Deterministic Massively Parallel Connectivity[*][†]

Sam Coy
S.Coy@warwick.ac.uk
University of Warwick
Coventry, United Kingdom

Artur Czumaj
A.Czumaj@warwick.ac.uk
University of Warwick
Coventry, United Kingdom

ABSTRACT

We consider the problem of designing fundamental graph algorithms on the model of Massive Parallel Computation (MPC). The input to the problem is an undirected graph G with n vertices and m edges, and with D being the maximum diameter of any connected component in G. We consider the MPC with *low local space*, allowing each machine to store only $\Theta(n^{\delta})$ words for an arbitrary constant $\delta > 0$, and with linear global space (which is the number of machines times the local space available), that is, with optimal utilization.

In a recent breakthrough, Andoni et al. (FOCS'18) and Behnezhad et al. (FOCS'19) designed parallel randomized algorithms that in $O(\log D + \log\log n)$ rounds on an MPC with low local space determine all connected components of a graph, improving on the classic bound of $O(\log n)$ derived from earlier works on PRAM algorithms.

In this paper, we show that asymptotically identical bounds can be also achieved for deterministic algorithms: we present a deterministic MPC low local space algorithm that in $O(\log D + \log\log n)$ rounds determines connected components of the input graph. Our result matches the complexity of state of the art randomized algorithms for this task. The techniques developed in our paper can be also applied to several related problems, giving new deterministic MPC algorithms for problems like finding a spanning forest, minimum spanning forest, etc.

We complement our upper bounds by extending a recent lower bound for connectivity on an MPC conditioned on the 1-vs-2-cycles conjecture (which requires $D \geq \log^{1+\Omega(1)} n$), by showing a related conditional hardness of $\Omega(\log D)$ MPC rounds for the entire spectrum of D, covering a particularly interesting range when $D \leq O(\log n)$.

CCS CONCEPTS

• **Theory of computation → Massively parallel algorithms**; **Pseudorandomness and derandomization**; • **Mathematics of computing → Graph algorithms**.

[*]A full version of this paper can be found at https://arxiv.org/abs/2108.04102.

[†]This research was supported in part by the Centre for Discrete Mathematics and its Applications (DIMAP), by EPSRC award EP/V01305X/1, by an EPSRC studentship, by a Weizmann-UK Making Connections Grant, and by an IBM Award.

KEYWORDS

derandomization, connectivity, MapReduce, MPC, massively parallel algorithms

ACM Reference Format:
Sam Coy and Artur Czumaj. 2022. Deterministic Massively Parallel Connectivity. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing (STOC '22), June 20–24, 2022, Rome, Italy*. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3519935.3520055

1 INTRODUCTION

Motivated largely by recent very successful development of a number of massively parallel computation frameworks, such as MapReduce [30], Hadoop [58], Dryad [40], and Spark [59], we have seen in the last decade a booming research interest in the design of parallel algorithms. While some of this research has been very applied, there has been also an increasing interest in fundamental and algorithmic research which aim to understand the computational power of such systems. In this paper we study the complexity of some fundamental graph problems on the *Massively Parallel Computation (MPC)* model, first introduced by Karloff et al. [42], which is now the standard theoretical model of study, as it provides a clean abstraction of these practical frameworks, see, e.g., [2, 9, 10, 39, 42].

The MPC model can be seen as a variant of the classical BSP model which balances the parallelism (like in the classical PRAM model) with the communication costs of distributed systems. An MPC is a parallel system with some number $\mathcal{M}$ of *machines*, each having $\mathcal{S}$ words of *local memory*. MPC computations are synchronous, with alternating rounds of computation and communication. In each synchronous round, each machine processes its local data and performs an arbitrary (unlimited) local computation. At the end of each round, machines exchange messages. Each message is sent to a single machine specified by the machine that is sending the message. Each machine may send and receive at most $\mathcal{S}$ words each round, as the messages must fit into the machine's local space.

It is known that a single step of PRAM can be simulated in a constant number of rounds on MPC (with $\mathcal{S} \geq n^{\Omega(1)}$) [39, 42], implying that many PRAM algorithms can be implemented in the MPC model without any asymptotic loss of complexity. However, the MPC model is often *more powerful* as it allows for a lot of local computation (in principle, an unbounded amount each round). For example, data stored on a single machine can be processed in a single round, even if the underlying local problem is computationally hard. Using this observation, we have recently seen a stream of papers demonstrating that several graph problems: connectivity, matching, maximal independent set, vertex cover, coloring, etc (see, e.g., [5, 6, 11, 13, 15, 16, 21, 28, 35, 38, 43, 45, 54]), can be solved on the MPC model much faster than on PRAM. However, most of these results relied on randomized algorithms, and comparatively little research has been done to study deterministic algorithms.

This leads to the main theme of this work: *to explore the power of the MPC model in the setting of deterministic algorithms*. Specifically, we investigate under what circumstances one can achieve (asymptotically) the same complexity bounds for deterministic MPC algorithms as those known for their randomized counterparts.

In this paper we consider graph problems, where the input is a graph $G = (V, E)$ with n vertices and m edges. For such a problem, the input (collection V of vertices and E of edges) is initially arbitrarily distributed among the machines, with *global space* $\mathcal{S}_{tot} = \mathcal{S} \cdot \mathcal{M} = \Omega(n + m)$. We focus on the design of *low space* (fully scalable) MPC algorithms for graph problems with optimal space utilization, that is, we assume that the local space of each machine is strongly sublinear in the number of vertices, i.e., for an arbitrarily small constant $\delta > 0$, we have $\mathcal{S} = O(n^{\delta})$, and the global space $\mathcal{S}_{tot}$, which is the product $\mathcal{S} \cdot \mathcal{M}$, is *linear*, that is, $\mathcal{S}_{tot} = O(n + m)$.

The low-space regime is particularly challenging due to the fact that a single vertex cannot necessarily store all incident edges on a single machine. For example, the very simple problem of distinguishing between a cycle of length n and two cycles of length $\frac{n}{2}$ seems to be hard for this model, and it is conjectured to require $\Omega(\log n)$ MPC rounds (see Conjecture 1). Despite these challenges, it has been recently shown that this setting allows for very efficient algorithms for some related fundamental graph problems. Firstly, a simple BSF-like approach can be used to solve the connectivity problem deterministically in $O(D)$ rounds on an MPC with $\mathcal{S} = O(n^{\delta})$ and linear global space $\mathcal{S}_{tot} = O(n + m)$. Secondly, it is not difficult to design an MPC algorithm that solves the connectivity problem (even deterministically) in $O(\log n)$ rounds on an MPC with $\mathcal{S} = O(n^{\delta})$ and $\mathcal{S}_{tot} = O(n + m)$; one approach here could be to use a PRAM algorithm (see, e.g., [23]), and there are also direct MPC implementations.

One of the recent highlights in this area has been a sequence of works on the connectivity problem (see, e.g., [3, 7, 13, 44, 47]), initiated with a seminal work of Andoni et al. [3] and culminating with a *randomized MPC algorithm* due to Behnezhad et al. [13] that for a given graph with each connected component having the diameter at most D, determines (with high probability) all connected components in $O(\log D + \log\log n)$ rounds on an MPC with $\mathcal{S} = O(n^{\delta})$ and $\mathcal{S}_{tot} = O(n + m)$. While this algorithm matches the conjectured bound for the 1-vs-2 cycle problem for $D = \Omega(n)$, it *significantly improves the complexity for low diameter graphs* (e.g., for random graphs we have $D = O(\log n)$). Furthermore, this complexity bound seems to be almost optimal, since for $D \geq \log^{1+\Omega(1)} n$, in [13] the authors show that an $o(\log D)$-rounds algorithm would imply the existence of an $o(\log n)$-rounds algorithm for the 1-vs-2 cycle problem.

Given the fundamental character of the connectivity problems for graph algorithms, it is important to understand to what extent the use of randomness in these algorithms is necessary. This is especially critical

because the design of optimal deterministic PRAM algorithm has been a notorious task in the past (see, e.g., [23]). This is the challenge addressed in our paper.

1.1 Our Contribution

In this paper we show that the recent randomized MPC connectivity algorithms (with $\mathcal{S} = O(n^{\delta})$ local space and $\mathcal{S}_{tot} = O(m+n)$ global space) can be derandomized without asymptotic loss of complexity. We consider the algorithmic approach due to Andoni et al. [3] and Behnezhad et al. [13] and abstract from it two key algorithmic steps that use randomization: finding a constant approximation of maximum matching on a path, and finding an approximation of a dominating set in a graph with a given minimum degree. Both steps rely on a simple random sampling approach, which is easy to implement using a randomized $O(1)$-rounds MPC algorithm but which seems to be difficult to implement deterministically, especially in our setting of low space MPC with optimal global space utilization. We use derandomization tools which rely on small sample spaces with limited dependence, and some new algorithmic techniques which allow us to implement these tools on an MPC. As a result, we obtain an algorithm (see Theorem 4.7) that for a given undirected graph G with diameter D, in $O(\log D + \log\log n)$ rounds on an MPC (with low local space and linear global space) deterministically identifies all connected components of G.

Main Theorem (Theorem 4.7). Let $\delta > 0$ be an arbitrary constant. Let G be an undirected graph with n vertices, m edges, and each connected component having diameter at most D. One can *deterministically* identify the connected components of G in $O(\log D + \log\log_{m/n} n)$ rounds on an MPC with $\mathcal{S} = O(n^{\delta})$ local space and $\mathcal{S}_{tot} = O(m + n)$ global space.

We note that $\log\log_{m/n} n \leq O(\log\log n)$, and we clarify that, since it can be that $m < n$, we take $O(\log\log_{m/n} n)$ to mean $O(\log\log_{\max\{2,m/n\}} n)$.

While we do not know whether the additive term $O(\log\log_{m/n} n)$ is necessary with linear global space, similarly as in [3] and [13], we can incorporate our derandomization framework to obtain a stronger bound for an MPC with *superlinear* global space.

Theorem 4.8. Let $0 \leq \gamma \leq \frac{1}{2}$ be arbitrary. Let G be an undirected graph with n vertices, m edges, and each connected component having diameter at most D. One can deterministically identify the connected components of G in $O(\log D + \log(\frac{\log n}{2+\gamma\log n}))$ rounds on an MPC with $\mathcal{S} = O(n^{\delta})$ local space and $\mathcal{S}_{tot} = O((m + n)^{1+\gamma})$ global space.

Observe that for $\gamma = \Omega(1)$, that is, when γ is larger than an arbitrary positive constant, Theorem 4.8 gives a deterministic $O(\log D)$-rounds MPC algorithm.

Lower bounds. We complement our results from Theorems 4.7 and 4.8 with a conditional lower bound. Behnezhad et al. [13] showed recently that for $D \geq \log^{1+\Omega(1)} n$, if the 1-vs-2-cycles conjecture (cf. Section 2.2.1) holds then no MPC algorithm can determine in $O(\log D)$ rounds each connected component of a graph with diameter at most D. We extend this bound to all values of D.

Theorem 5.2 (informal). If the 1-vs-2-cycles conjecture holds, then for any D, any MPC algorithm with local space $\mathcal{S} = O(n^{\delta})$ that with high probability for any given graph G correctly determines all connected components of G with diameter at most D requires $\Omega(\log D)$ rounds.

Theorem 5.2 shows that even for random graphs, which in a typical model have diameter $D = O(\log n)$, or even $D = o(\log n)$, no algorithm can achieve $o(\log D)$ MPC rounds complexity (assuming the 1-vs-2-cycles conjecture). No such bound was known before.

Extension to other related problems. Similarly as in [3], our derandomization techniques can be used to solve related graph problems. While the obtained bounds have a small loss of the complexity when only a linear global space is used, the achieve optimal round complexity when the global space is $\mathcal{S}_{tot} = (n+m)^{1+\Omega(1)}$.

Corollary 1.1. *Let G be an undirected graph with n vertices, m edges, and diameter D. In $O(\log D \cdot \log\log_{m/n} n)$ rounds on an MPC (with $\mathcal{S} = O(n^\delta)$ local space and $\mathcal{S}_{tot} = O(n+m)$ global space) one can deterministically compute a rooted spanning forest of G.*

With global space $\mathcal{S}_{tot} = (O(n+m))^{1+\gamma}$ for $0 \le \gamma \le \frac{1}{2}$, and the same local space $\mathcal{S} = O(n^\delta)$, one can reduce the number of rounds to $O\left(\min\left\{\log D \cdot \log(\frac{\log n}{2+\gamma \log n}), \log n\right\}\right)$.

