Brief Announcement: A Randomness-efficient Massively Parallel Algorithm for Connectivity

Moses Charikar
Stanford University
Stanford, CA, USA
moses@cs.stanford.edu

Weiyun Ma
Stanford University
Stanford, CA, USA
wyma@cs.stanford.edu

Li-Yang Tan
Stanford University
Stanford, CA, USA
liyang@cs.stanford.edu

ABSTRACT

We give a randomness-efficient Massively Parallel Computation (MPC) algorithm for deciding whether an undirected graph is connected. For Connectivity on n-vertex, m-edge graphs whose components have diameter at most $D = 2^{o(\log n/\log\log n)}$, our algorithm runs in $R = O(\log D + \log\log_{m/n} n)$ rounds and uses a total of $(\log n)^{O(R)}$ random bits, $O(m)$ machines, and $n^{1-\Omega(1)}$ space per machine with good probability.[1] Our algorithm achieves a super-polynomial saving in randomness complexity as compared to the breakthrough algorithm of Andoni et al. (FOCS '18) and the subsequent improvement by Behnezhad et al. (FOCS '19). Our algorithm has the same round complexity as that of Behnezhad et al., but uses more total space.

Our Connectivity algorithm is an instantiation of a general method we develop for converting randomized algorithms in the PRAM model to highly randomness-efficient MPC algorithms. We show that for $k = o(\log n/\log\log n)$ and $p = n^{O(1)}$, any time-k p-processor randomized PRAM algorithm computing a function on n input bits can be converted to an equivalent strongly sublinear MPC algorithm with $O(k)$ rounds and only a total of $(\log n)^{O(k)}$ random bits. Our Connectivity algorithm follows from applying this method to the recent CRCW PRAM algorithm of Liu, Tarjan, and Zhong (SPAA '20).

Our approach is based on the design of a *pseudorandom generator* for PRAM algorithms. The analysis of our generator is built on classic and influential results in circuit complexity (Håstad '86; Nisan and Wigderson '88), which we generalize from the setting of small-depth circuits to the more powerful setting of PRAM algorithms. The parameters that we achieve are optimal given the current state of the art in complexity theory, in the sense that further improvements will imply $\mathsf{P} \neq \mathsf{NC}^1$.

CCS CONCEPTS

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

[1] With *good probability* means with probability at least $1 - 1/\text{poly}((m \log n)/n)$, which is the same as in Liu, Tarjan, and Zhong (SPAA '20).

PODC '21, July 26–30, 2021, Virtual Event, Italy

ACM ISBN 978-1-4503-8548-0/21/07.
https://doi.org/10.1145/3465084.3467951

KEYWORDS

massively parallel computation, connectivity, PRAM, pseudorandom generators, average-case lower bounds

ACM Reference Format:
Moses Charikar, Weiyun Ma, and Li-Yang Tan. 2021. Brief Announcement: A Randomness-efficient Massively Parallel Algorithm for Connectivity. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing (PODC '21), July 26–30, 2021, Virtual Event, Italy.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3465084.3467951

1 INTRODUCTION

The *Massively Parallel Computation* (MPC) model [4, 13] has recently emerged as the standard framework for capturing the capabilities of modern parallel computing platforms such as MapReduce, Hadoop, and Spark. In this model a set of machines perform computation synchronously in rounds. In each round, each machine can perform a local computation on the messages it receives from the previous round, and send the computation results to other machines as input for the subsequent round. We work in the *strongly sublinear* MPC regime where the local *space* of any machine, i.e. the total size of the messages received or sent by the machine in a round, is at most $O(N^\epsilon)$ for a small constant $0 < \epsilon < 1$, where N is the size of the input.

In this work, we consider the problem of *graph connectivity* in the MPC model, which is a fundamental problem that has been intensively studied in parallel computation. A recent breakthrough of Andoni et al. [1] gave a $O(\log D \log\log_{m/n} n)$-round randomized MPC algorithm for Connectivity on n-vertex, m-edge, D-diameter graphs that uses $O(m)$ total space, breaking a longstanding $O(\log n)$-round barrier. If $O(m^{1+\Omega(1)})$ total space is allowed, their algorithm runs in $O(\log D)$ rounds. This was followed by the work of Behnezhad et al.[5], who improved the round complexity to $O(\log D + \log\log_{m/n} n)$ while using $O(m)$ total space.

