Universally-Optimal Distributed Algorithms for Known Topologies*

Bernhard Haeupler
CMU & ETH Zurich
USA & Switzerland
haeupler@cs.cmu.edu

David Wajc
Stanford University
USA
wajc@stanford.edu

Goran Zuzic
ETH Zurich
Switzerland
goran.zuzic@inf.ethz.ch

ABSTRACT

Many distributed optimization algorithms achieve *existentially-optimal* running times, meaning that there exists *some* pathological worst-case topology on which no algorithm can do better. Still, most networks of interest allow for exponentially faster algorithms. This motivates two questions:

(i) What network topology parameters determine the complexity of distributed optimization?

(ii) Are there *universally-optimal* algorithms that are as fast as possible on *every* topology?

We resolve these 25-year-old open problems in the known-topology setting (i.e., supported CONGEST) for a wide class of global network optimization problems including MST, $(1+\epsilon)$-min cut, various approximate shortest paths problems, sub-graph connectivity, etc.

In particular, we provide several (equivalent) graph parameters and show they are tight *universal lower bounds* for the above problems, fully characterizing their inherent complexity. Our results also imply that algorithms based on the low-congestion shortcut framework match the above lower bound, making them universally optimal if shortcuts are efficiently approximable.

CCS CONCEPTS

• **Mathematics of computing → Graph algorithms**; • **Theory of computation → Distributed algorithms**; *Routing and network design problems*;

KEYWORDS

Distributed Algorithms, Universal Optimality, Universal Lower Bounds, Shortcuts, Shortcut Quality

ACM Reference Format:

Bernhard Haeupler, David Wajc, and Goran Zuzic. 2021. Universally-Optimal Distributed Algorithms for Known Topologies. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC '21), June 21–25, 2021, Virtual, Italy*. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3406325.3451081

*A full version of this paper is available on arxiv [38]. Supported in part by NSF grants CCF-1527110, CCF-1618280, CCF-1814603, CCF-1910588, NSF CAREER award CCF-1750808, ONR award N000141912550, a Sloan Research Fellowship, a gift from Cisco Research, and funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (ERC grant agreement 949272).

STOC '21, June 21–25, 2021, Virtual, Italy

ACM ISBN 978-1-4503-8053-9/21/06.
https://doi.org/10.1145/3406325.3451081

1 INTRODUCTION

Much of modern large-scale graph processing and network analysis is done using systems like Google's Pregel [53], Facebook's Giraph [6, 39], or Apache's Spark GraphX [30]. These systems implement synchronous message-passing algorithms, in which nodes send (small) messages to their neighbors in each round.[1]

This has motivated a recent, broad, concentrated, and highly-successful effort to advance our theoretical understanding of such algorithms for fundamental network optimization problems, such as minimum-spanning trees (MST) [11, 14, 45, 47, 60], shortest paths [13, 18, 41–43, 49, 50, 55], flows [27], and cuts [9, 23, 56]. As a result, many fundamental optimization problems now have worst-case-optimal CONGEST algorithms, running in $\tilde{\Theta}(\sqrt{n}+D)$ rounds on every n-node network with diameter D.[2] In general, these running times cannot be improved due to unconditional lower bounds [8, 12, 62] showing that there *exist* pathological n-node topologies with small diameter on which any non-trivial optimization problem requires $\tilde{\Omega}(\sqrt{n})$ rounds. Fittingly, this type of worst-case optimality is also called *existential optimality*.

While these results are remarkable achievements, this paper emphatically argues that one cannot stop with such worst-case optimal algorithms. In particular, existential optimality says nothing about the performance of an algorithm compared to what is achievable on real-world networks, which are never worst-case, and might allow for drastically faster running times compared to a pathological worst-case instance. For example, it is a well-established fact that essentially all real-world topologies have network diameters that are very small compared to the network size [58] (this is known as the small-world effect), and essentially all mathematical models for practical networks feature diameters that are at most polylogarithmic in n. Despite this fact, for such networks the existentially-optimal algorithms take $\tilde{\Theta}(\sqrt{n})$ rounds, which is neither practically relevant, nor does it correspond to any observed practical barrier or bottleneck. This has motivated a concentrated effort in recent years to provide improved algorithms for families of networks of interest [23, 24, 26, 28, 31–35, 46]. Nonetheless, many practical networks do not fit any of the above families, and so no practically useful algorithm is known for these networks, even if we allow for preprocessing of a known network topology. This strongly motivates a broader search for algorithms which adjust to non-worst-case topologies.