Observe that for $\gamma = \Omega(1)$, Corollary 1.1 gives an $O(\log D)$-rounds MPC algorithm.

Corollary 1.2. *Let G be an undirected graph with weights $w : V \to \mathbb{Z}$ such that $w(e) \le poly(n)$ for all $e \in E$. Let D_{MFS} be the smallest diameter of a minimum spanning forest of G. For any $0 \le \gamma \le \frac{1}{2}$, one can deterministically find a minimum spanning forest and a bottleneck spanning forest of G in*

$$O\left(\min\left\{\left(\log D_{MFS} + \log(\frac{\log n}{2+\gamma \log n})\right) \cdot \log(\frac{\log n}{2+\gamma \log n}), \log n\right\}\right)$$

rounds on an MPC with $\mathcal{S} = O(n^\delta)$ local space and $\mathcal{S}_{tot} = (O(n+m))^{1+\gamma}$ global space.

These corollaries follow straightforwardly from Theorem 3.8, Theorem 4.1, and Theorem 4.7.

Observe that when $\mathcal{S}_{tot} = O(m+n)$, that is, when $\gamma \le O(\frac{1}{\log n})$, then the bounds above are just as good as on a PRAM, giving only deterministic $O(\log n)$ rounds MPC algorithms (bounds previously known). However, the main strength of Corollary 1.2 is for larger γ. For example, when the global space is $\mathcal{S}_{tot} = (n+m)^{1+\Omega(1)}$, that is, when γ is larger than an arbitrary positive constant, then Corollary 1.2 implies that both these problems have $O(\log D_{\text{MFS}})$-rounds deterministic MPC algorithms.

It is worth mentioning that the bound in Corollary 1.2 has a dependency on D_{MFS} (the diameter of a minimum spanning forest of the input graph), and not on the diameter D. This makes these bounds weaker, since while we clearly have $D_{\text{MFS}} \ge D$, in general we might have $D_{\text{MFS}} \gg D$. It is therefore tempting to hope for an MPC algorithm running in $\widetilde{O}(\log D)$ rounds, but we can showthat this is impossible unless the 1-vs-2-cycles conjecture fails. In fact, we prove that even in graphs of constant diameter and with small weights, conditioned on the 1-vs-2-cycles conjecture, any MPC algorithm (with $\mathcal{S} = O(n^\delta)$ and $\mathcal{S}_{tot} = \text{poly}(n)$) requires $\Omega(\log n)$ rounds. The proof of this claim is deferred to the full version.

Overcoming the limitations of the LOCAL model. We believe that an important contribution of our work is a demonstration that even though the MPC model is closely related to several classical models of distributed computing, the use of global communication can sometimes make MPC algorithms considerably more powerful when seen from the point of view of parallel computation than through its links to the distributed models.

Ghaffari, Kuhn, and Uitto [37] introduced the notion of *component-stable* low-space MPC algorithms, which are (informally) MPC algorithms where the outputs reported by the vertices in different connected components must be independent. They were introduced with the *intuition* that *all, or almost all known efficient MPC algorithms are component-stable*, so the restriction to study such algorithms can be done almost without loss of generality. The main result of [37] (see also [26]) is a connection between the complexity of component-stable low-space MPC algorithms and computation in the classical LOCAL model of distributed computing. Informally, [37] presented a technique that for many graph problems translates an $\Omega(T)$-round LOCAL lower bound into an $\Omega(\log T)$-round lower bound in the low-space MPC model conditioned on the 1-vs-2 cycles conjecture. Czumaj, Davies, and Parter [26] extended this framework to deterministic algorithms. In particular, this shows that the two problems we want to derandomize *require a super-constant number of MPC rounds*, unless we consider non-component-stable MPC algorithms. This follows from two lower bounds for deterministic algorithms in the LOCAL model: in a graph with maximum degree 2 there is no deterministic $o(\log^* n)$-time LOCAL algorithm that returns a constant approximation of the maximum matching [29, 46], and no deterministic constant-round LOCAL algorithm can find an $o(\Delta)$-approximation of a minimum dominating set in Δ-regular graphs [29, 46] (see also Section 6.4 in [57]).

While this suggests that the two problems required in our analysis cannot be solved deterministically in a constant number of MPC rounds, we prove that this is *not the case*: one can construct $O(1)$-rounds deterministic MPC algorithms for a constant approximation of a maximum matching in graphs with maximum degree at most 2, and for a good approximation of an appropriate instance of dense dominating set. As it is required by the framework from [26, 37] in view of the lower bounds for the LOCAL model [29, 46], our algorithms are not component-stable (since they rely on a global (non-component-stable) coordination between the outcomes of independent runs using variables coming from small limited independence probability spaces).

Derandomization. The central tool in our analysis is the derandomization approach using the method of conditional probabilities in the framework of pairwise independent random variables and k-wise ε-approximately independent random variables. While this framework is well established, its efficient implementation on an MPC, and especially one with low local space and optimal global space, requires a careful approach. In Section 2.4.1 we provide an overview of the approach using the method of conditional probabilities on MPC, but here we briefly discuss some main challenges.

The algorithm for finding a large matching in degree-2 graphs incorporates pairwise independent random variables combined with some nowadays rather standard (see, e.g., [8, 19, 24–26, 31, 36]) implementation on an MPC, and hence we focus on our approach for the other algorithm: the algorithm for the random leader contraction process, which is a version of an approximation of a minimum dominating set problem in dense graphs. Our approach (see Section 3.1) for this problem is more complicated. While most of the earlier works for derandomization of MPC algorithms use the notion of k-wise independence, our work uses the method of conditional probabilities for k-wise ε-approximately independent random variables (Definition 3.1). The use of this more complex notion is caused by the fact that the algorithm relies on a significant independence between the underlying random variables, and we require $\Omega(\log n)$-independence. Since $\Omega(\log n)$-independent random variables do not have sufficiently small probability spaces, we have to further enhance our setting and consider k-wise ε-approximately independent random variables with $k = O(\log n)$ and $\varepsilon = n^{-O(1)}$ (the proof that such a setting works in our case — as can be seen in Theorem 3.4 — is quite subtle, see the full version, and it relies on an interesting concentration bound for k-wise ε-approximate independence, see Theorem 3.3). The next challenge is how to implement the method of conditional probabilities for k-wise ε-approximately independent random variables in a constant number of rounds on an MPC with low local space and linear global space. While our approach shares some ideas from earlier works (e.g., from [17, 36], though these papers were allowed to use $O(\log n)$-wise independence and run in a polylogarithmic number of rounds), a proper combination of the use of small $O(\log n)$-wise $\frac{1}{\text{poly}(n)}$-approximately independent probability spaces and implementation in a constant number of rounds requires some careful design (see Section 3.1).

1.2 Related Work

The connectivity problem has been studied extensively in sequential, distributed, and parallel settings. Since this problem is frequently used as a building block to more advanced problems on graphs, a good understanding of its complexity is vital.

In the classical PRAM model of parallel computation, a long sequence of papers demonstrated that the connectivity can be solved in $O(\log n)$ parallel time on an EREW PRAM with an optimal $O(n+m)$ total work. For a long time this bound was only known for randomized algorithms: only after more than a decade of research, this bound was achieved also by deterministic algorithms (see [23] and the discussion therein). Very recently, by extending the ideas originally developed for MPC algorithms in [3, 13], Liu, Tarjan, and Zhong [47] gave an $O(\log D + \log\log_{m/n} n)$-time randomized algorithm on an ARBITRARY CRCW PRAM using $O(m)$ processors whp. No comparable deterministic bounds are known.

For MPC algorithms, while our paper focuses on the low local space MPC algorithms, originally most research was concerned with the setting when $\mathcal{S} = \Omega(n)$. Unlike the low space regime, if we consider an MPC with *larger local space*, then constant-round MPC algorithms are possible. For example, Lattanzi et al. [43] gave a constant-round MPC algorithm for connectivity in the $\mathcal{S} = n^{1+\Omega(1)}$ local space regime, and by exploring the relationship between MPC with $\mathcal{S} = O(n)$ and the Congested Clique model, the Congested Clique $O(1)$-rounds algorithm for connectivity due to Jurdziński and Nowicki [41] yields a $O(1)$-rounds connectivity MPC algorithm with $\mathcal{S} = O(n)$ and $\mathcal{S}_{tot} = O(n+m)$; recently Nowicki [53] extended this result and obtained a *deterministic* $O(1)$-rounds connectivity MPC algorithm with $\mathcal{S} = O(n)$ and $\mathcal{S}_{tot} = O(n+m)$.

The low local space regime is more challenging though. By simulating PRAM algorithms, we can easily obtain $O(\log n)$-rounds MPC algorithms in this regime. This bound seems to be as good as it gets, since it is widely conjectured (see the 1-vs-2 cycles Conjecture 1) that no MPC algorithm (neither deterministic nor randomized) with low local space and a polynomial number of machines can distinguish one n-vertex cycle from two $\frac{n}{2}$-vertex cycles in $o(\log n)$ rounds. However, if we have a *sufficiently large global space* (global space $\mathcal{S}_{tot} = O(n^{2.373-\delta})$ suffices) then even with $\mathcal{S} = O(n^\delta)$, it is easy to solve the connectivity problem deterministically in $O(\log D)$ MPC rounds using a simple matrix multiplication approach. In view of these results, recent study has been aiming to reach the bound of $O(\log D)$ rounds on an MPC with low local space $\mathcal{S} = O(n^\delta)$ and linear global space $\mathcal{S}_{tot} = O(n+m)$, culminating in almost matching bounds for *randomized* algorithms in [3, 13]. Let us mention also a related randomized algorithm in [7] that determines connected components in sparse graphs in $O(\log \frac{1}{\lambda} + \log\log n)$ rounds on an MPC with low local space $\mathcal{S} = O(n^\delta)$ and linear global space $\mathcal{S}_{tot} = \widetilde{O}(\frac{1}{\lambda^2}(n+m))$, where λ is a lower bound for the spectral gap of its connected components (this bound is never better than $O(\log D + \log\log n)$). All these results are randomized and we are not aware of any prior $o(\log n)$-rounds deterministic MPC algorithm in the low local space regime. Still, a recent related work [22] shows that one can *slightly reduce the randomness* used in [13]: $(\log n)^{O(\log D + \log\log_{m/n} n)}$ random bits suffice to obtain an MPC algorithm with local space $\mathcal{S} = O(n^\delta)$ and $\mathcal{M} = O(n+m)$ machines (and so with global space $O((n+m)\cdot n^\delta)$, which is larger than the linear global space $O(n+m)$ used in [13]) that determines graph connectivity in $O(\log D + \log\log_{m/n} n)$ rounds with probability $1 - 1/\text{poly}((m \log n)/n)$.

While there are some deterministic MPC algorithms with low round complexity and linear global space, such results are still rare; some recent examples include [8, 18, 24, 25, 27]. The occasional use of extra global space is largely because the derandomization has some overhead which may lead to extra $O(n^\delta)$ (or even larger) factor in the global space bounds.

The relation between various models in distributed computing has been recently extensively investigated. An earlier work in this area (see, e.g., [12]) observed a very close relation between the distributed Congested Clique model and the MPC model with $\mathcal{S} = O(n)$, implying that computationally (essentially) these two models have "almost" the same strength. Similarly, there have been several papers investigating a relation between the LOCAL model and the MPC model with low local space, see, e.g., [18, 26, 37, 38]. Apart from the conditional lower bounds (for component-stable algorithms) mentioned earlier [26, 37], we have also seen upper bounds, most notably [38, 54] and recent results for trees in [18].

1.3 Paper Overview

In Section 2 we give an overview of the problem of connectivity, some useful techniques in MPC, the previous (randomized) connectivity algorithms of [3, 13], and derandomization techniques.

In Section 3, we show how the subroutine of "random leader contraction" can be derandomized. In Section 4 we show that derandomized leader contraction (along with a second derandomized subroutine whose details we defer to the full version) allow us to derandomize the connectivity algorithms of [3, 13].

Finally, in Section 5 we give our lower bound result.

2 PRELIMINARIES

2.1 Graph Connectivity

Before we proceed, let us first formally introduce the connectivity problem in the MPC setting.

Definition 2.1. *Let $G = (V, E)$ be an undirected graph with a vertex set V and an edge set E. The goal is to compute a function $cc : V \longrightarrow \mathbb{N}$ such that every vertex $u \in V$ knows $cc(u)$ and for any pair of vertices $u, v \in V$, u and v are connected in G if and only if $cc(u) = cc(v)$.*

Our algorithms will rely on the operation of *vertex/edge contractions*. We will repeatedly contract one vertex u to another adjacent vertex v, which means deleting the edge $\{u, v\}$ and identifying vertices u and v by removing u and connecting to v all edges incident earlier to u. Further, since these operations will be performed in parallel, we will ensure that the contractions are independent in the sense that they all can be performed in a single round. Once we know the pairs u, v, these operations can be performed deterministically in a constant number of rounds on an MPC with $\mathcal{S} = O(n^\delta)$ local space and $\mathcal{S}_{tot} = O(n)$ global space (see also Section 2.2).

