Equivalence Classes and Conditional Hardness in Massively Parallel Computations

Danupon Nanongkai
KTH Royal Institute of Technology, Sweden
danupon@gmail.com

Michele Scquizzato
University of Padova, Italy
scquizza@math.unipd.it

Abstract

The *Massively Parallel Computation* (MPC) model serves as a common abstraction of many modern large-scale data processing frameworks, and has been receiving increasingly more attention over the past few years, especially in the context of classical graph problems. So far, the only way to argue lower bounds for this model is to condition on conjectures about the hardness of some specific problems, such as graph connectivity on promise graphs that are either one cycle or two cycles, usually called the *one cycle vs. two cycles* problem. This is unlike the traditional arguments based on conjectures about complexity classes (e.g., $\mathsf{P} \neq \mathsf{NP}$), which are often more robust in the sense that refuting them would lead to groundbreaking algorithms for a whole bunch of problems.

In this paper we present connections between problems and classes of problems that allow the latter type of arguments. These connections concern the class of problems solvable in a sublogarithmic amount of rounds in the MPC model, denoted by $\mathsf{MPC}(o(\log N))$, and some standard classes concerning space complexity, namely L and NL, and suggest conjectures that are robust in the sense that refuting them would lead to many surprisingly fast new algorithms in the MPC model. We also obtain new conditional lower bounds, and prove new reductions and equivalences between problems in the MPC model.

2012 ACM Subject Classification Theory of computation → Massively parallel algorithms

Keywords and phrases Massively parallel computation, conditional hardness, fine-grained complexity

Digital Object Identifier 10.4230/LIPIcs.OPODIS.2019.33

Funding This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No 715672. M. Scquizzato was also partially supported by the University of Padova under grant BIRD197859/19.

1 Introduction

The *Massively Parallel Computation* (MPC) model is arguably the most popular model of computation that captures the essence of several very successful general-purpose frameworks for massively parallel coarse-grained computations on large data sets, such as MapReduce [23], Hadoop [60], Spark [62], and Dryad [38]. The MPC model, introduced by Karloff et al. [40], and originally inspired by the MapReduce paradigm, aims at modeling distributed-memory parallel computations in the situation when the size of the input is so big that a single machine cannot even store the whole input, but just a strongly sublinear fraction of it. The computation proceeds in synchronous rounds, and in each of them the machines can exchange data with each other with the sole restriction that no one can ever receive more data than it is capable of storing. The goal is to keep the total number of rounds as low as possible.

This basic model has been much investigated in the past decade, mostly from an algorithmic point of view [40, 45, 33, 12, 52, 53, 2, 43, 35, 4, 39, 26, 13, 37, 55, 15, 22, 61, 29, 5, 11, 9, 8, 31, 34, 28, 36, 6, 14, 20, 17, 30, 16]. A common outcome is that, when N denotes the input size, a solution terminating in $O(\log N)$ rounds is possible, usually by simulating known PRAM algorithms [40, 33], but going below that resisted the efforts of many researchers. Recently, a few works managed to break the $O(\log N)$ barrier by relaxing a bit the strongly-sublinear constraint on the memory size, and showed that some graph problems allow for $o(\log N)$-round solutions in the so-called *near-linear memory* regime, whereby machines have memories of size $\tilde{O}(n)$, where n is the number of nodes in the graph [22, 29, 8, 9, 17]. However, without this kind of relaxations only a handful of problems are known to admit a $o(\log N)$-round algorithm [31, 34, 20]. A fundamental question is thus whether many known $O(\log N)$-round algorithms can be complemented with tight lower bounds.

Unfortunately, proving *unconditional* lower bounds – that is, without any assumptions – seems extremely difficult in this model, as it would imply a breakthrough in circuit complexity: Roughgarden et al. [55] showed that, when enough machines are available, proving any super-constant lower bound for any problem in P would imply new circuit lower bounds, and specifically would separate NC^1 from P – a long-standing open question in complexity theory that is a whisker away from the P vs. NP question. This means that the lack of super-constant lower bounds in the MPC model can be blamed on our inability to prove some computational hardness results.

In light of this barrier, the focus rapidly shifted to proving *conditional* lower bounds, that is, lower bounds conditioned on plausible hardness assumptions. One widely-believed assumption concerns graph connectivity, which, when machines have a memory of size $O(n^{1-\epsilon})$ for some constant $\epsilon > 0$, is conjectured to require $\Omega(\log n)$ MPC rounds [40, 53, 13, 55, 61].[1] The same conjecture is often made even for the special case of the problem where the graph consists of either one cycle or two cycles, usually called *one cycle vs. two cycles* problem. The one cycle vs. two cycles conjecture has been proven useful to show conditional lower bounds for some problems, such as minimum spanning trees in low-dimensional spaces [4], single-linkage clustering [61], 2-vertex connectivity [6], generation of random walks [44], as well as parameterized conditional lower bounds [16].[2]

However, it is not clear whether the one cycle vs. two cycles conjecture is true or not, and if not, what its refutation implies. This situation is in contrast with traditional complexity theory, where a refutation of a conjectured relationship between complexity classes would typically imply groundbreaking algorithmic results for a large number of problems; for example, if the $\mathsf{P} \neq \mathsf{NP}$ conjecture fails, then there would be efficient (polynomial-time) algorithms for *all* problems in NP, including a number of "hard" problems. To put it another way, a conjecture like $\mathsf{P} \neq \mathsf{NP}$ is more *robust* in the sense that it is extremely hard to refute – doing so requires a major algorithmic breakthrough. The goal of this paper is to explore conjectures of this nature in the MPC model.

[1] Observe that in the near-linear memory regime this conjecture breaks: graph connectivity can be solved in $O(1)$ MPC rounds [15].

[2] The one cycle vs. two cycles problem is usually stated such that, in the case of two cycles, these have $n/2$ nodes each. However, we observe that all the mentioned conditional lower bounds hold also when the two cycles may have arbitrary lengths.

1.1 Summary of Contributions

In this paper we show many connections between problems and classes of problems that lead to more robust conjectures for the MPC model. In particular, we study the connections between the class of problems solvable in a sublogarithmic amount of rounds in the MPC model, denoted by $\mathsf{MPC}(o(\log N))$, and the standard space complexity classes L and NL. (Recall that L and NL are the classes of decision problems decidable in logarithmic space on deterministic and nondeterministic Turing machines, respectively.) The connection between MPC and these complexity classes is enabled by a recent result showing how Boolean circuits can be efficiently simulated in the MPC model. In short, we present a set of observations and reductions that suggest that $\mathsf{L} \not\subseteq \mathsf{MPC}(o(\log N))$ and $\mathsf{NL} \not\subseteq \mathsf{MPC}(o(\log N))$ are two robust conjectures that might play crucial roles in arguing lower bounds in the MPC model, as they already imply tight conditional lower bounds for a large number of problems. In particular, with some assumptions on the total amount of memory (equivalently, machines) available in the system, we can conclude the following.