1.1 This work: Randomness-efficient MPC algorithm for Connectivity

This work focuses on the *randomness efficiency* of randomized algorithms. The goal of reducing the randomness usage of randomized algorithms is not only of basic theoretical interest, but also well-motivated from a practical point of view since high-quality random bits can be difficult or costly to obtain. Recently, there has been a surge of interest in improving the randomness efficiency of algorithms for various problems in different models of parallel and distributed computation [2, 6–8, 10, 11, 19].

We are primarily interested in designing randomness-efficient algorithms in the MPC model. For Connectivity, the aforementioned randomized MPC algorithms of [1, 5] use $\Omega(n)$ random bits. Our main result is the following:

Theorem 1 (Randomness-efficient MPC algorithm for Connectivity)**.** *For $D = 2^{o(\log n/\log\log n)}$, Connectivity on n-vertex, m-edge, D-diameter graphs can be computed by a strongly sublinear randomized MPC algorithm that runs in $R = O(\log D + \log\log_{m/n} n)$ rounds and uses a total of $(\log n)^{O(R)}$ random bits and $O(m)$ machines with good probability.*

We note that for $D = 2^{o(\log n/\log\log n)}$, the total number of random bits used by our algorithm is $(\log n)^{O(R)} = o(n^{\delta})$ for any arbitrarily small constant $\delta > 0$. Thus our algorithm achieves a super-polynomial saving in randomness complexity as compared to the algorithms of [1, 5].

For $D = \Omega(\log_{m/n} n)$, the round complexity of our algorithm matches that of the algorithm of [1] which runs in $O(\log D)$ rounds when $O(m^{1+\Omega(1)})$ total space is allowed. Our algorithm also has the same round complexity as that of [5], while the latter uses only $O(m)$ total space.

Our main result (Theorem 1) follows from a general method that we develop for converting a randomized algorithm in the *Parallel Random Access Machine* (PRAM) model into an equivalent MPC algorithm that is highly randomness-efficient and has round complexity asymptotically the same as the time of the PRAM. In the PRAM model, a set of processors run synchronously in time steps and have access to a shared memory. In each time step, each processor can read a cell in the shared memory, write to a cell in the shared memory, or perform its own computation locally. It is known that every PRAM algorithm can be efficiently simulated by an MPC algorithm [9, 13]. We show the stronger statement that we can efficiently simulate small-time randomized PRAM algorithms by MPC algorithms that are also randomness-efficient:

Theorem 2 (Converting randomized PRAM algorithms to randomness-efficient MPC algorithms)**.** *Let M be a randomized PRAM with k time steps, p processors, and c memory cells that computes a function $f : \{0,1\}^n \to \{0,1\}$, where each processor generates at most r independent random bits in each time step. Assume that $p, c \le n^{O(1)}$, $r \le (\log n)^{O(1)}$, and $k = o(\log n/\log\log n)$. Then f can be computed by a strongly sublinear randomized MPC algorithm that runs in $O(k)$ rounds and uses $O(p + c)$ machines and only a total of $(\log n)^{O(k)}$ random bits.*

Our method applies to all versions of PRAM, particularly the most powerful *concurrent read concurrent write* version of PRAM (CRCW PRAM), which allows for multiple processors to read and write to the same memory cell at the same time.[2] In contrast with prior work which mostly focused on obtaining randomness-efficient versions of *specific* parallel algorithms, a key feature of Theorem 2 is that it applies to general PRAM algorithms.

We obtain Theorem 1 by applying Theorem 2 to a very recent randomized CRCW PRAM algorithm by Liu, Tarjan, and Zhong [16], which computes Connectivity in $O(\log D + \log\log_{m/n} n)$-time using $O(m)$ processors and cells with good probability.

[2]In particular, our method applies to the most powerful Priority variant of CRCW PRAMs, where whenever multiple processors attempt to write to the same memory cell at a time step, the one with the smallest ID succeeds. It applies to the weaker variants of Arbitray or Common CRCW PRAMs as well.

1.2 Technical overview

At a high level, our Theorem 2 builds on and extends the classic derandomization framework for small-depth boolean circuits [17]. Ingredients that go into this framework include several influential results of circuit complexity, notably Håstad's average-case lower bounds [12] and the Nisan-Wigderson hardness versus randomness paradigm [18]. We begin by noting that CRCW PRAMs are computationally more powerful than circuits, and the best known simulation result only shows that every time-k p-processor CRCW PRAM can be simulated by a circuit of depth $O(k)$ and size $p^{2^{k+O(1)}}$ [15]. This is a considerable blow-up in size, and furthermore, the simulation is non-constructive. For both these reasons, we cannot apply the derandomization framework for small-depth boolean circuits as it is to CRCW PRAMs, but instead have to generalize each of the several components that go into the framework from the setting of boolean circuits to that of CRCW PRAMs. Due to the added power of CRCW PRAMs, this involves overcoming several technical challenges that we now discuss.