Let us also mention that if a graph G^* is obtained by a sequence of contractions of vertices/edges in a graph G, then G^* maintains all connected components of G.

2.2 Basic Algorithmic Tools on Low Space MPC

We will be using the algorithmic toolbox for MPC with low local space and linear global space as developed in earlier works, most notably by Goodrich et al. [39]. We will regularly use the following lemma describing some basic (*deterministic*) algorithmic results.

Lemma 2.2. **[39]** *Let δ be an arbitrary positive constant. Sorting of n numbers, computing prefix sums of n numbers, and computing the predecessor predicate*[1] *can be performed deterministically in a constant number of rounds on an MPC using $\mathcal{S} = O(n^\delta)$ local space, $\mathcal{S}_{tot} = O(n)$ global space, and poly(n) local computation.*

In the sorting and prefix sums (given $x_1, \dots, x_n$ and an associate operator $\oplus$, output $y_1, \dots, y_n$ with $y_j = \oplus_{i=1}^{j} x_j$) problems the input is arbitrarily distributed among the machines, and in the output, the i-th machine stores the i-th chunk of the sorted elements or the appropriate prefix sums.

Observe that this setup allows us perform some basic computations on graphs deterministically in a constant number of MPC rounds. This includes the tasks of computing vertex degree of every vertex, or ensuring that every vertex has all incident edges stored on the machine its allocated to (or if it's degree $\deg(u) = \Omega(n^\delta)$ then on $O(n^\delta/\deg(u))$ consecutive machines), etc.

Further, note that we typically will assume that G is a simple graph, or that we can make it simple. This is because using Lemma 2.2, one can make any graph simple in a constant number of rounds with $\mathcal{S} = O(n^\delta)$ and $\mathcal{S}_{tot} = O(n+m)$: sort all edges $\{u, v\}$ according to u and v and then remove all duplicates (after sorting this task can be easily done locally on each machine, or if some parallel edges do not fit a single machines, one can easily coordinate the removal between the machines). This can be further extended to the process of independent contractions (assuming that if we contract some vertices to a vertex u then u is not contracted to any other vertex).

Finally, let us notice that the framework of Lemma 2.2 allows us to perform quickly the following task of *colored summation*: Given a sequence of n pairs of numbers $\langle \text{color}_i, x_i \rangle$, $1 \le i \le n$, with $C = \{\text{color}_i : 1 \le i \le n\}$, compute $S_c = \sum_{i:\text{color}_i = c} x_i$ for all $c \in C$.

Lemma 2.3. *Let δ be an arbitrary positive constant. The problem of colored summation for n pairs of numbers can be solved deterministically in a constant number of rounds on an MPC using $\mathcal{S} = O(n^\delta)$ local space, $\mathcal{S}_{tot} = O(n)$ global space, and poly(n) local computation.*

2.2.1 One vs. Two Cycles Problem.

One of the notorious problems for low space MPC is the problem of distinguishing whether an input graph is an n-vertex cycle or two $\frac{n}{2}$-vertex cycles. This fundamental MPC problem has been studied extensively in the literature and has played a central role in the development of conditional lower bounds for MPC graph algorithms (see, e.g., [13, 45, 52, 56]). This problem is trivial for an MPC with local space $\Omega(n)$, but we do not know its complexity for low space MPCs. This leads to the celebrated (and widely believed) 1-vs-2 cycles conjecture:

Conjecture 1. ***(1-vs-2 cycles conjecture)*** *Let $\delta < 1$ be an arbitrary positive constant. No randomized MPC algorithm with local space $\mathcal{S} = O(n^\delta)$ and a polynomial number of machines can distinguish one n-vertex cycle from two $\frac{n}{2}$-vertex cycles in $o(\log n)$ rounds.*

The required failure probability in Conjecture 1 is typically $\frac{1}{\text{poly}(n)}$, but sometimes may be made as large as 0.01.

2.3 Intuitions Behind Recent Advances in Randomized Connectivity

We first present the main idea behind the approach due to Andoni et al. [3] that gives a randomized $O(\log D \log\log_{m/n} n)$-rounds MPC algorithm for connectivity. Then we briefly discuss how this approach can be extended to obtain a randomized $O(\log D + \log\log n)$-rounds MPC algorithm for the connectivity due to Behnezhad et al. [13]. Our main result is a derandomization of the latter algorithm.

The main idea behind an $\widetilde{O}(\log D)$-round connectivity algorithm, as introduced by Andoni et al. [3], is to repeatedly perform *vertex contractions*, which reduce the number of vertices, and *expansion*, which creates new edges within connected components. The critical parameter in the analysis is limited local space, and with this the global space $\mathcal{S}_{tot}$, which is $\mathcal{S}_{tot} = \mathcal{S} \cdot \mathcal{M}$. Graph contraction reduces the number of vertices and edges, but expansion increases the number of edges. Therefore we will need to ensure a balance, so that the number of edges is always at most $\mathcal{S}_{tot}$.

Let b be some integer parameter. A central observation is that if the input graph $G = (V, E)$ has all vertices of degree at least b, then a classical result about dominating sets ensures that G has a dominating U set of size $O(\frac{n \log n}{b})$, and moreover, such a dominating set can be easily found (*by a randomized algorithm*). Since U is a dominating set, all vertices in G are either in U or are neighbors of a vertex in U, and therefore one can locally contract all vertices from $V \setminus U$ to U, reducing the number of vertices in the contracted graph to $O(\frac{n \log n}{b})$. (This step was called *random leader contraction* in [3].)

[1]For the predecessor operation, the input consists of n objects $x_1, \dots, x_n$, each associated with a 0-1 value; the output is a sequence $y_1, \dots, y_n$ such that y_j is the last object $x_{j'}$ with $j' < j$ which has associate value of 1.

This observation suggests the following approach. Start with the original graph G with n vertices and m edges, and set $b = \frac{m}{n}$. Notice that G can be represented with global space $\mathcal{S}_{tot} = O(n+m)$, which we will see as $O(m)$. Then, perform "expansion" on G by adding some new edges without changing the connectivity structure, so that each vertex has degree at least b (unless its connected component has size at most b), while not increasing the total number of edges by too much, so that in total the size of the graph is smaller than $\mathcal{S}_{tot}$. Then, use the property above (contracting after leader contraction using dominating set) to reduce the number of vertices from n to $n_1 = O(\frac{n \log n}{b})$. In the obtained graph we perform expansion again, but this time, since we have fewer vertices, aim to ensure that each vertex has degree at least b_1 and the total number of added edges is $O(b_1 n_1) = O(m)$, so that it fits the global space of the MPC. The constraint $b_1 n_1 = O(m)$ leads us to the choice for $b_1 = \frac{m}{n_1} = O(\frac{mb}{n \log n})$, or equivalently, $b_1 = O(\frac{b^2}{\log n})$. We can see here that we will make double-exponential progress on b_i.

If we continue this approach, then after t iterations we obtain a graph with at most n_t vertices, which can be later expanded to ensure that each vertex has degree at least b_t with at most $b_t n_t = O(m)$ edges, where $n_t = O(\frac{n_{t-1} \log n}{b_{t-1}}) = O(\frac{m}{\log n}(\frac{\log n}{b})^{2^t})$ and $b_t = \frac{m}{n_t} = O((\frac{b}{\log n})^{2^t} \log n)$.

Notice that this process works in phases and each phase reducing the number of vertices by a factor $O(\frac{\log n}{b})$. Since we started with $b = \frac{m}{n}$, this would lead to a randomized MPC algorithm running in $O(\log \log_{m/n} n)$ phases. This general technique is called *double-exponential speed problem size reduction* in [3]. Note that the result of $O(\log \log_{m/n} n)$ phases holds for any $b_t = (\frac{m}{n_t})^{\Omega(1)}$ (and in fact, the algorithm of [3] uses $b_t = (\frac{m}{n_t})^{1/2}$: we presented a simpler version here for the purposes of exposition).

What is left is to design a procedure which performs expansion. This can be done (deterministically) using a standard "transitive closure"-like approach, where each vertex u connects itself, repeatedly, to each vertex in the 2-hop neighbourhood of u. Care needs to be taken to ensure we do not exceed our total memory $\mathcal{S}_{tot}$: therefore we stop repeating this process for vertex u when it knows its entire component or has degree at least b, whichever comes first.

If we combine vertex expansion with leader contraction, then we will obtain a randomized algorithm that in $O(\log \log_{m/n} n \cdot \log D)$ MPC rounds contract each connected component to a single vertex, and thereby computes all connected components.

There are two weaknesses of this approach: Firstly, its complexity is $\omega(\log D)$ rounds unless $m \geq n^{1+\Omega(1)}$. Secondly, the algorithm for random leader contraction described above achieves a high success probability only when b is large, and thus the total probability of succeeding is in the worst case only constant, say 0.99. Behnezhad et al. [13] extended the framework due to Andoni et al. [3] and evaded these two weaknesses to achieve $O(\log D + \log \log_{m/n} n)$ MPC rounds with high probability.

The first part of the improvement stems from a randomized algorithm that in $O(1)$ MPC rounds contracts edges to reduce the number of vertices of G by a constant factor with high probability. Using this algorithm, in $O(\log \log n)$ MPC rounds one can reduce the number of vertices from n to $n/\text{polylog}(n)$, without increasing the number of edges. As a result, the global space available per vertex becomes polylogarithmic, and the approach by Andoni et al. presented above can start with $b \geq \text{polylog}(n)$, ensuring that random leader contraction in [3] *will succeed with high probability*.

On its own, this will only increase the success probability, but the framework would still require $O(\log \log_{m/n} n)$ phases, each phase taking $O(\log D)$ rounds. The second and the main contribution of Behnezhad et al. [13] is the observation that vertices can perform each step (neighborhood expansion versus leader contraction) independently of other vertices. In each round, each vertex u (which has not been contracted yet) has its own *budget* of the local space it can use, with the constraint (informally) that the sum of all budgets is $O(\mathcal{S}_{tot})$. Each vertex starts with a budget of $(\frac{m}{n})^{1/2}$. Then, informally, every vertex v in a constant number of MPC rounds either increases its budget (from $b(v)$ to $b(v)^{1+c}$ for a small constant c), or learns its entire 2-hop neighborhood. Notice that a vertex can only increase its budget $O(\log \log n)$ times: once the budget of a vertex reaches n, the vertex is able to store all vertices in its allocated local space, which is always sufficient.

2.4 Introduction to Derandomization

The derandomization approach in this paper follows the classical approach. We study random sampling algorithms that originally assume full independence between random choices, and in the first step, we reduce the independence of the involved random variables, either to allow their pairwise independence, or k-wise ε-approximate independence (Definition 3.1). Then, we set an appropriate target function modeling some required properties of the algorithm. Next, we apply the method of conditional probabilities to the target function to obtain a deterministic MPC algorithm.

This outline of the approach combines three different aspects: the use of appropriate random variables with some limited independence coming from a small probability space (or equivalently, modeled by a family of hash functions of polynomial size, the use of the method of conditional probabilities to select one good hash functions the previous step (see Section 2.4.1), and the design of an efficient MPC algorithm implementing this approach. The last part is less standard and (for one problem) more challenging, and therefore we will pay special care to its description.

2.4.1 *Method of Conditional Probabilities on MPC.*

A central tool in our derandomization of algorithms is the *method of conditional probabilities* (or *expectations*) (see, e.g., [1, 51, 55]). We consider a setting where there is a family of hash functions $\mathcal{H}$ of size poly(n) and we know that there exists at least one $h^* \in \mathcal{H}$ with some desirable properties. Our goal is to find one such h^*. This task can easily be performed in poly(n) time by going through all functions in $\mathcal{H}$, but we want to perform it in a constant number of rounds on an MPC with local space $\mathcal{S} = O(n^\delta)$ and linear global space $\mathcal{S}_{tot} = O(n+m)$, which is insufficient to collect information about all evaluations of all functions from $\mathcal{H}$ on the entire MPC.