1. **Robustness:** The one cycle vs. two cycles conjecture is robust, since it is equivalent to conjecturing that $\mathsf{L} \not\subseteq \mathsf{MPC}(o(\log N))$, and refuting this conjecture requires showing $o(\log N)$-round algorithms for all problems in L. This class includes many important problems such as graph connectivity, cycle detection, and planarity testing (see problems in the bottom ellipse in Figure 1 for more).
2. **Equivalences:** All L-complete problems are *equivalent* in the sense that they require asymptotically the same number of rounds. This means that the one cycle vs. two cycles problem, which is L-complete (see Appendix A), is equivalent to many seemingly harder problems, such as graph bipartiteness, minimum cut, and formula evaluation (see problems in the bottom ellipse in Figure 1 for more).

   Additionally, all NL-complete problems and a few others are also equivalent. These problems include *st*-reachability, all-pairs shortest paths (both the directed and undirected cases) on unweighted graphs, diameter, and betweenness centrality (see problems in the top ellipse in Figure 1 for more).
3. **New conditional lower bounds:** Assuming the one cycle vs. two cycles conjecture (equivalently, $\mathsf{L} \not\subseteq \mathsf{MPC}(o(\log N))$), there are no $o(\log N)$-round algorithms for all L-hard problems and a few other problems. This implies new conditional lower bounds for more than a dozen of problems, such as betweenness centrality, planarity testing, graph bipartiteness, list ranking, formula evaluation, and densest subgraph (see problems in the big rectangle in Figure 1 for more). Previously only a few lower bounds were known, e.g., those for single-linkage clustering [61] and maximum matching [48]. (Of course, lower bounds for connectivity-related problems are trivially implied by the one cycle vs. two cycles conjecture.) Most of our lower bounds are tight (e.g., lower bounds for problems in the ellipses in Figure 1).
4. **A more robust conjecture.** For NL-hard problems, we can argue lower bounds under the more robust $\mathsf{NL} \not\subseteq \mathsf{MPC}(o(\log N))$ conjecture. These problems include perfect matching, single-source shortest paths, diameter, and network flow (see problems in the small rectangle in Figure 1 for more). Note that the $\mathsf{NL} \not\subseteq \mathsf{MPC}(o(\log N))$ conjecture is more robust (i.e., more likely to be true) since $\mathsf{L} \subseteq \mathsf{NL}$.

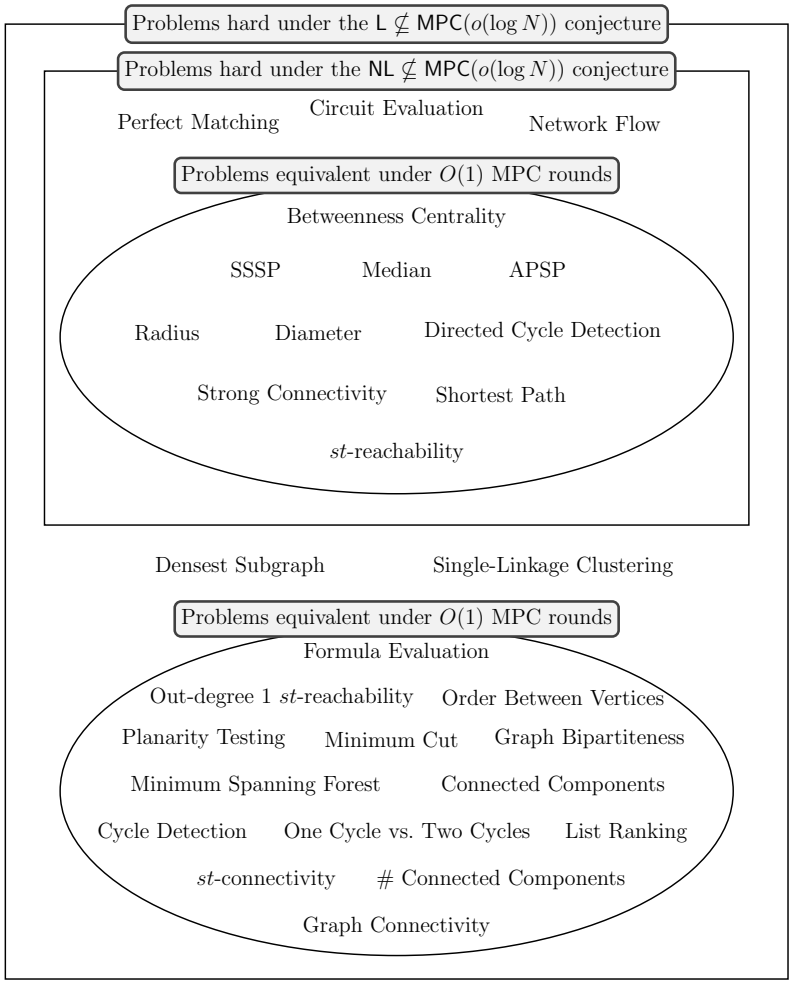

Figure 1 A classification of the complexity of some prominent problems in the MPC model. Problems in the top ellipse are on unweighted graphs.

1.2 Related Work

Fish et al. [26] were perhaps the first to establish a connection between the MPC model and classical complexity classes. Besides the introduction of a uniform version of the model, they showed that constant-round MPC computations can simulate sublogarithmic space-bounded Turing machines, and then proved strict hierarchy theorems for the MPC model under certain complexity-theoretic assumptions.

Roughgarden et al. [55] discuss connections between the MPC model and Boolean circuits. They show that standard degree arguments for circuits can be applied to MPC computations as well, and specifically that any Boolean function whose polynomial representation has degree d requires $\Omega(\log_s d)$ rounds of MPC using machines with memory s. This implies an $\Omega(\log_s n)$ lower bound on the number of rounds for graph connectivity. Perhaps more interestingly, the authors show a barrier for unconditional lower bounds by observing that, if enough machines are available, then proving any super-constant lower bound in the MPC model for any problem in P would imply new circuit lower bounds, and specifically would

separate NC^1 from P, thus answering a notorious open question in circuit complexity. This result follows by showing that, with a number of available machines polynomial in the number of input nodes of the circuit, NC^1 circuits can be efficiently simulated in the MPC model. We observe that their argument readily generalizes to show that any bounded fan-in Boolean circuit of depth d and of polynomial size can be simulated in $O(\lceil d/\log s\rceil)$ MPC rounds. Very recently, Frei and Wada [27] prove the same result improving over the amount of machines required for the simulation – from linear to strongly sublinear in the size of the circuit.