[1] For concreteness, we limit message sizes to $O(\log n)$ bits, for n the number of network nodes. This is exactly the classic CONGEST model of distributed computation [61], or the supported CONGEST model [65], if the network is known and preprocessing is allowed. In what follows, we use the terms topology and network graph interchangeably.

[2] Throughout, we use $\tilde{O}, \tilde{\Omega}$ and $\tilde{\Theta}$ to suppress $\operatorname{poly}\log n$ terms. E.g., $\tilde{O}(f(n)) = O(f(n)\log^{O(1)} n)$.

1.1 When Optimal Is Not Good Enough: Universal Optimality

The search for algorithms with beyond-worst-case guarantees is not new. Indeed, it goes back at least as far as 25 years ago, to an influential paper of Garay, Kutten, and Peleg [20]. In that work, Garay et al. improved upon the existentially-optimal $O(n)$ round minimum spanning tree (MST) algorithm of Awerbuch [3].[3] In particular, they gave an $\tilde{O}(n^{0.613}+D)$ round algorithm. This was in turn improved to an $\tilde{O}(\sqrt{n}+D)$ round algorithm that is existentially-optimal in n and D by Kutten and Peleg [47]. These latter two papers started and majorly shaped the area of distributed optimization algorithms. Garay, Kutten, and Peleg informally introduced the concept of universal-optimality as the ultimate guarantee for adjusting to non-worst-case topologies:

> This type of optimality may be thought of as "existential" optimality; namely, there are points in the class of input instances under consideration for which the algorithm is optimal. A stronger type of optimality, which we may analogously call "universal" optimality, occurs when the proposed algorithm solves the problem optimally on *every* instance.
> [...]
> The interesting question that arises is, therefore, whether it is possible to identify the *inherent graph parameters* associated with the distributed complexity of various fundamental network problems, and develop *universally-optimal* algorithms. [20]

However, formalizing this concept is not as straightforward as it seems—two different readings of the above quote allows for a variety of subtly-but-crucially different formal definitions. In this paper we provide two sensible definitions. (See Section 3 for a fully formal treatment.)

An algorithm $\mathcal{A}$ is *instance optimal* if, every instance (i.e., a network G and problem-specific input), its runtime is $\tilde{O}(1)$-competitive with every other always-correct algorithm, including the fastest algorithm for that instance. While interesting and useful in more restrictive contexts, we show that instance optimality is provably unachievable for CONGEST problems like MST.

Next, we say an algorithm $\mathcal{A}$ is *universally optimal* if, for every network G, worst-case runtime of $\mathcal{A}$ over all inputs on G is $\tilde{O}(1)$-competitive with the worst-case runtime of any other always-correct algorithm, particularly the fastest algorithm for G. Equivalently, this definition (implicitly) asks about the inherent graph parameter $X_\Pi(G)$ such that there is a correct algorithm for a problem Π running in $\tilde{O}(X_\Pi(G))$ rounds on G, and any correct algorithm for Π requires $\tilde{\Omega}(X_\Pi(G))$ rounds (on some input). While it is not clear whether there exists a single algorithm with such a property for any non-trivial task, we prove such an algorithm does exist for many distributed problems.

Note that X_Π is problem-specific by definition, since different problems Π could be characterized by different graph parameters. Remarkably, we show that all the problem studied in this paper form a "universal complexity class", and share a common parameter which captures their complexity.

[3] Note that there exists a pathological worst-case topology requiring $\Omega(n)$ rounds to solve MST: an n-node ring graph.

1.2 Our Results

We resolve both questions of Garay, Kutten, and Peleg [20] in the supported CONGEST model: we identify graph parameters that fully characterize the inherent complexity of distributed global network optimization, and prove the existence of universally-optimal distributed algorithms that match those parameters.