The method of conditional probabilities, developed by Erdös and Selfridge [32], Raghavan [55], and many others, finds such an h^* by iteratively refining $\mathcal{H}$ into smaller subsets, $\mathcal{H} = \mathcal{H}^{\langle 0 \rangle} \supseteq \mathcal{H}^{\langle 1 \rangle} \supseteq \dots$, until finally we end up with a one element set containing a desirable h^*. While refining $\mathcal{H}$, we ensure that we are not increasing (or not decreasing) the conditional expectation of some target function F on $\mathcal{H}$, that is, to ensure that $\mathbb{E}_{h \in \mathcal{H}_{i+1}}[F(h)] \leq \mathbb{E}_{h \in \mathcal{H}_i}[F(h)]$.

The two main challenges of this approach are efficiently computing the conditional expectations, and appropriately bounding the number of iterations needed to find a one element subset of $\mathcal{H}$.

We will incorporate the approach used recently in [20] and [24] (see also recent results in [25–27]; though a similar approach has been used in earlier, classic works on derandomization, see, e.g., [50] or [49, 55], with the only difference being that now we have to process by blocks of $O(\log n/\delta)$ bits, whereas in earlier works one was processing individual bits, one bit at a time[2]). We will search for function h^* by splitting the $O(\log n)$-bit seed defining it into smaller parts of $\chi < \log \mathcal{S} = O(\delta \log n) = O(\log n)$ bits each[3], and then processing one part at time, iteratively extending a fixed prefix of the seed until we have fixed the entire seed.

The selection for the target function $F(h)$ is central for this process to work. If we consider a partition of the currently considered subset $\mathcal{H}^*$ of $\mathcal{H}$ into $\mathcal{H}_1^*, \dots, \mathcal{H}_{2^\chi}^*$, then it is essential that each machine will be able to efficiently compute $\mathbb{E}_{h\in\mathcal{H}_i^*}[F(h)]$ for all $1 \le i \le 2^\chi$.

3 DERANDOMIZATION OF THE RANDOM LEADER CONTRACTION PROCESS

In this section we consider the following formalization of the task required in the random leader contraction process, as used in our deterministic MPC connectivity algorithm[4]:

> *Let b and n be integers with $b \le n$. Let $S_1, S_2, \dots, S_n$ be subsets of $\{1, \dots, n\}$ with $|S_i| = b$ and $i \in S_i$, for every $1 \le i \le n$. Our goal is to find a* small *subset $\mathcal{L} \subseteq \{1, \dots, n\}$ (set of "leaders") such that $S_i \cap \mathcal{L} \ne \emptyset$ for every $i \in \{1, \dots, n\}$.*

We will be assuming that $b \ge \log^{10} n$.

Algorithm 1: Random sampling algorithm for random leader contraction

```
1 Let $X_1, \dots, X_n$ be identically distributed 0-1 random
    variables from distribution $\mathcal{D}_{n,p}$
2 Return $\mathcal{L} := \{i \in \{1, \dots, n\} : X_i = 1\}$
```

Let us begin with the standard random sampling based algorithm for this task (Algorithm 1). It is not difficult to see (see, e.g., [3, Lemma III.1] and also [1, Theorem 1.2.2] for an existential and tighter result) that if $p \ge \frac{10\cdot\ln n}{b}$ then with high probability,

(i) for every $1 \le i \le n$, $\sum_{j\in S_i} X_j > 0$, and
(ii) $\sum_{i=1}^n X_i \le 2np$,

which implies that if we set $p = \frac{10\cdot\ln n}{b}$ then random variables $X_1, \dots, X_n$ correspond to a set $\mathcal{L}$ which is the solution to the problem of size at most $\frac{20n\ln n}{b}$.

This simple construction relies heavily on random sampling and in what follows we will derandomize a construction of set $\mathcal{L}$ with similar properties. We will consider a small k-wise ε-approximately independent family of hash functions $\mathcal{H}$ for distribution $\mathcal{D}_{n,p}$ (see Definition 3.1 and Theorem 3.2) and consider Algorithm 2, a revision of Algorithm 1.

Definition 3.1. (**k-wise ε-approximate independence**) *Let $0 \le \varepsilon \le 1$ and let $k, n, \ell \in \mathbb{N}$ with $k \le n$. A family of hash functions $\mathcal{H} = \{h : \{1, \dots, n\} \to \{0,1\}^\ell\}$ is called* k-wise ε-approximately independent *if for any $t \le k$, any distinct $i_1, \dots, i_t \in \{1, \dots, n\}$ and any (not necessarily distinct) $b_1, \dots, b_t \in \{0,1\}^\ell$, we have*

$$\left|\mathbf{Pr}\left[h(i_j) = b_j, 1 \le j \le t\right] - \left(\frac{1}{2^\ell}\right)^t\right| \le \varepsilon \enspace .$$

Random variables sampled from a k-wise ε-approximately independent family of hash functions[5] *are called* k-wise ε-approximately independent random variables.

Theorem 3.2. **[34, Theorem 1]** *Let $0 \le \varepsilon \le 1$ and let $k, n, \ell \in \mathbb{N}$ with $k \le n$. One can construct a k-wise ε-approximately independent family of hash functions $\mathcal{H} = \{h : \{1, \dots, n\} \to \{0,1\}^\ell\}$ such that choosing a random function h from $\mathcal{H}$ takes $O(k + \log(1/\varepsilon) + \log\log n)$ random bits. Furthermore, each $h \in \mathcal{H}$ can be specified and evaluated on any input in polylog$(n, \ell, 2^k, 1/\varepsilon)$ time and space.*

Algorithm 2: "Random" leader contraction using k-wise ε-approximate independence

```
1 Let $X_1, \dots, X_n$ be $k$-wise $\varepsilon$-approximately independent 0-1
    random variables for $\mathcal{D}_{n,p}$
2 Return $\mathcal{L} := \{i \in \{1, \dots, n\} : X_i = 1\}$
```

Next, we give a tail bound for k-wise, ε-approximate variables. We defer the proof to the full version due to space constraints.

Theorem 3.3. *Let $k \ge 4$ be an even integer. Suppose $X_1, \dots, X_n$ are k-wise ε-approximately independent 0-1 random variables (with identical marginal distribution). Let $X = X_1 + \dots + X_n$ and $\mu = \mathbb{E}[X]$, and let $t > 0$. Then*

$$\mathbf{Pr}[|X - \mu| \ge t] \le 8 \cdot \left(\frac{k\mu + k^2}{t^2}\right)^{k/2} + \varepsilon \cdot \left(\frac{n}{t}\right)^k \enspace .$$

By a careful setting of p, k, and ε, and an application of the tail bound of Theorem 3.3, we obtain the following:

Theorem 3.4. *Let $\log^{10} n \le b \le n$, $k \ge 4$ be even, $k = 15\log_b n$, $\varepsilon = n^{-6}$, and $p = b^{-\frac{1}{5}}$. If $X_1, \dots, X_n$ are k-wise ε-approximately independent random variables for distribution $\mathcal{D}_{n,p}$, then each of the following $n+1$ events hold with probability at least $1 - 9n^{-3}$:*

- *$\sum_{j\in S_i} X_j > 0$ for every $1 \le i \le n$, and*
- *$\sum_{i=1}^n X_i \le 2np = 2nb^{-\frac{1}{5}}$.*

Remark 3.5. *In our problem description we assume that $|S_i| = b$ for every $1 \le i \le n$. Clearly, the same claim also holds if $|S_i| \ge b$ for every $1 \le i \le n$: just remove all but b elements from each set (arbitrarily) and apply Theorem 3.4.*

[2]One can think of the most standard process as being modeled by a binary tree, and we consider a $O(n^\delta)$-ary tree.
[3]One could think that $\chi = \lfloor \log \mathcal{S} \rfloor - 1 \le \log(\mathcal{S}/2)$, to ensure that every machine has about half of its local space for the analysis of hash functions and another half for its normal data.
[4]The intuition: every vertex i has a leader from $\mathcal{L}$ in its own set S_i and then i can be contracted to such a leader vertex, reducing the size of the vertex set from n to $|\mathcal{L}|$. Thus our goal is to find a small $\mathcal{L}$ with the properties above.
[5]That is, random variables $X_1, \dots, X_n$ such that $X_i = h(i)$ with h selected uniformly at random from $\mathcal{H}$.

3.1 Efficient MPC Implementation of Deterministic Leader Contraction

In this section we will show how to use the analysis of Algorithm 2 in Theorem 3.4 and the derandomization framework using small families of k-wise ε-approximate independent hash functions to obtain a deterministic constant rounds MPC algorithm for leader contraction (Theorem 3.8). Our main focus here is on the case $b < \mathcal{S}$, though at the end we will consider also the case $b = \Omega(\mathcal{S})$.

We use the approach sketched in Section 2.4.1. Take a k-wise ε-approximately independent family of hash functions $\mathcal{H}$ for distribution $\mathcal{D}_{n,p}$ from Theorem 3.2, with the following parameters from Theorem 3.4: even $k = \Theta(\log_b n) \geq 4$, $\varepsilon = n^{-6}$, and $p = b^{-\frac{1}{5}}$. Observe that by Theorem 3.2, $|\mathcal{H}| = 2^{O(k+\log(1/\varepsilon)+\log\log n)} = \operatorname{poly}(n)$, so taking a function h from $\mathcal{H}$ can be done by specifying $O(\log n)$ bits, and evaluating a function h from $\mathcal{H}$ takes $\operatorname{polylog}(n)$ time.

By the probabilistic method, Theorem 3.4 implies that there is a function $h \in \mathcal{H}$ for which, if we generate random variables $X_1, \dots, X_n$ by taking a random hash function h from $\mathcal{H}$ and then defining $X_i = h(i)$, then

- for every $1 \leq i \leq n$, $\sum_{j \in S_i} X_j > 0$, and
- $\sum_{i=1}^n X_i < 2np + 1 = 2nb^{-\frac{1}{5}} + 1$.

Let any hash function $h \in \mathcal{H}$ satisfying these inequalities above be called *good*. Our goal is to find one good hash function $h^* \in \mathcal{H}$.

It is easy to find a good h^* if the MPC has sufficiently many machines (but still polynomial in n) by checking the inequalities above for all functions in $\mathcal{H}$, but we want to do it using optimal global space: $\mathcal{S}_{tot} = O(nb)$. (Recall that the input to the problem consists of n sets $S_1, \dots, S_n$ of size b each, of total size $O(nb)$.)

We partition the bits describing any function from $\mathcal{H}$ into blocks of size $\chi < \log \mathcal{S} = O(\delta \log n)$ bits each, and hence into $q = \frac{\lceil \log_2 |\mathcal{H}| \rceil}{O(\delta \log n)} = \frac{O(\log n)}{O(\delta \log n)} = O(1/\delta)$ blocks. We will determine $h^* \in \mathcal{H}$ by fixing its bits in blocks, in q phases, by refining $\mathcal{H}$ into $\mathcal{H} = \mathcal{H}_0 \supseteq \mathcal{H}_1 \supseteq \mathcal{H}_2 \supseteq \dots \supseteq \mathcal{H}_q$ with $\mathcal{H}_t = 2^\chi |\mathcal{H}_{t+1}|$ for $0 \leq t < q$, and $|\mathcal{H}_q| = 1$, and by ensuring that there is a good h in every $\mathcal{H}_t$.

We need some extra care here because of the conflict of two types of event: one, for each set S_i, aiming to ensure large values of $\sum_{j \in S_i} X_j$, while another requires that $\sum_{i=1}^n X_i$ is small. For that, we will determine the bits in a next block defining h^* using the concept of *pessimistic estimators*. Indeed, we would want to know whether in a subset $\mathcal{H}'$ of $\mathcal{H}_t$ currently under consideration there is one good h^*, but for that we need to estimate the probability that for a random $h \in \mathcal{H}'$, for random variables $X_1, \dots, X_n$ defined by h, we have for every $1 \leq i \leq n$, $\sum_{j \in S_i} X_j > 0$, and $\sum_{i=1}^n X_i \leq 2nb^{-\frac{1}{5}}$. But it is difficult to perform this task in a constant number of rounds on an MPC with low local space. Instead, we use an appropriate estimator for this event.

For any hash function $h \in \mathcal{H}$ defining 0-1 random variables $X_1, \dots, X_n$ with $X_i = h(i)$ for every $1 \leq i \leq n$, let us define the following functions:

- for every $1 \leq i \leq n$, $TC_i(h) := 1$ if $\sum_{j \in S_i} X_j = 0$ and 0 otherwise, and
- $L(h) := \sum_{i=1}^n X_i$.