Given the difficulty of proving lower bounds for all algorithms, one can (a) prove lower bounds for restricted classes of algorithms, or (b) prove conditional lower bounds: assume one lower bound, and transfer the conjectured hardness to other problems via reductions (with common examples being the theory of NP-hardness and its more recent analogue for problems in P, usually called *fine-grained complexity theory*). Both paths give a deep understanding and warn us what not to try when designing algorithms.

Within the first line of inquiry, Pietracaprina et al. [52] prove lower bounds for matrix multiplication algorithms that compute all the n^3 elementary products. Similar kinds of limitations are required by Beame et al. [13], Jacob et al. [39], Im and Moseley [36], and Assadi and Khanna [10] to prove lower bounds for st-connectivity, list ranking, graph connectivity, and maximum coverage, respectively. Of a similar flavor are the results of Afrati et al. [2], who show, for a fixed number of rounds (usually a single round), space-communication tradeoffs.

Within the second line of inquiry fall [4, 61, 6, 44], which use the conjecture on the hardness of graph connectivity as a hardness assumption for proving conditional lower bounds for other problems such as minimum spanning trees in low-dimensional spaces, single-linkage clustering, 2-vertex connectivity, and generating random walks, respectively. Very recently, Ghaffari et al. [30] present conditional lower bounds for other key graph problems such as approximate maximum matching, approximate vertex cover, maximal independent set, and maximal matching. Their lower bounds also rest on the hardness of graph connectivity, and are obtained by introducing a new general method that lifts (unconditional) lower bounds from the classical LOCAL model of distributed computing to the MPC model. A conditional lower bound following a different kind of argument is given by Andoni et al. [5], who show that an $n^{o(1)}$-round MPC algorithm that answers $O(n+m)$ pairs of reachability queries in directed graphs with n nodes and m edges can be simulated in the RAM model yielding faster Boolean matrix multiplication algorithms.

Several other models have been developed in the quest to establish rigorous theoretical foundations of (massively) parallel computing, with the PRAM being one of the most investigated. The MPC model is more powerful than the PRAM since PRAM algorithms can be simulated in the MPC model with constant slowdown [40, 33], and some problems (such as evaluating the XOR function) can be solved much faster in the MPC model.

Valiant's *bulk-synchronous parallel* (BSP) model [58] anticipated many of the features of MPC-type computations, such as the organization of the computation in a sequence of synchronous rounds (originally called *supersteps*). Several papers (e.g., [32, 47, 1, 56, 18]) explored the power of this model by establishing lower bounds on the number of supersteps or on the communication complexity required by BSP computations. Lower bounds on the number of supersteps are usually of the form $\Omega(\log_h N)$, where h is the maximum number of messages sent or received by any processor in any superstep.

Another model aiming at serving as an abstraction for modern large-scale data processing frameworks is the *k-machine* model [41]. Partly inspired by message-passing models in distributed computing, in the k-machine model there are k available machines, and in each

round any pair of machines is allowed to communicate using messages of a given size. Hard bounds on the point-to-point communication lead to very strong round lower bounds in this model [41, 49, 50].

The *congested clique* (see, e.g., [24]) is a model for network computations bearing some similarities with the MPC model. On one hand, algorithms for this model can be simulated in the MPC model – under some specific conditions on the size of the local memories [35, 29, 15]. On the other hand, analogously to the MPC model, proving a super-constant unconditional lower bound in the congested clique for a problem in NP would imply better circuit size-depth tradeoffs for such a problem than are currently known [24]. This induced further investigations of the model under the lens of complexity theory [42].

2 Preliminaries

In this section we present the MPC model in detail. We assume that the reader is familiar with the standard space complexity classes L and NL, and with the logspace-uniform circuit complexity classes NC^k and AC^k (see, e.g., the textbook [7]).

2.1 The MPC Model

The *Massively Parallel Computation (MPC)* model is a theoretical abstraction capturing the main distinguishing aspects of several popular frameworks for the parallel processing of large-scale datasets. It was introduced by Karloff, Suri, and Vassilvitskii [40], and refined in subsequent work [33, 13, 4].

In this model the system consists of p identical machines (processors), each with a local memory of size s. If N denotes the size of the input, then $s = O(N^{1-\epsilon})$ for some fixed constant $\epsilon > 0$, and the total amount of memory available in the system is $p \cdot s = O(N^{1+\gamma})$ for some fixed constant $\gamma \geq 0$. The space size is measured by words, each of $\Theta(\log N)$ bits. Initially, the input is adversarially distributed across the machines. The computation proceeds in synchronous rounds. In each round, each machine performs some computation on the data that resides in its local memory, and then, at the end of the round, exchanges messages with other machines. The total size of messages sent or received by each machine in each round is bounded by s. The goal is to minimize the total number of rounds.

For problems defined on graphs, the input size N is equal to $n+m$, where n is the number of nodes of the graph and m is the number of edges. When considering graph problems, in this paper we assume $s = O(n^{1-\epsilon})$. This regime of memory size, usually called *strongly sublinear memory* regime, is always in compliance with the aforementioned constraint on the size of the local memory, even when graphs are sparse, for which the constraint is the most restrictive.

Since we want to relate the MPC model to classical complexity classes, one must make sure that the model is *uniform*, by which we mean, roughly speaking, that the same algorithm solves the problem for inputs of all (infinitely many) sizes. Fish et al. [26] dealt with this issue observing that Karloff et al.'s original definition of the model [40] is non-uniform, allowing it to decide undecidable languages, and thus by reformulating the definition of the model to make it uniform. Building on that reformulation, and letting $f\colon \mathbb{N} \to \mathbb{R}^+$ be a function, we define the class $\mathsf{MPC}(f(N))$ to be the class of problems solvable in $O(f(N))$ MPC rounds by a uniform family of MPC computations.

3 Massively Parallel Computations and Space Complexity Classes

In this section we recall a recent result showing that Boolean circuits can be efficiently simulated in the MPC model, and then we build on it to derive new results and conjectures.