Specifically, we show that a graph parameter ShortcutQuality(G) is a universal CONGEST lower bound for many important distributed optimization problems—meaning that every correct algorithm requires at least $\tilde{\Omega}(\text{ShortcutQuality}(G))$ rounds to compute the output on any network G. The main challenge in obtaining such a result is the very rich space of possible algorithms that might produce correct results on some specially-crafted network G, possibly outperforming shortcut-based algorithms. To show this is not possible, we utilize prior work on network coding gaps that relate general algorithms for certain simple communication problems to shortcut quality [36]. We then combine this result with a combinatorial construction that shows how to find good-quality shortcuts if there exist fast distributed algorithms for subgraph connectivity verification (Section 5), a problem which readily reduces to many other distributed optimization problems [8]. This universal lower bound holds even when the topology is known, i.e., even in the supported CONGEST model.

The parameter ShortcutQuality(G) is particularly notable since it is a key parameter in the running time of distributed algorithms that are based on the so-called "low-congestion shortcut framework" (see Section 2.2). A long line of work has shown that many distributed optimization problems can be solved in $\tilde{O}(\text{ShortcutQuality}(G))$ rounds if one can efficiently construct near-optimal shortcuts. However, such constructions were only known for special types of graphs. We obtain efficient constructions for *all* graphs in the known-topology settings by connecting the question to recent advancements in hop-constrained oblivious routings [29] (see Section 6). Putting this together, we obtain the following result.

Theorem 1.1. *[Informal] The problems MST, $(1+\varepsilon)$-minimum cut, sub-graph connectivity, various approximate shortest path problems (and more) admit a universally-optimal supported-CONGEST algorithm based on the low-congestion shortcut framework.*

Moreover, we identify several different graph parameters beside ShortcutQuality(G) that are universal CONGEST lower bounds (for the same set of problems). While the pararmeters are equivalent up to $\tilde{O}(1)$ factors, they provide different interpretations of the barriers that preclude fast algorithms for distributed network optimization.

1.3 Related Work

Probably the most well-studied global optimization problem in the distributed message-passing literature, and the one that best illustrates the search for universal optimality, is the minimum spanning tree (MST) problem. It was precisely the study of this problem which initiated the quest for universal optimality, as put forth in 1993 by Garay, Kutten, and Peleg [20, 21]. This problem was first studied in a distributed setting in the seminal work of Gallager, Humblet, and Spira [19], who gave an $O(n \log n)$-round MST algorithm. This was later improved by Awerbuch [3] to $O(n)$ rounds, which is existentially optimal in n. Garay et al. [21], advocating for a more refined analysis, moved closer to the universal lower bound of $\Omega(D)$, giving an $\tilde{O}(D+n^{0.613})$-round MST algorithm. This was improved to $\tilde{O}(D+\sqrt{n})$ by Kutten and Peleg [47]. Peleg and Rubinovich [62] constructed networks proving this bound is also existentially optimal in n and D. These networks were then used to prove lower bounds for approximate MST by Elkin [12], and many other problems by Das Sarma et al. [8]. Many algorithms matching this existentially-optimal $\tilde{O}(D+\sqrt{n})$ upper bound were obtained over the years [11, 14, 47, 60], including using the low-congestion shortcut framework [23].

The above results foreshadowed much work on studying other graph parameters which allow for improved running time for the MST problem. One example is restricting the diameter. For example, for graphs of diameter 1 (i.e., the congested clique model), a sequence of works [25, 40, 51] culminated in an $O(1)$-time algorithm [44, 59]. For small-constant diameter, Lotker et al. [52] gave an $O(\log n)$ algorithm for diameter-2 graphs, and $\tilde{\Omega}(\sqrt[3]{n})$ and $\tilde{\Omega}(\sqrt[4]{n})$ lower bounds for graphs of diameter 3 and 4, with algorithms matching these bounds recently obtained using the low-congestion shortcut framework [46]. Indeed, the shortcut framework has been the driving force behind numerous improved results for restricted graph families [23, 24, 28, 32–35, 46]. For most of these results, the worst-case shortcut quality of a graph in the graph family serves as an upper bound for these algorithms' running time. Our work shows that shortcut quality is precisely the optimal running time for *any* graph, proving that this graph parameter is a universal lower bound for distributed algorithms, and that this lower bound is achievable algorithmically by efficient supported-CONGEST algorithms.