Next, we define an aggregation cost function to be

$$\operatorname{cost}(h) = n \cdot \sum_{i=1}^n TC_i(h) + L(h) \quad . \tag{1}$$

Our goal is to use function *cost* as an estimator of the quality of a subset $\mathcal{H}^*$ of $\mathcal{H}$, to determine if $\mathcal{H}^*$ has many good hash functions. It is easy to see the following properties of our function cost:

Claim 3.6. *For any $h \in \mathcal{H}$, h is good iff $cost(h) \leq 2nb^{-\frac{1}{5}}$.*

The next claim follows easily from Theorem 3.4:

Claim 3.7. *If $n \geq 10$ then $\mathbb{E}_{h \in \mathcal{H}}[cost(h)] < 2nb^{-\frac{1}{5}} + 1$.*

With Claim 3.7, now the idea is simple. We will be searching for a good $h \in \mathcal{H}$ by refining $\mathcal{H}$ into smaller and smaller parts $\mathcal{H} = \mathcal{H}_0 \supseteq \mathcal{H}_1 \supseteq \mathcal{H}_2 \supseteq \dots$

Suppose that at some moment we consider hash functions from $\mathcal{H}^* \subseteq \mathcal{H}$. Let $\mathcal{H}_1^*, \dots, \mathcal{H}_q^*$ be a partition of $\mathcal{H}^*$. Our key observation is that even though in a constant number of rounds we cannot deterministically compute the probability that a randomly chosen function h from $\mathcal{H}_i^*$ is good, we *can* efficiently compute $\mathbb{E}_{h \in \mathcal{H}_i^*}[\operatorname{cost}(h)]$. We use this observation to argue that if initially $\mathbb{E}_{h \in \mathcal{H}^*}[\operatorname{cost}(h)] < 2nb^{-\frac{1}{5}} + 1$, then there must be one index $1 \leq i \leq q$ such that $\mathbb{E}_{h \in \mathcal{H}_i^*}[\operatorname{cost}(h)] < 2nb^{-\frac{1}{5}} + 1$. Therefore, in our partition of $\mathcal{H}^*$ we will take such $\mathcal{H}_i^*$ for which $\mathbb{E}_{h \in \mathcal{H}_i^*}[\operatorname{cost}(h)] < 2nb^{-\frac{1}{5}} + 1$, to ensure that there is a good hash function $h \in \mathcal{H}_i^*$.

We will define a refinement of $\mathcal{H}$ into $\mathcal{H} = \mathcal{H}_0 \supseteq \mathcal{H}_1 \supseteq \mathcal{H}_2 \supseteq \dots \supseteq \mathcal{H}_q$ with $\mathcal{H}_t = 2^\chi |\mathcal{H}_{t+1}|$ for $0 \leq t < q$, and $|\mathcal{H}_q| = 1$, where $\mathcal{H}_t$ is defined in phase t, $1 \leq t \leq q$.

At the beginning of phase t, $1 \leq t \leq q$, we have determined the first $(t-1)\chi$ bits $b_1, \dots, b_{(t-1)\chi}$ defining any function from $\mathcal{H}$. Let $\operatorname{bits}_{\mathcal{H}}(h)$ be the bit representation of h in $\mathcal{H}$ and let

$$\mathcal{H}_{t-1} = \{h \in \mathcal{H} : \operatorname{bits}_{\mathcal{H}}(h) \text{ has prefix } b_1, \dots, b_{(t-1)\chi}\}.$$

We will additionally ensure the *invariant* that at the beginning of phase t, $1 \leq t \leq q$, we have $\mathbb{E}_{h \in \mathcal{H}_{t-1}}[\operatorname{cost}(h)] < 2nb^{-\frac{1}{5}} + 1$. Observe that by Theorem 3.4, this invariant holds for $t = 1$.

At the beginning of phase t, we distribute bits $b_1, \dots, b_{(t-1)\chi}$ to all machines. Next, we assume (and we can ensure this in a constant number of rounds using the methods from Section 2.2) that the elements $\{1, \dots, n\}$ are arbitrarily assigned to the machines in a balanced way, the same number to each machine, and that the machine to which element i is assigned has also copies of all elements in S_i. (We will later comment how to deal with the case $b \geq \frac{\mathcal{S}}{2}$.)

Next, let us consider a single machine, say M_j. It contains some elements $\mathcal{E}_j$ from $\{1, \dots, n\}$ and their corresponding sets S_i with $i \in \mathcal{E}_j$, and it knows the bits $b_1, \dots, b_{(t-1)\chi}$. Consider all possible χ-bits sequences $\mathbf{b} = b_{(t-1)\chi+1}, b_{(t-1)\chi+2}, \dots, b_{t\chi}$. For each such $\mathbf{b}$, let $\mathcal{H}_{t-1}^{\mathbf{b}}$ be the set of all hash functions $h \in \mathcal{H}_{t-1}$ whose bit representation in $\mathcal{H}$ has prefix $b_1, \dots, b_{(t-1)\chi}$ followed by $\mathbf{b}$. (Notice that sets $\mathcal{H}_{t-1}^{\mathbf{b}}$ over all $\mathbf{b}$ define a partition of $\mathcal{H}_{t-1}$.) Next, we want to compute $\mathbb{E}_{h \in \mathcal{H}_{t-1}^{\mathbf{b}}}[\operatorname{cost}(h)]$. For that, for each $\mathbf{b}$ and $h \in \mathcal{H}_{t-1}^{\mathbf{b}}$ machine M_j computes locally two numbers:

- $TC^{\langle j \rangle}(h)$, to be equal to the number of $i \in \mathcal{E}_j$ with $\sum_{r \in S_i} X_r = 0$, and

- $L^{\langle j \rangle}(h) := \sum_{i \in \mathcal{E}_j} X_i$.

Observe that in this setting, while no single machine can compute $\mathrm{cost}(h)$ on its own, for every $h \in \mathcal{H}_{t-1}$ we have $\mathrm{cost}(h) = n \cdot \sum_j TC^{\langle j \rangle}(h) + \sum_j L^{\langle j \rangle}(h)$ and therefore by exchanging the appropriate numbers between the machines one can compute $\mathrm{cost}(h)$ for any single h. Unfortunately, we cannot compute all values of $\mathrm{cost}(h)$ since there are too many hash functions $h \in \mathcal{H}_{t-1}$ and aggregating the information needed to compute all these values requires too much communication on an MPC. However, we can efficiently compute $\sum_{h \in \mathcal{H}^{\mathbf{b}}_{t-1}} \mathrm{cost}(h)$ for all $\mathbf{b}$, and this immediately allows us to obtain $\mathbb{E}_{h \in \mathcal{H}^{\mathbf{b}}_{t-1}}[\mathrm{cost}(h)]$, since $\mathbb{E}_{h \in \mathcal{H}^{\mathbf{b}}_{t-1}}[\mathrm{cost}(h)] = \frac{1}{|\mathcal{H}^{\mathbf{b}}_{t-1}|} \cdot \sum_{h \in \mathcal{H}^{\mathbf{b}}_{t-1}} \mathrm{cost}(h)$.

In order to compute $\sum_{h \in \mathcal{H}^{\mathbf{b}}_{t-1}} \mathrm{cost}(h)$ for all $\mathbf{b}$, first, each machine M_j computes locally $TC^{\langle j \rangle}(h)$ and $L^{\langle j \rangle}(h)$ for all $h \in \mathcal{H}_{t-1}$, and then uses these number to compute, for every $\mathbf{b}$,

$$\mathrm{Tcost}_j(\mathbf{b}) = \sum_{h \in \mathcal{H}^{\mathbf{b}}_{t-1}} \left(n \cdot TC^{\langle j \rangle}(h) + L^{\langle j \rangle}(h) \right) .$$

Next, we aggregate these numbers across all machines and use the colored summation problem (see Lemma 2.3) to compute $\sum_j \mathrm{Tcost}_j(\mathbf{b})$ for all $\mathbf{b}$. Then, we distribute all $\sum_j \mathrm{Tcost}_j(\mathbf{b})$ with all $\mathbf{b}$ to all machines, after which each machine can locally compute $\mathbb{E}_{h \in \mathcal{H}^{\mathbf{b}}_{t-1}}[\mathrm{cost}(h)]$ since, as we mentioned above:

$$\mathbb{E}_{h \in \mathcal{H}^{\mathbf{b}}_{t-1}}[\mathrm{cost}(h)] = \frac{1}{|\mathcal{H}^{\mathbf{b}}_{t-1}|} \cdot \sum_{h \in \mathcal{H}^{\mathbf{b}}_{t-1}} \mathrm{cost}(h).$$

By simple averaging arguments (the probabilistic method, Section 2.4.1), there is at least one χ-bits sequences $\mathbf{b}$ with $\mathbb{E}_{h \in \mathcal{H}^{\mathbf{b}}_{t-1}}[\mathrm{cost}(h)] \le \mathbb{E}_{h \in \mathcal{H}_{t-1}}[\mathrm{cost}(h)]$ and therefore we can complete the phase by selecting $\mathcal{H}_t$ to be any $\mathcal{H}^{\mathbf{b}}_{t-1}$ with $\mathbb{E}_{h \in \mathcal{H}^{\mathbf{b}}_{t-1}}[\mathrm{cost}(h)] \le \mathbb{E}_{h \in \mathcal{H}_{t-1}}[\mathrm{cost}(h)]$. (As before, ties are broken arbitrarily; e.g., in order to be consistent among all machines, we can choose the smallest number $\mathbf{b}$ which minimizes $\mathbb{E}_{h \in \mathcal{H}^{\mathbf{b}}_{t-1}}[\mathrm{cost}(h)]$.)

Let us now remark (as we promised earlier) what can be done when $b = \Omega(\mathcal{S}) \equiv \Omega(n^\delta)$. In that case, we achieve a slightly weaker bound, but fully sufficient for our use: we will obtain $\sum_{i=1}^n X_i \le O(n(\frac{\mathcal{S}}{2})^{-\frac{1}{5}})$ (instead of $\sum_{i=1}^n X_i \le O(nb^{-\frac{1}{5}})$). If $b > \frac{\mathcal{S}}{2}$ then we take $\frac{\mathcal{S}}{2}$ elements from each set S_i arbitrarily and consider the problem with all sets of size $\frac{\mathcal{S}}{2}$ (see also Remark 3.5). Then, for the modified sets S_i we apply the same as above, since we have enough local space to fit each set S_i on a single machine. Therefore, the arguments above suffice to construct 0-1 variables $X_1, \dots, X_n$ such that for every $1 \le i \le n$, $\sum_{j \in S_i} X_j > 0$, and $\sum_{i=1}^n X_i < 2n(\frac{\mathcal{S}}{2})^{-\frac{1}{5}} + 1$.

In this way, after q phases, each phase consisting of a constant number of rounds, we find $\mathcal{H}_q$ which has a single element $h \in \mathcal{H}$ for which $\mathrm{cost}(h) \le \mathbb{E}_{h \in \mathcal{H}}[\mathrm{cost}(h)] < 2n(\min\{b, \frac{\mathcal{S}}{2}\})^{-\frac{1}{5}} + 1$. This is the hash function h^* sought: it generates a set $\mathcal{L} \subseteq \{1, \dots, n\}$ with $i \in \mathcal{L}$ iff $h^*(i) = 1$ such that (i) $S_i \cap \mathcal{L} \ne \emptyset$ for every $1 \le i \le n$ and (ii) $|\mathcal{L}| < 2n(\min\{b, \frac{\mathcal{S}}{2}\})^{-\frac{1}{5}} + 1$. Since $q = O(1/\delta) = O(1)$, this proves the following theorem.

Theorem 3.8. *Let b and be n be integer with $\log^{10} n \le b \le n$. Let $S_1, S_2, \dots, S_n$ be subsets of $\{1, \dots, n\}$ with $|S_i| = b$ and $i \in S_i$, for every $1 \le i \le n$. Then one can deterministically find a subset $\mathcal{L} \subseteq \{1, \dots, n\}$ with $|\mathcal{L}| \le O(n \cdot (\min\{b, \mathcal{S}\})^{-\frac{1}{5}})$ such that $S_i \cap \mathcal{L} \ne \emptyset$ for every $i \in \{1, \dots, n\}$ in a constant number of MPC rounds with local space $\mathcal{S} = O(n^\delta)$ and global space $\mathcal{S}_{tot} = O(nb)$.*

Remark 3.9. *A reader may notice a discrepancy between the existential result in Theorem 3.4 and the result in Theorem 3.8, in that our MPC algorithm achieves a slightly weaker bound for the size of $\mathcal{L}$, $|\mathcal{L}| \le O(n(\min\{b, \mathcal{S}\})^{-\frac{1}{5}}$ instead of $|\mathcal{L}| \le O(nb^{-\frac{1}{5}})$. We observe however that from the point of view of our applications, both results are equally good: since $\mathcal{S} = O(n^\delta)$, both results imply $|\mathcal{L}| \le O(nb^{-\Theta(1)})$, which is what is required in our analysis in the next sections.*

4 DETERMINISTIC CONNECTIVITY IN MPC

In this section we show the main result of this paper: derandomization of the algorithms of Andoni et al. [3] and Behnezhad et al. [13] as sketched in Section 2.3. First, we present a derandomized subroutine for reducing the number of vertices in a graph by a constant fraction in a constant number of rounds. The approach to this derandomization is similar to, but simpler than, our approach in Section 3, so we omit the details here.