3.1 Efficient Circuit Simulation in the MPC Model

We now recall the main result in [27] which, roughly speaking, says that any bounded fan-in Boolean circuit of depth d and of polynomial size can be simulated in $O(\lceil d/\log s \rceil)$ MPC rounds. This result is already implicit in [55], where it is achieved by a simple simulation whereby each gate of the circuit is associated with a machine whose responsibility is to compute the output of the gate. This requires the availability of a number of machines polynomial in the number of inputs n of the circuit, and thus linear in the size of the circuit. Very recently, Frei and Wada [27] came up with a more sophisticated strategy, which uses only a strongly sublinear amount of machines. Their strategy employs two distinct simulations: for NC^1 circuits they exploit Barrington's well-known characterization of NC^1 in terms of bounded-width polynomial-size branching programs, and thus simulate such branching programs in a constant number of rounds; for the higher levels of the NC hierarchy, the Boolean circuits themselves are directly simulated, suitably dividing the computation into the simulation of sub-circuits of depth $O(\log n)$, each to be accomplished in $O(1)$ rounds.

The authors work in the original model of Karloff et al. [40], but their result seamlessly applies in the refined MPC model.

▶ **Theorem 1** ([27]). *Let DMPC^i denote the class of problems solvable by a deterministic MPC algorithm in $O(\log^i N)$ rounds with $O(N^{1-\epsilon})$ local memory per machine and $O(N^{2(1-\epsilon)})$ total memory. Then,*

$$\mathsf{NC}^{i+1} \subseteq \mathsf{DMPC}^i$$

for every $i \in \mathcal{N}$ and for every $\epsilon \in (0, 1/2)$. (When $i = 0$, the result holds also for $\epsilon = 1/2$.)

Setting $i = 0$, we have the following.

▶ **Corollary 2.** *The class NC^1 can be simulated in $O(1)$ MPC rounds with $O(N^{1-\epsilon})$ local memory per machine and $O(N^{2(1-\epsilon)})$ total memory, for any constant $\epsilon \in (0, 1/2]$.*

Since $\mathsf{NC}^1 \subseteq \mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{NC}^2$ (see, e.g., [51]), an immediate by-product of Theorem 1 is that some standard space complexity classes can be efficiently simulated in the MPC model.

▶ **Corollary 3.** *The class NC^2, and thus the classes L and NL, can be simulated in $O(\log N)$ MPC rounds with $O(N^{1-\epsilon})$ local memory per machine and $O(N^{2(1-\epsilon)})$ total memory, for any constant $\epsilon \in (0, 1/2)$.*

3.2 New Consequences of Circuit Simulations

In this section we discuss new consequences of the fact that the MPC model is powerful enough to efficiently simulate general classes of Boolean circuits.

▶ **Theorem 4.** *Consider the MPC model where the size of the local memory per machine is $O(N^{1-\epsilon})$ for any constant $\epsilon \in (0, 1/2]$, and assume that $\Omega(N^{2(1-\epsilon)})$ total memory is available. Let $f\colon \mathbb{N} \to \mathbb{R}^+$ be a function. Then, if any L-hard problem can be solved in $O(f(N))$ MPC rounds, so can all the problems in the class L. Moreover, either all L-complete problems can be solved in $O(f(N))$ MPC rounds, or none of them can.*

Proof. Both claims follow directly from the definitions of L-hardness and L-completeness, and from Corollary 2. Let A be an L-hard problem that can be solved in $O(f(N))$ MPC rounds. By definition of L-hardness, every problem in L is NC^1 reducible to A. By assumption, $\epsilon \in (0, 1/2]$ and $\Omega(N^{2(1-\epsilon)})$ total memory is available, and thus, by Corollary 2, an NC^1 reduction can be simulated in $O(1)$ MPC rounds, giving the first claim. Therefore, in particular, if any L-complete problem can be solved in $O(f(N))$ MPC rounds, so can all the other L-complete problems. In other words, either all L-complete problems can be solved in $O(f(N))$ MPC rounds, or none of them can. ◀

We remark that in Theorem 4 no assumption is placed on the function $f(N)$, which therefore can be of any form, even a constant. Hence, Theorem 4 says that all the known L-complete problems such as graph connectivity, graph bipartiteness, cycle detection, and formula evaluation, are *equivalent* in the MPC model, and in a very strong sense: they all require asymptotically the same number of rounds. (Analogous equivalences are common in computer science, e.g., in fine-grained complexity theory, where equivalence classes of problems within P, such as the APSP class [59], are established.) Thus, this simple result provides an explanation of the striking phenomenon that for these well-studied problems we seem unable to break the $O(\log N)$ barrier in the MPC model. It also implies that the conjectures on the hardness of graph connectivity and on the hardness of the one cycle vs. two cycles problem are equivalent, at least when $\Omega(N^{2(1-\epsilon)})$ total memory is available.

The next theorem provides an even stronger barrier for improvements in the MPC model.

▶ **Theorem 5.** *Consider the MPC model where the size of the local memory per machine is $O(N^{1-\epsilon})$ for any constant $\epsilon \in (0, 1/2]$, and assume that $\Omega(N^{2(1-\epsilon)})$ total memory is available. Let $f : \mathbb{N} \to \mathbb{R}^+$ be a function. If any L-hard problem can be solved in $O(f(N))$ MPC rounds, then either all NL-complete problems can be solved in $O(f(N))$ MPC rounds, or none of them can. Moreover, if any NL-hard and any L-hard problem can be solved in $O(f(N))$ MPC rounds, so can all the problems in the class NL.*

Proof. Let A be an L-hard problem that can be solved in $O(f(N))$ MPC rounds. Then, by Theorem 4, every problem in the class L can be solved in $O(f(N))$ MPC rounds and thus, in particular, every log-space reduction can be computed in $O(f(N))$ MPC rounds. By definition of NL-completeness, every problem in NL, and thus, in particular, any NL-complete problem, is log-space reducible to any other NL-complete problem, and this proves the first statement.

Let B be an NL-hard problem that can be solved in $O(f(N))$ MPC rounds. By definition of NL-hardness, every problem in NL is log-space reducible to B. Since we have just argued that if any L-hard problem can be solved in $O(f(N))$ MPC rounds, so can any log-space reduction, the second statement follows. ◀

Once again, we stress that in Theorem 5 no assumption is placed on the function $f(N)$, which therefore can be of any form, even a constant.

Theorem 5 indicates that, unless L = NL, in the MPC model the connectivity problem on directed graphs, which is both NL-complete and L-hard, is strictly harder than on undirected graphs in the sense that breaking the current logarithmic barrier, if possible, would be strictly harder.

3.2.1 New Conjectures

The common belief that problems such as graph connectivity and list ranking cannot be solved in $o(\log N)$ MPC rounds, along with the equivalence result of Theorem 4, justify the following conjecture.