The power of preprocessing. In this work we study CONGEST algorithms [61], both under the assumption that the topology is unknown or known to the nodes. The latter is the *supported CONGEST* model, introduced by Schmid and Suomela [65], who were motivated by Software-Defined Networking (SDN). As they argued, in SDN enabled networks, the underlying communication topology is known, while the input (e.g., edge weights, subgraph to test connectivity of, etc) may vary. It is therefore natural to preprocess the graph in advance in order to *support* the solution of possible inputs defined on this graph. In this model Ghaffari [22] gave a polynomial-time preprocessing k-broadcast algorithm which is optimal among *routing-based* distributed algorithms on the given input. It was noted in [16, 17, 65] that, while preprocessing intuitively seems very powerful, CONGEST lower bounds [1, 8] generally hold even in the supported model. Due to this, [17] asks whether preprocessing offers any benefit at all, and they offer several (specifically-made) tasks that do exhibit a separation. However, the question remains open for well-studied problems in the field. This paper offers a partial answer to this question that, up to efficient construction of near-optimal shortcuts, preprocessing does not help for many well-known problems.

Strengthened notions of optimality. Various notions of optimality similar to instance optimality have been studied in depth in many fields, and some algorithms achieving these desired properties are known in various computational models. Instance optimality with respect to a certain class of algorithms has been proven in aggregation algorithms for database systems [15], shared memory distributed algorithms [10], geometric algorithms [2], and distribution testing and learning algorithms [68, 69]. Indeed, the entire field of online algorithms concerns itself with the notion of competitive analysis, which can be seen as a form of instance optimality [5, 54]. For other models, stronger notions of optimality were long sought after, but remain elusive. For example, one of the oldest open questions in computer science is the dynamic optimality conjecture of Sleator and Tarjan [67], which states that splay trees are instance-optimal among all binary search trees. The problems studied here join the growing list of problems in various computational models for which such instance-optimal algorithms are known. This notion of instance optimality was discussed in the distributed setting by Elkin [11], who achieved instance optimality with respect to coarsening will-maintaining protocols in the LOCAL model (see Section 3.2 for a comparison between this work and our results).

1.4 Paper Outline

We define the computational models and problems studied in this paper, and discuss necessary technical background in Section 2. We then formalize the notion of universal optimality in Section 3, and point out some delicate points concerning this notion. In Section 4 we give a technical overview of the paper, presenting the key ideas behind our matching bounds for supported CONGEST, and point out a number of asymptotically equivalent tight universal bounds which follow from our work. We then substantiate our universal lower bound in Section 5, deferring the most technically involved part of this lower bound—proving the existence of disjointness gadgets—to the full version. We then present our matching supported CONGEST upper bound in terms of shortcut quality in Section 6. We conclude with a discussion of open questions in Section 7. Due to space constraints, a number of proofs are deferred to the full version of the paper, most notably a discussion of sub-diameter bounds in the supported CONGEST model. (See also discussion in Section 3.2.)

2 PRELIMINARIES

2.1 Models and Problems

We will focus on the following two models of communication.

CONGEST [61]. In this setting, a network is given as a connected undirected graph $G = (V, E)$ with diameter D and $n := |V|$ nodes. Initially, nodes know n and D and their unique $O(\log n)$-bit ID, IDs of their neighbors, and their problem-specific inputs. Communication occurs in synchronous rounds; during a round, each node

can send $O(\log n)$ bits to each of its neighbors. The goal is to design protocols that minimize the number of rounds until all nodes are guaranteed to output a solution (some global function of the problem-specific inputs) and terminate.

Supported CONGEST [65]. This setting is same as the classic CONGEST setting, with the addition that each node knows all unique IDs and the entire topology G at the start of the computation. Note that this is equivalent to the CONGEST where $\text{poly}(n)$-round preprocessing is allowed before the problem-specific input is revealed.

The following problems will be used throughout the paper. Notably, the spanning connected subgraph verification problem will serve as the core of our lower bounds.

Spanning connected subgraph [8]. A subgraph H of G is specified by having each node known which of its incident edges belong to H. The problem is solved when all nodes know whether or not H is connected and spans all nodes of G.

MST, shortest path, min-cut. Every node knows the $O(\log n)$-bit weights of each of its edges. The problem is solved when all nodes know the value of the final solution (i.e., the weight of the MST/min-cut/shortest path), and which of its adjacent edges belong to the solution.

These problem definitions follow [8] in requiring that all nodes know the weight of the final solution. This is needed in the reductions from the spanning connected subgraph problem given in [8], allowing us to leverage them in this paper. Our results seamlessly carry over to the arguably more natural problem variant where each node only needs to know which of its incident edges are in the solution (see the full paper for details).