Theorem 4.1. *Let $G = (V, E)$ be an arbitrary undirected simple graph with no isolated vertices. In a constant number of MPC rounds (with $\mathcal{S} = O(|V|^\delta)$ local space and $\mathcal{S}_{tot} = O(|V| + |E|)$ global space) one can deterministically construct an undirected simple graph $G^* = (V^*, E^*)$ such that*

- *G^* is obtained by a sequence of independent contractions of edges in G and*
- *$|V^*| \le \frac{99}{100}|V|$.*

We also introduce a brief remark which will be useful later:

Remark 4.2. *If $m < n \log^{10} n$ then the round complexity of repeatedly applying Theorem 4.1 until $m > n \log^{10} n$ is $\Theta(\log \log_{m/n} n)$.*

4.1 Deterministic Connectivity

Consider the connectivity algorithm of Andoni et al. [3] (see also Section 2.3). In each phase: there are n_i nodes; each node has its degree increased to at least b; the random leader contraction picks a set of $\widetilde{\Theta}(n_i/b)$ leaders (where $b = (\frac{m}{n_i})^{1/2}$) and then each node is contracted to some leader in its component. We first note that as a result of

Theorem 4.1 , we can reduce the number of vertices deterministically from n to $n/\mathrm{polylog}(n)$ in $O(\log \log n)$ rounds. This means that the precondition of Theorem 3.8, that $b > \log^{10} n$, is easily satisfiable in that many rounds.

We now modify the algorithm from [3] in the following way. In each phase, we increase the degrees of all vertices in the graph to at least $b = (\frac{m}{n_i})^{1/2}$, as before (keeping in mind that if a vertex is in a connected component of size at most b then we detected its connected component in this step, and so we drop it from any further analysis). We then use our deterministic leader contraction algorithm above (Theorem 3.8) to select and contract into leaders in such a way as to reduce the number of vertices, deterministically in a constant number of rounds, from n_i to $O(\frac{n_i}{(\min\{b, \mathcal{S}\})^{1/5}})$. This slightly weaker bound on the size of the set of leaders (comparing to $\widetilde{O}(n_i/b)$ in [3]) does not asymptotically affect the running time, and with $\mathcal{S} = n^{\Omega(1)}$, after $O(\log\log_{m/n} n)$ phases we can reach $b = O(n_i)$, after which the algorithm correctly determines all connected components. Therefore the random leader contraction phase in [3] can be substituted for our own from Theorem 3.8 (as long as we reduce the number of vertices by a factor of polylog(n) beforehand, recalling Theorem 4.1 and Remark 4.2) to obtain the following:

Theorem 4.3. *Let G be an undirected graph with n vertices, m edges, and diameter D. In $O(\log D \cdot \log\log_{m/n} n)$ rounds on an MPC (with $\mathcal{S} = O(n^\delta)$ local space and $\mathcal{S}_{tot} = O(m+n)$ global space) one can deterministically identify the connected components of G.*

While the main focus of this paper is on the linear global space regime, let us also mention that the algorithm of Andoni et al. [3, Theorem I.2] achieves a lower complexity when available global space increases. Since the only randomized step of the algorithm is leader contraction, we can use our deterministic leader contraction to obtain the following straightforward extension.

Theorem 4.4. *Let $0 \le \gamma \le \frac{1}{2}$ be arbitrary. Let G be an undirected graph with n vertices, m edges, and each connected component having diameter at most D. One can deterministically identify the connected components of G in $O\left(\min\{\log D \cdot \log(\frac{\log n}{2+\gamma \log n}), \log n\}\right)$ rounds on an MPC with $\mathcal{S} = O(n^\delta)$ local space and $\mathcal{S}_{tot} = O((m+n)^{1+\gamma})$ global space.*

Observe that when $\gamma = \Omega(1)$, that is when γ is larger than some positive constant, then Theorem 4.4 gives an asymptotically optimal round complexity of $O(\log D)$ rounds (conditioned on the 1-vs-2 cycles conjecture (Conjecture 1), see Theorems 5.1 and 5.2).

4.2 Faster Deterministic Connectivity

For randomized algorithms, Behnezhad et al. [13] extended the connectivity algorithm of Andoni et al. [3] (derandomized in Section 4.1) and improved the complexity to $O(\log D + \log\log_{m/n} n)$ MPC rounds, with high probability. In this section, we show how to combine the approach of Behnezhad et al. [13] with our deterministic leader contraction algorithm (Theorem 3.8) to get a deterministic $O(\log D + \log\log_{m/n} n)$-rounds MPC connectivity algorithm.

Similarly to our analysis of the connectivity algorithm of Andoni et al. [3] in Section 4.1, we observe that all randomized steps in the algorithm of Behnezhad et al. [13] can be reduced to the two procedures derandomized earlier in Theorem 3.8 and Theorem 4.1.

The first step of the randomized algorithm due to Behnezhad et al. [13] is to ensure that $m \ge n \log^{10} n$ in $O(\log\log_{m/n} n)$ rounds by contracting vertices if necessary; by Theorem 4.1 (see also Remark 4.2) this step can be implemented deterministically in $O(\log\log_{m/n} n)$ rounds on an MPC with $\mathcal{S} = O(n^\delta)$ local space and $\mathcal{S}_{tot} = O(m+n)$ global space. The remaining task is to determine connected components in a graph with n vertices and m edges with $m \ge n \log^{10} n$ in $O(\log D + \log\log_{m/n} n)$ rounds on an MPC with $\mathcal{S} = O(n^\delta)$ and $\mathcal{S}_{tot} = O(m)$.

As mentioned in Section 2.3, the main idea behind the speed up in [13] is to interleave the operations of degree expansions and vertex contractions for vertices of various degrees, while maintaining the total global space used bounded by $\mathcal{S}_{tot}$. For that, we quantify individual space available for any single vertex u by assigning to all vertices their *levels* $\ell(u)$ ($0 \le \ell(u) \le O(\log\log_{m/n} n)$) and associated *budgets* $b(u)$ (budget $b(u)$ controls how much space vertex u can use; in [13] the budget for vertices of level i is $\beta_i = \beta_0^{1.25^i}$, where $\beta_0 = (\frac{m}{n})^{1/2}$, though the analysis works for $\beta_i = (\beta_0)^{c^i}$ for any constant $c > 1$). Then, informally, Behnezhad et al. [13] showed how to ensure that every vertex u in a constant number of MPC rounds either increases its level (and hence its budget) or learns its entire 2-hop neighborhood. Since a vertex can increase its level only $O(\log\log_{m/n} n)$ times, once the budget of a vertex reaches n, the vertex is able to store all vertices in its allocated local space to determine its connected component. Similarly, learning a 2-hop neighborhood allows to perform edge contractions to (very informally) halve the diameter. With some additional arguments (see [13, 14] and also [47]), this leads to an $O(\log D + \log\log_{m/n} n)$-rounds MPC connectivity algorithm.

Algorithm 3: RelabelIntraLevel (G, b, ℓ) (Randomized algorithm from [13, p. 1620])

Input: $b(u)$ denotes the *budget* and $\ell(u)$ the *level* of vertex u

```
1 Mark an active vertex $v$ as "saturated" if it has at least $b(v)$
    active neighbors that have the same level as $v$.
2 If an active vertex $v$ has a neighbor $u$ with $\ell(u) = \ell(v)$ that
    is marked as saturated, $v$ is also marked as saturated.
3 Mark every saturated vertex $v$ as a "leader" independently
    with probability $\min\{\frac{3\log n}{b(v)}, 1\}$.
4 For every leader vertex $v$, set $\ell(v) := \ell(v) + 1$ and
    $b(v) := b(v)^{1.25}$.
5 Every non-leader saturated vertex $v$ that sees a leader vertex
    $u$ of the same level (i.e., $\ell(u) = \ell(v)$) in its 2-hop (i.e.,
    $\mathrm{dist}(v, u) \le 2$), chooses one as its leader arbitrarily.
6 Every vertex is contracted to its leader. That is, for any
    non-leader vertex $v$ with leader $u$, every edge $\{v, w\}$ is
    replaced with an edge $\{u, w\}$. Then remove vertex $v$ from
    the graph.
7 Remove duplicate edges or self-loops and remove
    saturated/leader flags from the vertices.
```

We remark that Algorithm 3 RelabelIntraLevel is the only subroutine from the main connectivity algorithm of [13] which requires derandomization (all others are deterministic). RelabelIntraLevel works solely with a subset of vertices that we call *active* and it is the only procedure that increases the budgets and levels.

In RelabelIntraLevel, the only randomized step is Step 3, which takes some subset of vertices and marks each vertex v as a "leader" independently at random with probability $\min\{\frac{3\log n}{b(v)}, 1\}$. All selected leaders are then promoted to the next level ($\ell(v) := \ell(v)+1$ in Step 4), and all nodes which have a leader of the same level within 2 hops contract to such a leader arbitrarily (Steps 5–6). The entire analysis of the algorithm of Behnezhad et al. [13] relies on this randomized process.

For our analysis it is useful to observe that the "leader contraction" processes in RelabelIntraLevel as described above are *independent for each level*: two vertices which do not start RelabelIntraLevel with the same level do not interact with each other (edges between them are irrelevant to the execution of RelabelIntraLevel).

There are two principal technical challenges to consider when derandomizing this process. Firstly, our deterministic leader contraction process for a given level i only reduces the number of vertices from N_i to $O(\frac{N_i}{(\min\{\beta_i,\mathcal{S}\})^{1/5}})$ ([13] reduces to $O(\frac{N_i \log n}{\beta_i})$), and unlike the connectivity algorithm of [3], we can no longer rely on the "phase" structure (and the framework of double exponential speed problem size reduction) to overcome this. Secondly, vertices of multiple levels might be performing leader contractions simultaneously, and we need to ensure that we can perform these leader contractions in parallel and in linear global space. We address these issues in the following two lemmas. The proofs of these lemmas require a careful examination of the analysis of the connectivity algorithm of [13], arguing that our derandomization of the randomized subroutines does not affect important properties such as the total global space used and the maximum level of a vertex. We defer the proofs to the full version of the paper due to space constraints.

Lemma 4.5. *Let S_i denote the set of saturated vertices at level i after Step 2 of RelabelIntraLevel, let L_i denote the set of selected leaders at level i after Step 3 of the same execution of RelabelIntraLevel, let β_i denote the budget of vertices at level i, let $b(v)$ denote the budget of vertex v, and let γ be an arbitrary constant such that $0 < \gamma \le 1$. If we make the following modifications to RelabelIntraLevel:*

- *set $\beta_{i+1} := \beta_i \cdot (\min\{\beta_i, \mathcal{S}\})^{\gamma/4}$,*
- *replace Step 3 of RelabelIntraLevel with any MPC algorithm that in $O(1)$ rounds selects $|L_i| = O(\frac{|S_i|}{(\min\{\beta_i,\mathcal{S}\})^{\gamma}})$ leaders for each level i with high probability or deterministically, and*
- *replace the budget update rule in Step 4 of RelabelIntraLevel with $b(v) := b(v) \cdot (\min\{b(v), \mathcal{S}\})^{\gamma/4}$,*

then the connectivity algorithm of [13] remains correct with the same asymptotic complexity.

Lemma 4.6. *Let S_i denote the set of saturated vertices at level i after Step 2 of an execution of RelabelIntraLevel (in [13]), and let β_i denote the budget of vertices at level i.*

Copies of Algorithm 2 can be run in parallel, for each possible level and in a constant number of MPC rounds, to deterministically select $O(|S_i| \cdot (\min\{\beta_i, \mathcal{S}\})^{-\frac{1}{5}})$ leaders for each level.

By combining Lemmas 4.5 and 4.6, we can conclude that using a deterministic leader contraction algorithm which selects $O(\frac{|S_i|}{\min\{\beta_i,\mathcal{S}\}^{\gamma}})$ leaders for a constant $0 < \gamma \le 1$, for all levels, in a constant number of rounds, does not affect the proof of correctness or the asymptotic running time of the connectivity algorithm of [13]. Further, by Theorem 3.8, Algorithm 2 can be run in parallel for all levels in a constant number of rounds and be implemented on MPC for $\gamma = \frac{1}{5}$. We combine these with Theorem 4.1, and recalling Remark 4.2 we obtain the following.