▶ **Conjecture 1.** *No L-hard problem can be solved in $o(\log N)$ MPC rounds with $O(N^{1-\epsilon})$ local memory per machine, for any constant $\epsilon \in (0,1)$, not even with a polynomial amount of total memory. Equivalently,*

$$\mathsf{L} \not\subseteq \mathit{MPC}(o(\log N)).$$

We now show the claimed equivalence.

▶ **Proposition 6.** *The two statements in Conjecture 1 are equivalent.*

Proof. We shall argue that if any of the two statements is wrong, so is the other, and vice versa. Assume $\mathsf{L} \subseteq \mathsf{MPC}(o(\log N))$. Then, some L-complete, and hence L-hard, problem is contained in $\mathsf{MPC}(o(\log N))$, that is, it can be solved in $o(\log N)$ MPC rounds. To show the other direction, assume that there exists an L-hard problem that can be solved in $o(\log N)$ MPC rounds with a polynomial amount of total memory. Then, by Theorem 4, every problem in L can be solved in $o(\log N)$ MPC rounds, i.e., $\mathsf{L} \subseteq \mathsf{MPC}(o(\log N))$. ◀

We would like to remark that, in light of Theorem 4, Conjecture 1 is totally equivalent to the preceding conjectures on the hardness of graph connectivity or of the one cycle vs. two cycles problem [40, 53, 13, 55, 61]; however, Theorem 4 significantly strengthens the evidence for such conjectures.

Likewise, Theorem 5 provides a justification for the following conjecture.

▶ **Conjecture 2.** *No NL-hard and L-hard problem can be solved in $o(\log N)$ MPC rounds with $O(N^{1-\epsilon})$ local memory per machine, for any constant $\epsilon \in (0,1)$, not even with a polynomial amount of total memory. Equivalently,*

$$\mathsf{NL} \not\subseteq \mathit{MPC}(o(\log N)).$$

The two statements in Conjecture 2 are equivalent. The argument is similar to that used for the preceding proposition.

Observe that since $\mathsf{L} \subseteq \mathsf{NL}$, Conjecture 1 implies Conjecture 2. Hence, unless $\mathsf{L} = \mathsf{NL}$, Conjecture 2 is weaker than Conjecture 1, and thus more likely to be true.

4 Reductions and Equivalences in Massively Parallel Computations

In this section we discuss two equivalence classes of problems and some conditional lower bounds in the MPC model. The two equivalence classes both contain problems equivalent to each other under $O(1)$-round MPC reductions and for which the best known upper bound is $O(\log N)$ rounds, but differ in terms of the low-space computational complexity characterization of the problems they contain.

As a consequence of the results of Section 3, most of these reductions and equivalences follow from known hardness and completeness results for low-space complexity classes such as L and NL.

We will also show novel reductions and equivalences in the MPC model. Some of such reductions crucially require the availability of up to polynomially many machines (equivalently, a total amount of memory up to polynomial in the input size), which are used to host up to a polynomial number of copies of the input data. The quick creation of so many input replicas can be achieved through the use of a simple two-step broadcast procedure, as shown in the following lemma.

▶ **Lemma 7.** *The input data can be replicated up to a polynomial number of times in $O(1)$ MPC rounds.*

4.1 An Equivalence Class for Undirected Graph Connectivity

In this section we discuss the MPC equivalence class for graph connectivity in undirected graphs. This problem, which asks to determine whether a given undirected graph is connected or not, was one of the first problems to be shown L-hard under (uniform) NC^1 reductions [21], and then it was placed in L by the remarkable algorithm of Reingold [54]. Exploiting the results of Section 3, we know that one can recycle all the reductions that have been developed in classical complexity theory for showing hardness and completeness for class L in the MPC model as well, since these can all be simulated in $O(1)$ MPC rounds with $O(N^{2(1-\epsilon)})$ total memory. This immediately implies that the class of L-complete problems forms an equivalence class in the MPC model as well. Specifically, for example, either all the following problems can be solved with a sublogarithmic MPC algorithm, or none of them can:graph connectivity, connectivity for promise graphs that are a disjoint union of cycles, *st*-connectivity, *st*-reachability for directed graphs of out-degree one, cycle detection, order between vertices, formula evaluation, and planarity testing.

Recycling (some) old SL-completeness results. Many more problems can be placed in this MPC equivalence class almost effortlessly: this is the case for some problems complete for the class *symmetric logarithmic space* (SL), a class defined by Lewis and Papadimitriou [46] to capture the complexity of undirected *st*-connectivity before this was eventually settled by the breakthrough of Reingold. Completeness in SL is defined in terms of log-space reductions, and *st*-connectivity is one complete problem for it. Since $\mathsf{L} \subseteq \mathsf{SL}$, Reingold's algorithm made these two classes collapse, thus widening the class L with many new problems. However, completeness for SL does not translate into completeness for L, since the latter is defined in terms of a lower-level kind of reduction. Luckily, some of the log-space reductions devised to show hardness for SL turn out to be actually stronger than log-space. This is the case, e.g., of testing whether a given graph is bipartite (or, equivalently, 2-colorable), as we show next.

▶ **Lemma 8.** *Graph bipartiteness is equivalent to st-connectivity under $O(1)$-round MPC reductions, with $O(n^{1-\epsilon})$ local memory per machine for any constant $\epsilon \in (0,1)$, and $O(n(n+m))$ total memory.*

A good source of problems complete for SL is [3].

From decision to non-decision problems. Complexity classes such as L contain problems phrased as decision problems. Nevertheless, it is often easy to transform them into their non-decision version. As an example, consider *order between vertices* (ORD). ORD is the decision version of list ranking, the problem of obtaining a total ordering from a given successor relation [25]. It is easy to argue the following equivalence.

▶ **Lemma 9.** *List ranking is equivalent to order between vertices under $O(1)$-round MPC reductions, with $O(n^{1-\epsilon})$ local memory per machine for any constant $\epsilon \in (0,1)$, and $O(n^3)$ total memory.*

Non-pairwise reductions. Sometimes back-and-forth reductions between two problems are not known. In this case their equivalence may nevertheless be established through a series of reductions involving related problems.

▶ **Lemma 10.** *Graph connectivity, st-connectivity, # connected components, connected components, minimum spanning forest, and minimum cut are all equivalent under $O(1)$-round MPC reductions, with $O(n^{1-\epsilon})$ local memory per machine for any constant $\epsilon \in (0,1)$, and $\tilde{O}(n^2 m(n+m))$ total memory.*

We can now summarize all the results of this section.