2.2 The Low-Congestion Shortcut Framework

In this section we briefly summarize the *low-congestion shortcut framework* [23]. The framework is the state-of-the-art for message-passing algorithms for all global network optimization problems we study that go beyond worst-case topologies. This framework was devised to demonstrate that the $\tilde{\Omega}(\sqrt{n})$ lower bound [8, 62] does not hold for at least some natural topologies such as planar graphs. Since then, shortcuts have been used to achieve $o(\sqrt{n})$ running times on many different graph topologies (see [32] for an overview).

The framework introduces the following simple and natural communication problem, called *part-wise aggregation*.

Definition 2.1. *[Part-wise aggregation] Given disjoint and connected subsets of nodes $P = (P_1, \ldots, P_k)$, where $P_i \subseteq V$ are called parts, and (private) $O(\log n)$-bit input values at each node, compute within each part in parallel a simple aggregate function. For instance, each node may want to compute the minimum value in its part.*

The part-wise aggregation problem naturally arises in divide-and-conquer algorithms, such as Borůvka's MST algorithm [57], in which a network is sub-divided and simple distributed computations need to be performed in each part. More importantly and surprisingly, many other, seemingly unrelated, distributed network optimization problems like finding approximate min-cuts [23], a DFS-tree [26], or solving various (approximate) shortest path problems [31] similarly reduce to solving $\tilde{O}(1)$ part-wise aggregation instances.

Unfortunately, the parts in such applications can be very "long and windy", inducing subgraphs with very large strong diameter, even if the underlying network topology has nice properties and a small diameter. Therefore, in order to communicate efficiently, parts have to utilize edges from the rest of the graph to decrease the number of communication hops (i.e., the *dilation*). On the other hand, overusing an edge might cause *congestion* issues. Balancing between congestion and dilation naturally leads to the following definition of *low-congestion shortcuts* [23].

Definition 2.2. *[Shortcut quality] A shortcut for parts $(P_1, \ldots, P_k)$ is $(H_1, \ldots, H_k)$, where H_i is a subset of edges of G. The shortcut has **dilation** d and **congestion** c if (1) the diameter of each $G[P_i] \cup H_i$ is at most d (i.e., between every $u, v \in P_i$ there exists a path of length at most d using edges of $G[P_i] \cup H_i$), and (2) each edge e is in at most c different sets H_i. The **quality** of the shortcut is $Q = c + d$.*

Classic routing results by Leighton, Maggs and Rao [48] show that such a shortcut allows for the part-wise aggregation to be solved in $\tilde{O}(c + d)$ rounds, even distributedly. This motivates the definition of quality. We say a network topology G admits low-congestion shortcuts of quality Q if a shortcut with quality Q exists for *every* partition into disjoint connected parts of G's nodes. Denoting the minimum such quality Q by $\textsc{ShortcutQuality}(G)$, this approach leads to algorithms whose running times are parameterized by $\textsc{ShortcutQuality}(G)$.

Lemma 2.3. *[[8, 23, 31]] If G admits Q-quality shortcuts for every (valid) set of parts and such shortcuts are computable by a T-round CONGEST algorithm, then G has $\tilde{O}(Q + T)$ round CONGEST algorithms for minimum spanning tree, $(1 + \varepsilon)$-min-cut, approximate shortest-paths, and various other problems.*

Several natural classes of network topologies, including planar networks [23], networks with bounded genus, pathwidth and treewidth [32–34] or expansion [28], and all minor-closed network families [35] admit good shortcuts which can be efficiently constructed ([28, 32, 34]). Along with Lemma 2.3, this implies ultra-fast message-passing algorithms, often with $\tilde{O}(1)$ or $\tilde{O}(D)$ running times, for a wide variety of network topologies and network optimization problems. In this work we show that low-congestion shortcut-based algorithms are optimal on every network—yielding universally optimal algorithms for the problems studied here (and many more).

2.3 Moving Cuts

In this section we describe *moving cuts*, a useful tool for proving distributed information-theoretic lower bounds. Moving cuts are used to lift strong unconditional lower bounds from the classic communication complexity setting into the distributed setting. This approach was used to prove existentially-optimal (in n and D) lower bounds in Das Sarma et al. [8], and moving cuts can be seen as a generalization of their techniques. Moving cuts were only explicitly defined in [36], where they were used to prove network coding gap for simple pairwise communication tasks. (More on such tasks in Section 2.4.)