Theorem 4.7. *Let G be an undirected graph with n vertices, m edges, and diameter D. In $O(\log D + \log\log_{m/n} n)$ rounds on an MPC (with $\mathcal{S} = O(n^{\delta})$ local space and $\mathcal{S}_{tot} = O(m+n)$ global space) one can deterministically identify the connected components of G.*

Let us also notice that as in [13], our algorithm does not require prior knowledge of D.

Similarly as in Theorem 4.4, one can easily (using the same approach as above) incorporate our derandomization framework to the algorithm of Behnezhad et al. [13] with superlinear global space to obtain the following theorem.

Theorem 4.8. *Let $0 \le \gamma \le \frac{1}{2}$ be arbitrary. Let G be an undirected graph with n vertices, m edges, and each connected component having diameter at most D. One can deterministically identify the connected components of G in $O(\log D + \log(\frac{\log n}{2+\gamma \log n}))$ rounds on an MPC with $\mathcal{S} = O(n^{\delta})$ local space and $\mathcal{S}_{tot} = O((m+n)^{1+\gamma})$ global space.*

As in Theorem 4.4, for $\gamma = \Omega(1)$, Theorem 4.8 gives an asymptotically optimal round complexity of $O(\log D)$ rounds (conditioned on the 1-vs-2 cycles conjecture 1, see Theorems 5.1 and 5.2).

5 LOWER BOUND FOR CONNECTIVITY

It is known that with sufficiently large local space per machine one can determine all connected components of a graph in a constant number of MPC rounds, even deterministically (see, e.g., [43, 53]). However, we do not expect the same result for MPCs in the $O(n^{\delta})$ local space regime: we seem to require $\Omega(\log n)$ MPC rounds for even 2-regular graphs, as stated in the 1-vs-2 cycles conjecture (see Section 2.2.1, Conjecture 1). However, the 1-vs-2 cycles conjecture only claims that the connectivity problem is hard for graphs with large diameter and in does not preclude that for graphs with $o(n)$ diameter this problem is *much easier*. For example, one could hope that for graphs with low diameter, say, $D = \text{polylog}(n)$ or $D = n^{1-\Omega(1)}$, a constant rounds MPC algorithm in the n^{δ} space regime exists. Behnezhad et al. [13] partially quashed this possibility and proved that unless the 1-vs-2 cycles conjecture fails, if $D \ge \log^{1+\delta} n$ then the best we could hope for is $\Omega(\log D)$ MPC rounds.

Theorem 5.1. **[13]** *Let δ and γ be arbitrary constants $0 < \beta, \gamma < 1$. Let $D \ge \log^{1+\gamma} n$. Consider an MPC with local space $\mathcal{S} = O(n^{\beta})$ and global space $\mathcal{S}_{tot} = \Omega(n+m)$. Any MPC algorithm that with probability at least $1 - \frac{1}{n^3}$ determines each connected component of any given n-vertex graph with diameter at most D requires $\Omega(\log D)$ rounds (on an MPC with local space $\mathcal{S}$ and global space $\mathcal{S}_{tot}$), unless there is an MPC algorithm that with probability at least $1 - \frac{1}{n}$ distinguishes in $o(\log n)$ rounds (on an MPC with local space $\mathcal{S}$ and global space $\mathcal{S}_{tot}$) whether the input graph is an n-vertex cycle or consists of two $\frac{n}{2}$-vertex cycles.*

Rather surprisingly, the arguments in Theorem 5.1 require that D is large, at least $\log^{1+\Omega(1)} n$. We complement Theorem 5.1 by extending the analysis to arbitrarily small values of D as follows.

Theorem 5.2. *Let δ be an arbitrary constant $0 < \delta < 1$ and let $D \le \log^{3/2} n$. Consider an MPC with local space $\mathcal{S} = O(n^{\delta})$ and global space $\mathcal{S}_{tot} = \Omega(n+m)$. Any MPC algorithm that with probability at least $1 - \frac{1}{n^3}$ for any given n-vertex graph G correctly determines all connected components of G with diameter at most D requires $\Omega(\log D)$ rounds (on an MPC with local space $\mathcal{S}$ and global space $\mathcal{S}_{tot}$), unless there is an MPC algorithm that with probability at least $1 - \frac{1}{n}$ distinguishes in $o(\log n)$ rounds (on an MPC with local space $\mathcal{S}$ and global space $\mathcal{S}_{tot}$) if the input graph is an n-vertex cycle or consists of two $\frac{n}{2}$-vertex cycles.*

Remark 5.3. *Observe that we do not prove exactly the same claim as [13] in Theorem 5.1 for small values of D, since we consider a slightly different problem: we require that the algorithm correctly determines all connected components with diameter at most D, and hence, the input to the algorithm can have larger diameter; we just ignore the output of larger connected components.*

Our approach uses similar arguments to those in [13]: we show that if we are given as a black-box an MPC algorithm that with high probability correctly determines all connected components of G with diameter at most D in $o(\log D)$ rounds (on an MPC with local space $\mathcal{S}$ and $\mathcal{M}$ machines), we can use this algorithm to distinguish in $o(\log n)$ rounds (on an MPC with local space $\mathcal{S}$ and $\mathcal{M}$ machines) if the input graph is an n-vertex cycle or two $\frac{n}{2}$-vertex cycles.

The proof of Theorem 5.2 is by contradiction. Suppose that there is an algorithm ALG^* that for a given graph G on n vertices determines with probability at least $1 - \frac{1}{n^3}$ all connected components of G with diameter at most D in $o(\log D)$ MPC rounds with local space $\mathcal{S} = n^\delta$ and on $\mathcal{M}$ machines. We will show how to use ALG^* to disprove the 1-vs-2 cycles Conjecture 1 and design an algorithm that in $o(\log n)$ rounds on an MPC with local space $\mathcal{S} = n^\delta$ and $\mathcal{M}$ machines, with probability at least $1 - \frac{1}{n}$ distinguishes whether the input graph is an n-vertex cycle or consists of two $\frac{n}{2}$-vertex cycles.

The idea is as follows. Let $G = (V, E)$ either be an n-vertex cycle or consists of two disjoint $\frac{n}{2}$-vertex cycles. We will use ALG^* to distinguish between these two cases, when G is an n-vertex cycle and when G consists of two disjoint $\frac{n}{2}$-vertex cycles, by repeatedly applying the following procedure (see Algorithm 4):

(i) split each cycle into paths of length *typically* at most D by "temporary" removing some edges,
(ii) determine all vertices on each path (i.e., in each connected component) using ALG^*,
(iii) contract each path into a single vertex, and
(iv) reverse the temporary removal of the edges to obtain a smaller representation of the original graph.

The underlying idea is that if we started with graph G on n-vertices, then steps *(i)*, *(iii)*, and *(iv)* can be performed in a constant number of MPC rounds, step *(ii)* can be done in $o(\log D)$ MPC rounds (using ALG^*), and we reduced the problem to an instance on $\widetilde{O}(n/D^{\Theta(1)})$ vertices. Thus, in short, after $o(\log D)$ MPC rounds we reduce the size from n to $O(n/D^{\Theta(1)})$, then we repeat this step the entire graph can be stored on a single MPC machine, in which case the problem can be solved in a single MPC round. Since this approach will reduce G into a graph on at most n^δ vertices in $O(\log_D n)$ repetitions, this will give an algorithm that can determine whether the original G is an n-vertex cycle or consists of two disjoint $\frac{n}{2}$-vertex cycles in $o(\log D) \cdot O(\log_D n) = o(\log n)$ MPC rounds, contradicting the 1-vs-2 cycles Conjecture 1.

The main differences between the approach from [13] and Algorithm 4 is that in step 1, we have the probability of dropping an edge $p = \frac{1}{\sqrt{D}}$ independent of n, and that (because of the change in step 1) we use ALG^* that will process all connected components of diameter at most D, but which also allows as its input some connected components with larger diameter. (The reason the arguments from [13] were weaker in this step is that the proof wanted to ensure that with high probability *all* components have diameter $O(\log D)$, and

```
Algorithm 4: Reducing length of each cycle
  Input: Graph $H$ with maximum degree at most 2;
         parameters $n$ and $D$
1 Temporarily remove each edge in $H$ with probability
    $p = \frac{1}{\sqrt{D}}$, obtaining graph $H^*$
2 Find all connected components of $H^*$ using algorithm $\text{ALG}^*$
3 Determine the correctness of all connected components in
    $H^*$ returned by $\text{ALG}^*$
4 Contract each connected component of $H^*$ correctly
    classified by $\text{ALG}^*$ to a single vertex in $H$ and return the
    obtained graph (including all edges removed in step 1)
```

this requires some dependency on n; we avoid such dependency.) The use of step 3 is to determine all connected components in H^* of diameter at most D, which are contract into a single vertex in H, but to possibly do nothing to vertices in larger connected components in H^*, which are likely to be misclassified by ALG^*. The idea here is as follows: Consider an arbitrary connected component C of H^* and consider the output of ALG^* on C. If ALG^* correctly classifies all vertices in C (as being in the same connected component and disjoint from the rest of H^*) then all these vertices will be contracted to a single vertex in H; otherwise, all edges from C will stay in H.

The following claim describes key properties of Algorithm 4 (see the full version for a proof).

Claim 5.4. *If the input graph H in Algorithm 4 has maximum degree at most 2, has $N \le n$ vertices, and each connected component is of size greater than n^δ, and if $9 < D \le \log^{3/2} n$, then with probability at least $1 - \frac{1}{n^2}$ the following three conditions hold:*

(i) the total number of vertices in all connected components of H^* of diameter greater than D is at most $\frac{N}{\sqrt{D}}$,
(ii) Algorithm 4 runs in $o(\log D)$ MPC rounds,
(iii) the graph returned by Algorithm 4 has at most $\frac{3N}{\sqrt{D}}$ edges.

With Claim 5.4 at hand, we are now ready to prove Theorem 5.2.

Proof of Theorem 5.2: The claim is trivial for $D \le 9$ and so let us assume that $D > 9$.

We start with G with n vertices and with the promise that either G is an n-vertex cycle or G consists of two $\frac{n}{2}$-vertex cycles. We apply Algorithm 4 repeatedly to G, so that by Claim 5.4, after t repetitions, with probability at least $1 - \frac{t}{n^2}$,

(i) the repetitions can be implemented in $o(t \cdot \log D)$ MPC rounds, and
(ii) either G was contracted to a graph that has a connected component with at most n^δ vertices,
(iii) or G was contracted to a graph with at most $n \cdot \left(\frac{3}{\sqrt{D}}\right)^t$ edges and vertices.

This implies that if $D > 9$, then for any $t \ge \frac{(1-\delta)\log n}{\log(\sqrt{D}/3)}$ we have a guarantee (with probability at least $1 - \frac{t}{n^2}$) that after at most t repetitions the contracted graph fits a single MPC machine, and then we can determine whether the contracted graph is a single cycle or two cycles, determining whether G is an n-vertex cycle or two $\frac{n}{2}$-vertex cycles. Observe that our choice of t ensures that the algorithm runs (with probability at least $1 - \frac{t}{n^2}$) in $o\left(\frac{\log n}{\log D} \cdot \log D\right) \equiv o(\log n)$ rounds on an MPC with local space $\mathcal{S} = n^{\delta}$.

This implies that if ALG* can take any graph on at most n vertices and then determine with probability at least $1 - \frac{1}{n^3}$ all connected components with diameter at most D in $o(\log D)$ MPC rounds with local space $\mathcal{S} = n^{\delta}$ and global space $\mathcal{S}_{tot}$, then the scheme above can use ALG* to design an MPC algorithm that in $o(\log n)$ rounds on an MPC with local space $\mathcal{S} = n^{\delta}$ and global space $\mathcal{S}_{tot}$, with probability at least $1 - \frac{1}{n}$ distinguishes whether the input graph is an n-vertex cycle or consists of two $\frac{n}{2}$-vertex cycles. This completes the proof of Theorem 5.2. □

6 CONCLUSIONS

In this paper we show that the novel, powerful technique of $o(\log n)$-rounds randomized connectivity MPC algorithms for graphs with low diameter due to Andoni et al. [3] and Behnezhad et al. [13] can be efficiently derandomized without any asymptotic loss.