▶ **Theorem 11.** *The following problems are all equivalent under $O(1)$-round MPC reductions, with $O(n^{1-\epsilon})$ local memory per machine for any constant $\epsilon \in (0,1)$, and $\tilde{O}(n^2 m(n+m))$ total memory: graph connectivity, connectivity for promise graphs that are a disjoint union of cycles, st-connectivity, st-reachability for directed graphs of out-degree one, cycle detection, order between vertices, formula evaluation, planarity testing, graph bipartiteness, list ranking, # connected components, connected components, minimum spanning forest, and minimum cut.*

Conditional hardness: L-hard problems. Finally, there are problems known to be L-hard, but not known to be in L, such as densest subgraph and perfect matching. Since for these problems only one-way reductions from problems in L are known, we don't know whether they are part of the equivalence class of undirected graph connectivity.

4.2 An Equivalence Class for Directed Graph Connectivity

In this section we discuss the MPC equivalence class for graph connectivity in directed graphs. The problem corresponding to *st*-connectivity in directed graphs is *st-reachability*, that is, the problem of detecting whether there is a path from a distinguished node s to a distinguished node t in a directed graph. *st*-reachability is the prototypical complete problem for NL [51, 57, 7].

Recall that hardness in NL is defined with respect to log-space reducibility, but we do not know whether log-space computations can be simulated in $o(\log N)$ MPC rounds – in fact, in Section 3 we conjecture they cannot. However, it turns out that many of the known log-space reductions that establish NL-hardness of problems can actually be simulated in $O(1)$ MPC rounds. This is the case, for example, of the reductions between *st*-reachability and *shortest path*, the other canonical example NL-complete problem which, given an undirected (unweighted) graph, two distinguished nodes s and t, and an integer k, asks to determine if the length of a shortest path from s to t is k.

▶ **Lemma 12.** *Shortest path on unweighted graphs is equivalent to st-reachability under $O(1)$-round MPC reductions, with $O(n^{1-\epsilon})$ local memory per machine for any constant $\epsilon \in (0,1)$, and $O(n^{\delta}(n+m))$ total memory where δ is a small enough positive constant.*

There are other NL-complete problems that can be shown to be equivalent under $O(1)$-round MPC reductions. Some examples are directed cycle detection, by a simple adaptation of the preceding reductions, and strong connectivity, which follows from a result in [19]. We suspect that many other log-space reductions are actually (or can easily be translated into) $O(1)$-round MPC reductions, thus enabling us to enlarge the equivalence class for graph connectivity in directed graphs almost effortlessly by leveraging known results in complexity theory.

When this is not possible, one might have to devise novel reductions. We now do so for some important shortest-path-related problems as well as for some graph centrality problems.

4.2.1 New Fine-Grained MPC Reductions: Constant-Round Equivalences Between Graph Centrality Problems, APSP, and Diameter

In this section we shall exploit the *shortest path* problem as a prototypical problem for which the fastest known MPC algorithm takes $O(\log n)$ rounds, and prove a collection of constant-round equivalences with many other graph problems.

We start by showing the simple fine-grained equivalence between APSP and shortest path.

▶ **Lemma 13.** *APSP is equivalent to shortest path under $O(1)$-round MPC reductions, with $O(n^{1-\epsilon})$ local memory per machine for any constant $\epsilon \in (0,1)$, and $O(n^2(n+m))$ total memory.*

Proof. The reduction from shortest path to APSP is obvious. The other direction is also immediate when we have enough machines, and specifically $O(n^2(n+m))$ total memory: by Lemma 7 we can create $2\binom{n}{2}$ copies of the input graph in $O(1)$ MPC rounds, and then in parallel, one pair for each copy, compute the shortest path for each (ordered, if the graph is directed) pair of nodes. ◀

In the following results we will use roughly the same reduction. We start with the problem of determining the diameter of a graph.

▶ **Lemma 14.** *Shortest path is $O(1)$-round MPC reducible to diameter, with $O(n^{1-\epsilon})$ local memory per machine for any constant $\epsilon \in (0,1)$, and $O(n+m)$ total memory.*

Proof. We start with the case of undirected graphs. Given an instance of shortest path, the idea is to alter the input graph by sticking two new and sufficiently long paths to nodes s and t, so that the path of largest total weight includes both s and t.

This is sufficient if the original graph G is connected; otherwise, the diameter is infinite, and from this information we cannot determine the length of a shortest path from s to t. Hence, we shall first make G connected in a way that alters the distance between s and t only if they are not connected in G. Since the distance between any two nodes can be at most $(n-1)M$, this can be achieved by adding to the graph a new node v and n edges of weight nM between v and any other node. Then, we append two additional chains to s and t, each with $2n$ edges of weight M, and denote this modified graph by G'.

This reduction can be performed in $O(1)$ MPC rounds, it increases the number of nodes and the number of edges by $O(n)$, and the maximum absolute weight by a factor of $O(n)$. Therefore, any MPC algorithms that runs in $O(T(n,m,M))$ rounds in the new graph G' can be used to solve the original instance G in $O(T(n,m,M))$ rounds as well.

Observe that the diameter of the modified graph G' must include the two chains appended to s and t. Hence any algorithm for the diameter when executed on graph G' always returns $4nM$ plus the shortest-path distance between s and t in G'. By construction, the latter quantity, which we denote by α, is at most $(n-1)M$ if s and t are connected in G, and (exactly) $2nM$ otherwise. Thus the answer to shortest path is α if the diameter of G' is at most $4nM + (n-1)M$, and infinity otherwise.

In the directed case, we use the same weighted graph G' as before, adding one parallel edge for each edge, both with the same weight but with opposite directions. The rest of the algorithm is the same and its analysis is analogous to the undirected case. ◀

▶ **Lemma 15.** *Shortest path is $O(1)$-round MPC reducible to radius, with $O(n^{1-\epsilon})$ local memory per machine for any constant $\epsilon \in (0,1)$, and $O(n+m)$ total memory.*

▶ **Lemma 16.** *Shortest path is $O(1)$-round MPC reducible to median, with $O(n^{1-\epsilon})$ local memory per machine for any constant $\epsilon \in (0,1)$, and $O(n+m)$ total memory.*

Now we consider the evaluation of the betweenness centrality of nodes. In contrast to the previous reductions, in the following one we shall create n copies of the reduction graph leveraging Lemma 7, and then perform some computation in parallel.

▶ **Lemma 17.** *Shortest path is $O(1)$-round MPC reducible to betweenness centrality, with $O(n^{1-\epsilon})$ local memory per machine for any constant $\epsilon \in (0,1)$, and $O(n(n+m))$ total memory.*

An immediate consequence of these results is the following.

