.

By Lemma 10, each node $u \in \tilde{B}_d$ has at least $d^{0.1}$ nodes from S_u that are sampled, each of which has at most $d^{2\varepsilon}$ sampled neighbors. For the purpose of the analysis, let the set A_u include exactly $d^{0.1} = d^{4\varepsilon}$ of such nodes and let $\{X_v\}_{v \in A_u}$ be the random variables denoting the event that v joins \mathcal{I} . We denote $X = \sum_{v \in A_u} X_v$ as their sum. For any v , we have

$$\frac{1}{d^{3\varepsilon}} - \frac{1}{n^3} \leq \Pr \left[z_v < \frac{n^3}{d^{3\varepsilon}} \right] \leq \frac{1}{d^{3\varepsilon}}.$$

By pairwise independence,

$$\Pr[X_v = 1] \geq \Pr \left[z_v < \frac{n^3}{d^{3\varepsilon}} \right] - \sum_{v' \in N(v) \cap S(B)} \Pr \left[z_{v'} \leq z_v < \frac{n^3}{d^{3\varepsilon}} \right] \geq \frac{1}{d^{3\varepsilon}} - \frac{1}{n^3} - \frac{d^{2\varepsilon}}{d^{6\varepsilon}} \geq \frac{1}{3d^{3\varepsilon}}.$$

It follows that $\mathbb{E}[X] = \sum_{v \in A_u} \Pr[X_v = 1] \geq \frac{d^\varepsilon}{3}$. Our goal is now to bound $\Pr[X = 0]$. Observe that for any two vertices $v, v' \in A_u$, we have that

$$\mathbb{E}[X_v X_{v'}] \leq \Pr \left[z_v < \frac{n^3}{d^{3\varepsilon}} \cap z_{v'} < \frac{n^3}{d^{3\varepsilon}} \right] \leq d^{-6\varepsilon},$$

by pairwise independence. Thus, we get

$$\frac{\text{Var}[X]}{\mathbb{E}[X]^2} \leq \frac{\sum_{v \in A_u} \text{Var}[X_v] + \sum_{v, v' \in A_u} \text{Cov}[X_v, X_{v'}]}{\mathbb{E}[X]^2}.$$

29:20 Massively Parallel Ruling Set Made Deterministic

We know that

$$\sum_{v \in A_u} \text{Var}[X_v] \leq d^{4\varepsilon} \cdot \Pr[X_v = 1](1 - \Pr[X_v = 1]) \leq d^{4\varepsilon} \cdot \frac{1}{3d^{3\varepsilon}} = \frac{d^\varepsilon}{3},$$

$$\sum_{v, v' \in A_u} \text{Cov}[X_v, X_{v'}] \leq d^{8\varepsilon} (\mathbb{E}[X_v X_{v'}] - \mathbb{E}[X_v] \mathbb{E}[X_{v'}]) \leq d^{8\varepsilon} (d^{-6\varepsilon} - 1/9d^{6\varepsilon}) \leq d^{2\varepsilon}.$$

Therefore,

$$\text{Var}[X] \leq \frac{d^\varepsilon}{3} + d^{2\varepsilon} \leq \frac{4d^\varepsilon}{3}, \text{ and } \frac{\text{Var}[X]}{\mathbb{E}[X]^2} \leq \frac{\frac{4d^\varepsilon}{3}}{\left(\frac{d^\varepsilon}{3}\right)^2} = \frac{4d^\varepsilon}{3} \cdot \frac{9}{d^{2\varepsilon}} = \frac{36}{d^\varepsilon}.$$

Applying Chebyshev's inequality, we have

$$\Pr[X = 0] \leq \Pr[|X - \mathbb{E}[X]| \geq \mathbb{E}[X]] \leq \frac{\text{Var}[X]}{\mathbb{E}[X]^2} \leq \frac{45}{d^\varepsilon}. \quad \blacktriangleleft$$

B Missing Proofs for Low-Memory MPC Result

Proof of Lemma 20. Consider a high-degree vertex $u \in U$ at the start of the i -th iteration. By Lemma 19, each node in U is incident to a node that joins the set M by the end of this iteration. Since all vertices incident to M are removed from V , the lemma follows. \blacktriangleleft

Proof of Lemma 21. First, consider a vertex v that joins the set M at some iteration j . Observe that no neighbor of v in G had joined M earlier, otherwise, u would have been removed. By Lemma 19, all vertices that join M at iteration j have induced degree at most $2^{O(\log f)}$. Then, the neighbors of M are removed from V and, thus, cannot join M anymore. This proves that vertices in M have degree at most $2^{O(\log f)}$. Second, consider a vertex w that at the end of the $\lfloor \log f \rfloor$ -th iteration is still in V . This means that w does not neighbor M and that, by Lemma 20, w has degree at most $2^{O(\log f)}$, finishing the claim. \blacktriangleleft

B.1 Coloring of G^2

Here, we discuss how to compute a $\text{poly}(\Delta)$ coloring of G^2 to fulfill the assumption made in Lemma 17.

Whenever $\Delta = n^{\Omega(1)}$, the initial assignment of IDs to vertices, typically from 1 to n , effectively serves as a $\text{poly}(\Delta)$ coloring of G^2 . In the case where $\Delta \leq n^\delta$ for constant $\delta < \alpha/2$, we ensure $\Delta^2 \ll n^\alpha$. This implies that the 2-hop neighborhood of every node can be stored within the local memory of a single machine. Storing the 2-hop neighbors on a single machine permits the use of Linial's coloring reduction technique [37], which achieves a $O(\Delta^6)$ coloring in $O(1)$ rounds. However, this approach necessitates a global space usage of $O(n^{1+2\delta})$, potentially exceeding $O(n + m)$. To improve the global space usage, after three runs of Lemma 17, the degree of each vertex which has not been removed is at most $\Delta^{0.22}$. Since each sampled vertex is incident to a high-degree vertex of initial degree at least $O(\Delta/f)$, we can charge high-degree vertices $O(\Delta^{0.66}) \ll \Delta/f$ space consumption. This reduction allows us to gather the 2-hop neighbors of all active nodes onto single machines without breaching the global space limit. A further optimization involves substituting the first three runs of Lemma 17 with a weaker version, detailed below, addressing all but at most $\frac{n}{\Delta^{0.01}}$ vertices. The proof follows from that of Lemma 17.

► **Lemma 22.** *Let $G = (V, E)$ be a graph with an upper bound Δ on the maximum degree. There is a sublinear MPC algorithm that computes in $O(1)$ rounds a subset $V' \subseteq V$ ensuring that, for all but at most $\frac{n}{\Delta^{0.01}}$ vertices $v \in V$ with $\deg_G(v) \geq \log(n) \cdot \Delta^{0.6}$, it holds that $|N_G(v) \cap V'| \in \left[\frac{1}{3\sqrt{\Delta}} |N_G(v)|, \frac{1}{\sqrt{\Delta}} |N_G(v)| \right]$.*

Applying Lemma 22 initially and excluding up to $\frac{n}{\Delta^{0.01}}$ vertices not meeting our criteria allows for the execution of $O(\log \log \Delta)$ iterations for the well-behaved vertices. The excluded vertices are subsequently addressed by repeating the same process. After $O(1)$ iterations, the remaining vertex count drops to $O(\frac{n}{\Delta^2})$, fitting the global space needed to store their 2-hop neighborhoods within $O(n)$. Consequently, after $O(\log \log \Delta)$ rounds, all vertices are processed without affecting the asymptotic total number of rounds.