In addition to the main concrete result of this paper, Theorem 4.7, we believe that the main take-home message of this work is that many powerful randomized MPC algorithms can be efficiently derandomized in the MPC setting, even with low local space and optimal global space utilization. We have already seen in the past some examples demonstrating the strength of deterministic MPC, and we hope that our work contributes to the advances in this area, showing that even such fundamental problem as graph connectivity can be solved deterministically as good as the state-of-the-art randomized algorithms. One interesting feature of our work is that the derandomization techniques incorporated in our work are highly non-component-stable. As the result, our work is another example (see also [24, 26, 27]) showing that the powerful framework of conditional MPC lower bounds due to Ghaffari et al. [37] (which as for now, is arguably the most general framework of lower bounds known for MPC algorithms) is not always suitable, and a lower bound (even if only conditioned on the 1-vs-2 cycles conjecture) for component-stable MPC algorithms may not preclude the existence of more efficient non-component-stable MPC algorithms.

We believe that the techniques developed in our paper are versatile and that our framework of deterministic MPC algorithms can be applied more broadly in similar settings. For example, in addition to the bounds mentioned in Section 1.1 concerning spanning forest, MST, and bottleneck spanning tree, our framework could also be applied to approximate minimum spanning forest algorithms (see Andoni et al. [3, Theorem I.8]), but we hope that our approach will find many more applications. In particular, we would not be surprised if a number of fundamental graph problems could be solved (even deterministically) in $\widetilde{O}(\log D)$ MPC rounds.

Finally, the recent advances in the randomized algorithms for the connectivity in the MPC model led also to similar results for the classic PRAM model: Liu, Tarjan, and Zhong [47] gave an $O(\log D + \log\log_{m/n} n)$-time randomized algorithm on an ARBITRARY CRCW PRAM using $O(m)$ processors whp. It is an interesting open problem whether a similar bound can be achieved deterministically. The approach present in our paper seems to require too many resources to achieve this task.

ACKNOWLEDGEMENTS

The second author would like to thank Peter Davies and Merav Parter for many fascinating conversations about derandomization of MPC algorithms and their helpful input.

REFERENCES

[1] Noga Alon and Joel H. Spencer. 2016. *The Probabilistic Method* (4th ed.). John Wiley & Sons, New York, NY. https://doi.org/10.1002/9780470277331
[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).* ACM Press, New York, NY, 574–583. https://doi.org/10.1145/2591796.2591805
[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).* IEEE Computer Society Press, Los Alamitos, CA, 674–685. https://doi.org/10.1109/FOCS.2018.00070 Also in [4].
[4] Alexandr Andoni, Clifford Stein, Zhao Song, Zhengyu Wang, and Peilin Zhong. 2018. Parallel Graph Connectivity in Log Diameter Rounds. *CoRR, arXiv* abs/1805.03055 (2018). Also in [3].
[5] 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).* ACM Press, New York, NY, 322–335. https://doi.org/10.1145/3357713.3384321
[6] 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).* SIAM, Philadelphia, PA, 1616–1635. https://doi.org/10.1137/1.9781611975482.98
[7] 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).* ACM Press, New York, NY, 461–470. https://doi.org/10.1145/3293611.3331596
[8] 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).* ACM Press, New York, NY, 243–252. https://doi.org/10.1145/3382734.3404504
[9] 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).* ACM Press, New York, NY, 273–284. https://doi.org/10.1145/2463664.2465224
[10] Paul Beame, Paraschos Koutris, and Dan Suciu. 2017. Communication Steps for Parallel Query Processing. *J. ACM* 64, 6 (2017), 40:1–40:58. https://doi.org/10.1145/3125644
[11] 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).* ACM Press, New York, NY, 481–490. https://doi.org/10.1145/3293611.3331609
[12] Soheil Behnezhad, Mahsa Derakhshan, and MohammadTaghi Hajiaghayi. 2018. Brief Announcement: Semi-MapReduce Meets Congested Clique. *CoRR, arXiv* abs/1802.10297 (2018). https://doi.org/10.48550/arXiv.1802.10297
[13] 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).* IEEE Computer Society Press, Los Alamitos, CA, 1615–1636. https://doi.org/10.1109/FOCS.2019.00095 Also in [14].
[14] Soheil Behnezhad, Laxman Dhulipala, Hossein Esfandiari, Jakub Łącki, and Vahab S. Mirrokni. 2019. Near-Optimal Massively Parallel Graph Connectivity. *CoRR, arXiv* abs/1910.05385 (2019). Also in [13].
[15] 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).* ACM Press, New York, NY, 59–68. https://doi.org/10.1145/3470631
[16] 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).* IEEE Computer Society Press, Los Alamitos, CA, 1637–1649. https://doi.org/10.1109/FOCS.2019.00096
[17] Bonnie Berger, John Rompel, and Peter W. Shor. 1989. Efficient NC Algorithms for Set Cover with Applications to Learning and Geometry. In *Proceedings of the 30th IEEE Symposium on Foundations of Computer Science (FOCS).* IEEE Computer Society, Los Alamitos, CA, 54–59. https://doi.org/10.1109/SFCS.1989.63455
[18] Sebastian Brandt, Rustam Latypov, and Jara Uitto. 2021. Brief Announcement: Memory Efficient Massively Parallel Algorithms for LCL Problems on Trees. In

Proceedings of the 35th International Symposium on Distributed Computing (DISC). LIPIcs, Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 50:1–50:4.

[19] 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)*, Vol. 91. LIPIcs, Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 11:1–11:16. https://doi.org/10.4230/LIPIcs.DISC.2017.11 Journal version appeared in [20].

[20] Keren Censor-Hillel, Merav Parter, and Gregory Schwartzman. 2020. Derandomizing Local Distributed Algorithms under Bandwidth Restrictions. *Distributed Computing* 33, 3-4 (2020), 349–366. https://doi.org/10.1007/s00446-020-00376-1

[21] 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)*. ACM Press, New York, NY, 471–480.

[22] Moses Charikar, Weiyun Ma, and Li-Yang Tan. 2021. Brief Announcement: A Randomness-Efficient Massively Parallel Algorithm for Connectivity. In *Proceedings of the 40th ACM Symposium on Principles of Distributed Computing (PODC)*. ACM Press, New York, NY, 431–433. https://doi.org/10.1145/3465084.3467951

[23] Ka Wong Chong, Yijie Han, and Tak Wah Lam. 2001. Concurrent Threads and Optimal Parallel Minimum Spanning Trees Algorithm. *J. ACM* 48, 2 (2001), 297–323. https://doi.org/10.1145/375827.375847

[24] Artur Czumaj, Peter Davies, and Merav Parter. 2020. Graph Sparsification for Derandomizing Massively Parallel Computation with Low Space. In *Proceedings of the 32nd Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*. ACM Press, New York, NY, 175–185.

[25] 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)*. ACM Press, New York, NY, 309–318.

[26] Artur Czumaj, Peter Davies, and Merav Parter. 2021. Component Stability in Low-Space Massively Parallel Computations. In *Proceedings of the 40th ACM Symposium on Principles of Distributed Computing (PODC)*. ACM Press, New York, NY, 481–491. https://doi.org/10.1145/3465084.3467903

[27] Artur Czumaj, Peter Davies, and Merav Parter. 2021. Improved Deterministic $(\Delta+1)$-Coloring in Low-space MPC. In *Proceedings of the 40th ACM Symposium on Principles of Distributed Computing (PODC)*. ACM Press, New York, NY, 469–479. https://doi.org/10.1145/3465084.3467937

[28] 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)*. ACM Press, New York, NY, 471–484.

[29] Andrzej Czygrinow, Michal Hańćkowiak, and Wojciech Wawrzyniak. 2008. Fast Distributed Approximations in Planar Graphs. In *Proceedings of the 22nd International Symposium on Distributed Computing (DISC) (Lecture Notes in Computer Science, Vol. 5218)*. Springer Verlag, Berlin, Heidelberg, 78–92.

[30] Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: Simplified Data Processing on Large Clusters. *Commun. ACM* 51, 1 (January 2008), 107–113. https://doi.org/10.1145/1327452.1327492

[31] 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)*. ACM Press, New York, NY, 94–103. https://doi.org/10.1145/3293611.3331626

[32] Paul Erdös and John L. Selfridge. 1973. On a Combinatorial Game. *Journal of Combinatorial Theory, Series A* 14, 3 (1973), 298–301.

[33] Guy Even, Oded Goldreich, Michael Luby, Noam Nisan, and Boban Veličković. 1992. Approximations of General Independent Distributions. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing (STOC)*. ACM Press, New York, NY, 10–16. https://doi.org/10.1145/129712.129714

[34] Guy Even, Oded Goldreich, Michael Luby, Noam Nisan, and Boban Veličković. 1998. Efficient Approximation of Product Distributions. *Random Structures and Algorithms* 13, 1 (1998), 1–16. A preliminary version appeared in [33].

[35] 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)*. ACM Press, New York, NY, 129–138. https://doi.org/10.1145/3212734.3212743

[36] 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)*. LIPIcs, Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 29:1–29:17.

[37] 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)*. IEEE Computer Society Press, Los Alamitos, CA, 1650–1663.

[38] 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)*. SIAM, Philadelphia, PA, 1636–1653. https://doi.org/10.1137/1.9781611975482.99

[39] 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) (Lecture Notes in Computer Science, Vol. 7074)*. Springer Verlag, Berlin, Heidelberg, 374–383.

[40] 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.

[41] Tomasz Jurdziński and Krzysztof Nowicki. 2018. MST in $O(1)$ Rounds of Congested Clique. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, Philadelphia, PA, 2620–2632. https://doi.org/10.1137/1.9781611975031.167

[42] 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)*. SIAM, Philadelphia, PA, 938–948. https://doi.org/10.1137/1.9781611973075.76

[43] 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)*. ACM Press, New York, NY, 85–94. https://doi.org/10.1145/1989493.1989505

[44] Jakub Łącki, Vahab S. Mirrokni, and Michał Włodarczyk. 2018. Connected Components at Scale via Local Contractions. *CoRR, arXiv* abs/1807.10727 (2018).

[45] 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)*. ACM Press, New York, NY, 364–377.

[46] Christoph Lenzen and Roger Wattenhofer. 2008. Leveraging Linial's Locality Limit. In *Proceedings of the 22nd International Symposium on Distributed Computing (DISC) (Lecture Notes in Computer Science, Vol. 5218)*. Springer Verlag, Berlin, Heidelberg, 394–407. https://doi.org/10.1007/978-3-540-87779-0_27

[47] Sixue Cliff Liu, Robert E. Tarjan, and Peilin Zhong. 2020. Connected Components on a PRAM in Log Diameter Time. In *Proceedings of the 32nd Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*. ACM Press, New York, NY, 359–369. https://doi.org/10.1145/3350755.3400249 A preliminary version appeared in [48].

[48] S. Cliff Liu, Robert E. Tarjan, and Peilin Zhong. 2020. Connected Components on a PRAM in Log Diameter Time. *CoRR, arXiv* abs/2003.00614 (2020). Conference version appeared in [47].

[49] Michael Luby. 1993. Removing Randomness in Parallel Computation without a Processor Penalty. *J. Comput. System Sci.* 47, 2 (1993), 250–286.

[50] Rajeev Motwani, Joseph Naor, and Moni Naor. 1994. The Probabilistic Method Yields Deterministic Parallel Algorithms. *J. Comput. System Sci.* 49, 3 (1994), 478–516. https://doi.org/10.1016/S0022-0000(05)80069-8

[51] Rajeev Motwani and Prabhakar Raghavan. 1995. *Randomized Algorithms.* Cambridge University Press, New York, NY.

[52] Danupon Nanongkai and Michele Scquizzato. 2022. Equivalence Classes and Conditional Hardness in Massively Parallel Computations. *Distributed Computing* 35, 2 (2022), 165–183. https://doi.org/10.1007/s00446-021-00418-2

[53] Krzysztof Nowicki. 2021. A Deterministic Algorithm for the MST Problem in Constant Rounds of Congested Clique. In *Proceedings of the 53rd Annual ACM Symposium on Theory of Computing (STOC)*. ACM, New York, NY, 1154–1165.

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

[55] Prabhakar Raghavan. 1988. Probabilistic Construction of Deterministic Algorithms: Approximating Packing Integer Programs. *J. Comput. System Sci.* 37, 2 (1988), 130–143. https://doi.org/10.1016/0022-0000(88)90003-7

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

[57] Jukka Suomela. 2013. Survey of Local Algorithms. *Comput. Surveys* 45, 2 (Feb. 2013), 24:1–24:40. https://doi.org/10.1145/2431211.2431223

[58] Tom White. 2015. *Hadoop: The Definitive Guide: Storage and Analysis at Internet Scale* (4th ed.). O'Reilly Media, Sebastopol, CA.

[59] 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)*. USENIX Association, Boston, MA.