▶ **Proposition 18.** *Shortest path, SSSP, APSP, diameter, radius, median, and betweenness centrality are all equivalent under $O(1)$-round MPC reductions, with $O(n^{1-\epsilon})$ local memory per machine for any constant $\epsilon \in (0,1)$, and $O(n^2(n+m))$ total memory.*

Proof. The two reductions involving SSSP are obvious. The reduction from diameter (or radius) to APSP is also obvious, since determining the maximum (or minimum) in a set of values can be easily done in $O(1)$ MPC rounds. The theorem then follows from Lemmas 13 to 17. ◀

We can now summarize all the results of this section.

▶ **Theorem 19.** *The following problems are all equivalent under $O(1)$-round MPC reductions, with $O(n^{1-\epsilon})$ local memory per machine for any constant $\epsilon \in (0,1)$, and $O(n^2(n+m))$ total memory: st-reachability, strong connectivity, directed cycle detection, unweighted shortest path, unweighted SSSP, unweighted APSP, unweighted diameter, unweighted radius, unweighted median, and unweighted betweenness centrality.*

References

1. Micah Adler, Wolfgang Dittrich, Ben H. H. Juurlink, Miroslaw Kutylowski, and Ingo Rieping. Communication-optimal parallel minimum spanning tree algorithms. In *Proceedings of the 10th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 27–36, 1998.
2. Foto N. Afrati, Anish Das Sarma, Semih Salihoglu, and Jeffrey D. Ullman. Upper and Lower Bounds on the Cost of a Map-Reduce Computation. *PVLDB*, 6(4):277–288, 2013.
3. Carme Àlvarez and Raymond Greenlaw. A compendium of problems complete for symmetric logarithmic space. *Computational Complexity*, 9(2):123–145, 2000.
4. Alexandr Andoni, Aleksandar Nikolov, Krzysztof Onak, and Grigory Yaroslavtsev. Parallel algorithms for geometric graph problems. In *Proceedings of the 46th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 574–583, 2014.
5. Alexandr Andoni, Clifford Stein, Zhao Song, Zhengyu Wang, and Peilin Zhong. Parallel Graph Connectivity in Log Diameter Rounds. In *Proceedings of the 59th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 674–685, 2018.
6. Alexandr Andoni, Clifford Stein, and Peilin Zhong. Log Diameter Rounds Algorithms for 2-Vertex and 2-Edge Connectivity. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 14:1–14:16, 2019.
7. Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.
8. Sepehr Assadi, MohammadHossein Bateni, Aaron Bernstein, Vahab S. Mirrokni, and Cliff Stein. 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)*, pages 1616–1635, 2019.
9. Sepehr Assadi, Yu Chen, and Sanjeev Khanna. Sublinear Algorithms for (Δ+1) Vertex Coloring. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 767–786, 2019.
10. Sepehr Assadi and Sanjeev Khanna. Tight Bounds on the Round Complexity of the Distributed Maximum Coverage Problem. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2412–2431, 2018.

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

12 Bahman Bahmani, Ravi Kumar, and Sergei Vassilvitskii. Densest Subgraph in Streaming and MapReduce. *PVLDB*, 5(5):454–465, 2012.

13 Paul Beame, Paraschos Koutris, and Dan Suciu. Communication steps for parallel query processing. *J. ACM*, 64(6), 2017.

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

15 Soheil Behnezhad, Mahsa Derakhshan, and MohammadTaghi Hajiaghayi. Semi-MapReduce Meets Congested Clique. *CoRR*, abs/1802.10297, 2018.

16 Soheil Behnezhad, Laxman Dhulipala, Hossein Esfandiari, Jakub Lacki, and Vahab Mirrokni. Near-Optimal Massively Parallel Graph Connectivity. In *Proceedings of the 60th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, 2019. To appear.

17 Soheil Behnezhad, MohammadTaghi Hajiaghayi, and David G. Harris. Exponentially Faster Massively Parallel Maximal Matching. In *Proceedings of the 60th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, 2019. To appear.

18 Gianfranco Bilardi, Michele Scquizzato, and Francesco Silvestri. A Lower Bound Technique for Communication in BSP. *ACM Trans. Parallel Comput.*, 4(3):14:1–14:27, 2018.

19 Ashok K. Chandra, Larry J. Stockmeyer, and Uzi Vishkin. Constant Depth Reducibility. *SIAM J. Comput.*, 13(2):423–439, 1984.

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

21 Stephen A. Cook and Pierre McKenzie. Problems Complete for Deterministic Logarithmic Space. *J. Algorithms*, 8(3):385–394, 1987.

22 Artur Czumaj, Jakub Lacki, Aleksander Madry, Slobodan Mitrovic, Krzysztof Onak, and Piotr Sankowski. Round compression for parallel matching algorithms. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 471–484, 2018.

23 Jeffrey Dean and Sanjay Ghemawat. MapReduce: simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, 2008.

24 Andrew Drucker, Fabian Kuhn, and Rotem Oshman. On the power of the congested clique model. In *Proceedings of the 2014 ACM Symposium on Principles of Distributed Computing (PODC)*, pages 367–376, 2014.

25 Kousha Etessami. Counting Quantifiers, Successor Relations, and Logarithmic Space. *J. Comput. Syst. Sci.*, 54(3):400–411, 1997.

26 Benjamin Fish, Jeremy Kun, Ádám Dániel Lelkes, Lev Reyzin, and György Turán. On the Computational Complexity of MapReduce. In *Proceedings of the 29th International Symposium on Distributed Computing (DISC)*, pages 1–15, 2015.

27 Fabian Frei and Koichi Wada. Efficient Circuit Simulation in MapReduce. In *Proceedings of the 30th International Symposium on Algorithms and Computation (ISAAC)*, 2019. To appear.

28 Buddhima Gamlath, Sagar Kale, Slobodan Mitrovic, and Ola Svensson. Weighted Matchings via Unweighted Augmentations. In *Proceedings of the 38th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 491–500, 2019.

29 Mohsen Ghaffari, Themis Gouleakis, Christian Konrad, Slobodan Mitrovic, and Ronitt Rubinfeld. Improved massively parallel computation algorithms for MIS, matching, and vertex cover. In *Proceedings of the 37th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 129–138, 2018.

30 Mohsen Ghaffari, Fabian Kuhn, and Jara Uitto. Conditional Hardness Results for Massively Parallel Computation from Distributed Lower Bounds. In *Proceedings of the 60th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, 2019. To appear.