Central to our approach is the design of a *pseudorandom generator* for CRCW PRAMs, which is a function that stretches a small number of truly random bits, which we call the *random seeds*, into a near-exponential number of bits that look random, which we call the *pseudorandom bits*. Our overarching strategy for proving Theorem 2 is to replace the random bits used by a given CRCW PRAM with pseudorandom bits produced by a pseudorandom generator while preserving the success probability. The pseudorandom generator we use is the classic *Nisan-Wigderson generator* [18] which is designed to fool small-depth circuits of polynomial size. We show that the Nisan-Wigderson generator can in fact also fool the more powerful class of small-time CRCW PRAMs with polynomially many processors:

Theorem 3 (Nisan-Wigderson generator fools CRCW PRAMs, informal statement)**.** *Let $G : \{0,1\}^d \to \{0,1\}^m$ be the Nisan-Wigderson generator with $d = (\log m)^{O(k)}$. Then any deterministic CRCW PRAM with k time steps and $p = m^{O(1)}$ processors cannot distinguish between the distributions U_m and $G(U_d)$ well. Here, U_d and U_m denote the uniform distribution on $\{0,1\}^d$ and $\{0,1\}^m$ respectively.*

The seed length d that we achieve here is optimal given the current state of the art in complexity theory, in the sense that an improvement to $(\log m)^{o(k)} = (\log p)^{o(k)}$ will imply $\mathsf{P} \neq \mathsf{NC}^1$. Indeed, this is true even for the weaker model of Boolean circuits, as had been noted by Nisan [17]. Briefly, an efficient pseudorandom generator for size-p depth-k circuits with seed length $(\log p)^{o(k)}$ implies a lower bound against the circuits of depth $k = \omega(\log p/\log\log p)$, and it is known that every function in NC^1 can be computed by a circuit of depth $k = O(\log p/\log\log p)$ (see e.g. [14, 20]).

A challenge arising in the PRAM case is that, in a randomized CRCW PRAM on n input bits, a processor can generate as many as $r = (\log n)^{O(1)}$ random bits in a single time step, which requires us to feed pseudorandom bits to processors in groups of size at most r. In comparison, random bits in a circuit are wired to the gates in

the same way as the input bits. To address this challenge, we prove a stronger version of Theorem 3 that, even if a processor is allowed to read as many as r input bits in a single time step, the PRAM still cannot distinguish between the pseudorandom bits and the truly random bits.

Our proof of correctness of the Nisan-Wigderson generator crucially relies on the existence of a function that is *average-case hard* against CRCW PRAMs, i.e. showing that no CRCW PRAM algorithm can compute this function correctly on noticeably more than half of the inputs. In the simpler setting of small-depth circuits, the analogous lower bound is given by a seminal result of Håstad [12], which showed that a polynomial-size circuit of depth $o(\log \ell / \log\log \ell)$ cannot compute the parity function on ℓ input bits correctly on more than a fraction of $1/2 + o(1)$ of the inputs. This lower bound underlies the proof of correctness of the Nisan-Wigderson generator in fooling small-depth circuits. For the more challenging setting of CRCW PRAMs, Beame and Håstad [3] proved a *worst-case* lower bound, showing that a CRCW PRAM with polynomially many processors that runs in time $o(\log \ell / \log\log \ell)$ cannot compute the ℓ-bit parity function on *all* inputs. Our next result establishes an average-case lower bound against CRCW PRAMs, thereby simultaneously strengthening both [12] and [3]:

Theorem 4 (Average-case hardness of parity in CRCW PRAMs)**.** *Any CRCW PRAM with $o(\log \ell / \log\log \ell)$ time steps and $\ell^{O(1)}$ processors cannot compute the parity function on ℓ input bits correctly on more than a fraction of $1/2 + o(1)$ of the inputs.*

In addition to being the key tool in our proof that the Nisan-Wigderson generator fools CRCW PRAMs (Theorem 3), we believe that Theorem 4 is of independent interest as it sheds new light on the limitations of CRCW PRAMs.