Before defining moving cuts, we briefly discuss the communication complexity model and distributed function computation problems.

Distributed computation of a Boolean function f. In this problem, two distinguished (multi-)sets of nodes $\{s_i \in V\}_{i=1}^k$ and $\{t_i \in V\}_{i=1}^k$ are given. For a given function $f : \{0,1\}^k \times \{0,1\}^k \to \{0,1\}$ and inputs $x, y \in \{0,1\}^k$, we want every node in G to learn $f(x,y)$. However, x_i and y_i are given only to s_i and t_i as their respective private inputs. Nodes in G have access to shared random coins. We are interested in the worst-case running time to complete the above task in the supported CONGEST model. The problem is motivated by the (classic) *communication complexity model*, which is the special case of CONGEST with a two-node, single-edge graph, where Alice controls one node (with k input bits) and Bob controls the other node (ditto), and *single-bit* messages are sent in each round. The time to compute f in this model is referred to as its communication complexity. In this paper we are mostly interested in the k-bit *disjointness function*, $\text{disj} : \{0,1\}^k \times \{0,1\}^k \to \{0,1\}$, given by $\text{disj}(x,y) = 1$ if for each $i \in [k]$, we have $x_i \cdot y_i = 0$, and $\text{disj}(x,y) = 0$ otherwise. I.e., if x and y are indicator vectors of sets, this function indicates whether these sets are disjoint. This function is known to have communication complexity $\Theta(k)$ [8, 64].

Having defined the distributed function computation model, we are now ready to define our lower-bound certificate on the time to compute a function between nodes $\{s_i\}_{i=1}^k$ and $\{t_i\}_{i=1}^k$: a *moving cut*.

Definition 2.4 ([36]). *Let $S = \{(s_i, t_i)\}_{i=1}^k$ be a set of source-sink pairs in a graph $G = (V, E)$. A **moving cut** for S is an assignment of positive integer edge lengths $\ell : E \to \mathbb{Z}_{\geq 1}$. We say that:*

*(i) ℓ has **capacity** $\gamma := \sum_{e \in E}(\ell_e - 1)$;*
*(ii) ℓ has **distance** β when $\text{dist}_\ell(\{s_i\}_{i=1}^k, \{t_j\}_{j=1}^k) \geq \beta$, i.e., the ℓ-distance between all sinks and sources is at least β.*

The following lemma showcases the utility of moving cuts.

Lemma 2.5. *If G contains a moving cut for k pairs $S = \{(s_i, t_i)\}_{i=1}^k$ with distance at least β and capacity strictly less than k, then distributed computation of* disj *between $\{s_i\}_{i \in [k]}$ and $\{t_i\}_{i \in [k]}$ takes $\tilde{\Omega}(\beta)$ time. This lower bound holds even for bounded-error randomized algorithms that know G and S.*

Broadly, Lemma 2.5 follows from a simulation argument. Given a sufficiently-fast $\tilde{O}(\beta)$-time distributed algorithm for disj between $\{s_i\}_{i=1}^k$ and $\{t_i\}_{i=1}^k$ and a moving cut for $\{(s_i, t_i)\}_{i=1}^k$ of capacity less than k and distance β, we show how to obtain a communication complexity protocol with a sufficiently small complexity $O(k)$ to contradict the classic $\Omega(k)$ communication complexity lower bound for disjointness. This yields the lower bound. The full proof follows the arguments (implicitly) contained in [8, 36]. See the full version of this paper for details.

Lemma 2.5 motivates the search for moving cuts of large distance and bounded capacity. For a fixed set of k pairs S, we define $\textsc{MovingCut}(S)$ to be the largest distance β of a moving cut for S of capacity strictly less than k.

2.4 Relation of Moving Cuts to Communication

Consider the simple communication problem for a set of pairs $S = \{(s_i, t_i)\}_{i=1}^k$, termed *multiple unicasts*. In this problem, each s_i has a single-bit message x_i it wishes to transmit to t_i. We denote by $\textsc{Communicating}(S)$ the time of the fastest algorithm for this problem which knows G and S (but not the messages). One natural way to solve this problem is to store-and-forward (or "route") the messages x_i through the network. We denote the fastest such algorithm's running time by $\textsc{Routing}(S)$. While faster solutions can be obtained by encoding and decoding messages in intermediary nodes, prior work has shown the gap between the fastest routing and unrestricted (e.g., coding-based) algorithms is at most $\tilde{O}(1)$ [36]. Indeed, the following lemma asserts as much, and shows that moving cuts characterize the time required to complete multiple unicasts. As the model and terminology of [36] is slightly different from ours, we provide a proof of this lemma in the full version of this paper.