31 Mohsen Ghaffari and Jara Uitto. 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)*, pages 1636–1653, 2019.

32 Michael T. Goodrich. Communication-efficient parallel sorting. *SIAM J. Comput.*, 29(2):416–432, 1999.

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

34 MohammadTaghi Hajiaghayi, Saeed Seddighin, and Xiaorui Sun. Massively Parallel Approximation Algorithms for Edit Distance and Longest Common Subsequence. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1654–1672, 2019.

35 James W. Hegeman and Sriram V. Pemmaraju. Lessons from the Congested Clique applied to MapReduce. *Theor. Comput. Sci.*, 608:268–281, 2015.

36 Sungjin Im and Benjamin Moseley. A Conditional Lower Bound on Graph Connectivity in MapReduce. *CoRR*, abs/1904.08954, 2019.

37 Sungjin Im, Benjamin Moseley, and Xiaorui Sun. Efficient massively parallel methods for dynamic programming. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 798–811, 2017.

38 Michael Isard, Mihai Budiu, Yuan Yu, Andrew Birrell, and Dennis Fetterly. Dryad: distributed data-parallel programs from sequential building blocks. In *Proceedings of the 2007 EuroSys Conference*, pages 59–72, 2007.

39 Riko Jacob, Tobias Lieber, and Nodari Sitchinava. On the Complexity of List Ranking in the Parallel External Memory Model. In *Proceedings of the 39th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 384–395, 2014.

40 Howard J. Karloff, Siddharth Suri, and Sergei Vassilvitskii. A model of computation for MapReduce. In *Proceedings of the 21st annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 938–948, 2010.

41 Hartmut Klauck, Danupon Nanongkai, Gopal Pandurangan, and Peter Robinson. Distributed Computation of Large-Scale Graph Problems. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 391–410, 2015.

42 Janne H. Korhonen and Jukka Suomela. Towards a Complexity Theory for the Congested Clique. In *Proceedings of the 30th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 163–172, 2018.

43 Ravi Kumar, Benjamin Moseley, Sergei Vassilvitskii, and Andrea Vattani. Fast Greedy Algorithms in MapReduce and Streaming. *ACM Trans. Parallel Comput.*, 2(3), 2015.

44 Jakub Lacki, Slobodan Mitrovic, Krzysztof Onak, and Piotr Sankowski. Walking Randomly, Massively, and Efficiently. *CoRR*, abs/1907.05391, 2019.

45 Silvio Lattanzi, Benjamin Moseley, Siddharth Suri, and Sergei Vassilvitskii. Filtering: a method for solving graph problems in MapReduce. In *Proceedings of the 23rd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 85–94, 2011.

46 Harry R. Lewis and Christos H. Papadimitriou. Symmetric Space-Bounded Computation. *Theor. Comput. Sci.*, 19:161–187, 1982.

47 Philip D. MacKenzie and Vijaya Ramachandran. Computational bounds for fundamental problems on general-purpose parallel models. In *Proceedings of the 10th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 152–163, 1998.

48 Krzysztof Onak. Personal communication, 2019.

49 Gopal Pandurangan, Peter Robinson, and Michele Scquizzato. Fast Distributed Algorithms for Connectivity and MST in Large Graphs. *ACM Trans. Parallel Comput.*, 5(1), 2018.

50 Gopal Pandurangan, Peter Robinson, and Michele Scquizzato. On the Distributed Complexity of Large-Scale Graph Computations. In *Proceedings of the 30th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 405–414, 2018.

51 Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

52 Andrea Pietracaprina, Geppino Pucci, Matteo Riondato, Francesco Silvestri, and Eli Upfal. Space-round tradeoffs for MapReduce computations. In *Proceedings of the 26th ACM International Conference on Supercomputing (ICS)*, pages 235–244, 2012.

53 Vibhor Rastogi, Ashwin Machanavajjhala, Laukik Chitnis, and Anish Das Sarma. Finding connected components in Map-Reduce in logarithmic rounds. In *Proceedings of the 29th IEEE International Conference on Data Engineering (ICDE)*, pages 50–61, 2013.

54 Omer Reingold. Undirected connectivity in log-space. *J. ACM*, 55(4):17:1–17:24, 2008.

55 Tim Roughgarden, Sergei Vassilvitskii, and Joshua R. Wang. Shuffles and Circuits (On Lower Bounds for Modern Parallel Computation). *J. ACM*, 65(6), 2018.

56 Michele Scquizzato and Francesco Silvestri. Communication Lower Bounds for Distributed-Memory Computations. In *Proceedings of the 31st International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 627–638, 2014.

57 Michael Sipser. *Introduction to the Theory of Computation*. PWS Publishing Company, 1997.

58 Leslie G. Valiant. A Bridging Model for Parallel Computation. *Commun. ACM*, 33(8):103–111, 1990.

59 Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the International Congress of Mathematicians 2018 (ICM 2018)*, pages 3431–3472, 2018.

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

61 Grigory Yaroslavtsev and Adithya Vadapalli. Massively parallel algorithms and hardness for single-linkage clustering under ℓ_p-distances. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 5596–5605, 2018.

62 Matei Zaharia, Reynold S. Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J. Franklin, Ali Ghodsi, Joseph Gonzalez, Scott Shenker, and Ion Stoica. Apache Spark: a unified engine for big data processing. *Commun. ACM*, 59(11):56–65, 2016.

APPENDIX

A L-Completeness of the One Cycle vs. Two Cycles Problem

In [21, Theorem 3] it is shown that graph connectivity when the given graph is known to be a disjoint union of cycles is L-hard. A careful inspection of the reductions used to establish this result reveals that the problem remains hard even when the graph is known to be made up of either one or three cycles. By reducing from a different problem, we show that it remains hard even when the graph is known to be made up of either one or two cycles.

▶ **Proposition 20.** *Graph connectivity for promise graphs that are either one cycle or two cycles is L-complete.*

Proof. Membership in L is guaranteed by the algorithm of Reingold [54]. To show L-hardness, we shall exhibit an NC^1 reduction from order between vertices. Given an instance (G, a, b) for order between vertices, we build a new graph G' as follows: (1) the two arcs pointing to a and to b, denoted (a', a) and (b', b), respectively, are removed, (2) the direction of each of the remaining $n - 3$ arcs is discarded, and (3) edges $\{s, a\}$, $\{a', b'\}$, and $\{b, t\}$ are added, where s denotes the source and t the sink of G, respectively. This construction is an NC^1 reduction. The resulting graph G' consists of two cycles if a precedes b in G, and of one single cycle otherwise. ◀