Our proof Theorem 4 extends Håstad's proof framework for the case of circuits to the more general setting of CRCW PRAMs. The specifics of the CRCW PRAM model necessitates significant changes to Håstad's proof. One main difference is that, since the dependence of the state of a processor in a PRAM on the input is more complicated than that of a gate in a circuit, we use the notion of the *degree of the partition* associated to a processor introduced by [3] as a quantitative measure of such dependence. Moreover, Håstad's proof uses an induction that collapses two consecutive levels of a circuit into one at each step, an operation that does not have a natural analogy in PRAMs. As a solution, we propose a new inductive framework that does not modify the given PRAM but zooms in to only a subset of the time steps at each inductive step.

ACKNOWLEDGMENTS

Moses Charikar was supported by a Simons Investigator Award. Weiyun Ma was supported by a Stanford Graduate Fellowship. Li-Yang Tan was supported by NSF CAREER Award CCF-1942123.

REFERENCES

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

[2] Philipp Bamberger, Fabian Kuhn, and Yannic Maus. 2020. Efficient deterministic distributed coloring with small bandwidth. In *Proceedings of the 39th Symposium on Principles of Distributed Computing*. 243–252.

[3] Paul Beame and Johan Hastad. 1989. Optimal bounds for decision problems on the CRCW PRAM. *Journal of the ACM (JACM)* 36, 3 (1989), 643–670.

[4] Paul Beame, Paraschos Koutris, and Dan Suciu. 2017. Communication steps for parallel query processing. *J. ACM* 64, 6 (2017), 40:1–40:58.

[5] Soheil Behnezhad, Laxman Dhulipala, Hossein Esfandiari, Jakub Lacki, and Vahab Mirrokni. 2019. Near-optimal massively parallel graph connectivity. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 1615–1636.

[6] Artur Czumaj, Peter Davies, and Merav Parter. 2020. Graph sparsification for derandomizing massively parallel computation with low space. In *Proceedings of the 32nd ACM Symposium on Parallelism in Algorithms and Architectures*. 175–185.

[7] Artur Czumaj, Peter Davies, and Merav Parter. 2020. Simple, deterministic, constant-round coloring in the congested clique. In *Proceedings of the 39th Symposium on Principles of Distributed Computing*. 309–318.

[8] Mohsen Ghaffari, David G Harris, and Fabian Kuhn. 2018. On derandomizing local distributed algorithms. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 662–673.

[9] Michael T. Goodrich, Nodari Sitchinava, and Qin Zhang. 2011. Sorting, Searching, and Simulation in the Mapreduce Framework. In *Proceedings of the 22nd International Conference on Algorithms and Computation (ISAAC)* (Yokohama, Japan). 374–383. https://doi.org/10.1007/978-3-642-25591-5_39

[10] David G Harris. 2019. Derandomized concentration bounds for polynomials, and hypergraph maximal independent set. *ACM Transactions on Algorithms (TALG)* 15, 3 (2019), 1–29.

[11] David G Harris. 2019. Distributed local approximation algorithms for maximum matching in graphs and hypergraphs. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 700–724.

[12] Johan Håstad. 1987. *Computational Limitations of Small-Depth Circuits*. MIT Press, Cambridge, MA, USA.

[13] Howard Karloff, Siddharth Suri, and Sergei Vassilvitskii. 2010. A model of computation for MapReduce. In *Proceedings of the Twenty-first Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 938–948.

[14] Maria Klawe, Wolfgang J Paul, Nicholas Pippenger, and Mihalis Yannakakis. 1984. On monotone formulae with restricted depth. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*. 480–487.

[15] M LI and Y YESHA. 1989. New lower bounds for parallel computation. *Journal of the Association for Computing Machinery* 36, 3 (1989), 671–680.

[16] Sixue Cliff Liu, Robert E Tarjan, and Peilin Zhong. 2020. Connected components on a pram in log diameter time. In *Proceedings of the 32nd ACM Symposium on Parallelism in Algorithms and Architectures*. 359–369.

[17] Noam Nisan. 1991. Pseudorandom bits for constant depth circuits. *Combinatorica* 11, 1 (1991), 63–70.

[18] Noam Nisan and Avi Wigderson. 1994. Hardness vs randomness. *Journal of computer and System Sciences* 49, 2 (1994), 149–167.

[19] Václav Rozhoň and Mohsen Ghaffari. 2020. Polylogarithmic-time deterministic network decomposition and distributed derandomization. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*. 350–363.

[20] Luca Trevisan and Tongke Xue. 2013. A derandomized switching lemma and an improved derandomization of AC0. In *2013 IEEE Conference on Computational Complexity*. IEEE, 242–247.