Lemma 2.6. *([36]) For any set of pair S, we have that*

$$\textsc{MovingCut}(S) = \tilde{\Theta}(\textsc{Communicating}(S)) = \tilde{\Theta}(\textsc{Routing}(S)).$$

Furthermore, routing algorithms (and, by extension, moving cuts) are intricately related to shortcuts. We first extend shortcuts to (not necessarily connected) pairs in the straightforward way: given a set of pairs $S = \{(s_i, t_i)\}_{i=1}^k$ we say that a set of paths $\{H_i\}_{i=1}^k$ with endpoints $\{s_i, t_i\}$ are q-quality shortcuts if both their dilation (length of the longest path) and congestion (maximum number of paths containing any given edge) are at most q. We define $\textsc{ShortcutQuality}_2(S)$ as the minimum shortcut quality achievable for S. The seminal work of Leighton et al. [48] relates (pairwise) shortcuts and routing algorithms.

Lemma 2.7. *([48]) For any set of pairs S, we have that* $\textsc{Routing}(S) = \tilde{\Theta}(\textsc{ShortcutQuality}_2(S))$.

While the above statements hold for all sets of pairs, we will mostly be concerned with sets of pairs S which can be connected by vertex-disjoint paths in G—which we refer to as *connectable* sets of pairs. We argue that worst-case connectable pair sets characterize distributed optimization, and hence we define $\textsc{MovingCut}(G) := \max\{\textsc{MovingCut}(S) \mid S \text{ is connectable}\}$, and analogously for $\textsc{Communicating}(G)$, $\textsc{Routing}(G)$, and $\textsc{ShortcutQuality}_2(G)$. These definitions together with lemmas 2.6 and 2.7 immediately imply the following relationships.

Lemma 2.8. *For any graph G, we have*

$$\begin{aligned}\textsc{MovingCut}(G) &= \tilde{\Theta}(\textsc{Communicating}(G)) \\ &= \tilde{\Theta}(\textsc{Routing}(G)) \\ &= \tilde{\Theta}(\textsc{ShortcutQuality}_2(G)).\end{aligned}$$

2.5 Oblivious Routing Schemes

Here we revisit the multiple unicasts problem for a set of pairs $S = \{(s_i, t_i)\}_{i=1}^k$, whose complexity is captured by the parameter $\textsc{Communicating}(S)$. By lemmas 2.6 and 2.7, this parameter is also equal (up to polylog terms) to the best shortcut quality for

these pairs, $\textsc{ShortcutQuality}_2(S)$: that is, the minimum congestion+dilation over all sets of paths connecting each (s_i, t_i) pair. If all pairs are aware of each other and the topology, multiple unicasts is therefore solvable optimally (up to polylogs), using standard machinery (see [36]). However, what if each s_i needs to transmit its message to t_i without knowing other pairs (s_j, t_j) in the network? Can we achieve such near-optimal routing with the choice of s_i being *oblivious* to the other pairs? The following definitions set the groundwork needed to describe precisely such an oblivious routing scheme.

Definition 2.9. *A **routing scheme** for a graph $G = (V, E)$ is a collection $R = \{R_{s,t}\}_{s,t \in V}$ where $R_{s,t}$ is a distribution over paths between s and t. The routing scheme R has **dilation** d if for all $s, t \in V$, all paths p in the support of $R_{s,t}$ have at most d hops.*

Fix demands between pairs $\mathcal{D} : V \times V \to \{0, 1, \ldots, n^{O(1)}\}$. We say a routing scheme $R = \{R_{s,t}\}_{s,t}$ has a *fractional routing congestion* w.r.t. $\mathcal{D}$ of $\text{cong}(\mathcal{D}, R) := \max_{e \in E} \mathbb{E}_{p \sim R_{s,t}} \left[\mathcal{D}_{s,t} \cdot \mathbb{1}[e \in p]\right]$. We denote by $\text{opt}^{(h)}(\mathcal{D})$ the minimum $\text{cong}(\mathcal{D}, R)$ over all routing schemes R with dilation h. The (hop-constrained) oblivious routing problem asks whether there exists a routing scheme R which is oblivious to the demand (i.e., does not depend on $\mathcal{D}$), but which is nevertheless competitive with the optimal (hop constrained) fractional routing congestion over all demands.

Definition 2.10. *An h-**hop oblivious routing scheme** for a graph $G = (V, E)$ with **hop stretch** $\beta \geq 1$ and **congestion approximation** $\alpha \geq 1$ is a routing scheme R with dilation $\beta \cdot h$ and which for all demands $\mathcal{D} : V \times V \to \mathbb{R}_{\geq 0}$ satisfies*

$$\text{cong}(\mathcal{D}, R) \leq \alpha \cdot \text{opt}^{(h)}(\mathcal{D}).$$

The following lemma, which follows by standard Chernoff bounds, motivates the interest in such an oblivious routing in the context of computing optimal shortcuts for a set of pairs.

Lemma 2.11. *Let G be a graph, and $S = \{(s_i, t_i)\}$ be a set of pairs. Let $h \in [Q, 2Q)$, where $Q = \textsc{ShortcutQuality}_2(G)$. Finally, let R be an h-hop oblivious routing for G with hop stretch $\beta \geq 1$ and congestion approximation $\alpha \geq 1$. Then, sampling a path $p \sim R_{s_i,t_i}$ for each $(s_i, t_i) \in S$ yields $O(\log n) \cdot \max\{\alpha, \beta\} \cdot Q$ shortcuts for S w.h.p.*

The seminal result of Räcke [63] asserts that for $h = n$ (i.e., no hop constraint), an h-hop oblivious routing scheme with congestion approximation $\alpha = O(\log n)$ exists. The main result of [29] asserts that good hop-constrained oblivious routing exists for *every* hop bound h.

Theorem 2.12. *([29]) For every graph $G = (V, E)$ and $h \geq 1$, an h-hop oblivious routing with hop stretch $O(\log^6 n)$ and congestion approximation $O(\log^2 n \cdot \log^2(h \log n))$ is computable in polytime.*

3 UNIVERSAL OPTIMALITY

In this section we discuss different notions of optimality either explicitly or implicitly apparent in the literature, starting with a high-level description. We then provide a formal definition of universal and instance optimality.

3.1 Different Notions of Optimality

Existential optimality (discussed below) is the standard worst-case asymptotic notion optimality with respect to simple graph parameters prevalent in both distributed computing and throughout theoretical computer science at large.

Existential Optimality. The MST algorithm of Kutten and Peleg [47], which terminates in $\tilde{O}(D + \sqrt{n})$ rounds on every network with n nodes and diameter D, is optimal with respect to n and D in the following *existential sense*. For every n and $D = \Omega(\log n)$ there *exists* a network G with n nodes and diameter D, and a set of inputs on G, such that any algorithm that is correct on all inputs requires at least $\tilde{\Omega}(D + \sqrt{n})$ rounds.[4] However, the drawback of existential optimality is immediate: it says nothing about the performance of an algorithm compared to what is achievable on networks of interest. For example, in every planar graph one can compute the MST in $\tilde{O}(D)$ rounds, outperforming the $\tilde{\Omega}(D + \sqrt{n})$-bound when $D \ll \sqrt{n}$. Another, less immediate, drawback is that existential optimality crucially depends on the parameterization. If we parameterize only by n, one would not need to look past the $O(n)$ MST algorithm described by Awerbuch [3]. In the other extreme, one could start searching for existential optimality with respect to an ever-more-complicated set of parameters, ad nauseam.

Due to these drawbacks, Garay, Kutten, and Peleg [21] informally proposed to study stronger notions of optimality (see their quote presented in Section 1.1). Based on the quote from [21], we define the following two notions of optimality.

Instance Optimality. We say that an algorithm is *instance optimal* if it is $\tilde{O}(1)$-competitive with every other always-correct algorithm on *every network topology G* and *every valid input*.

Universal Optimality. We define universal optimality as a useful middle ground between (weaker) existential and (often unachievable) instance optimality. We say that an algorithm is *universally optimal* if for *every network G*, the *worst-case running time across all inputs* on G is $\tilde{O}(1)$-competitive with the worst-case running time of any other always-correct algorithm running on G. In other words, such an algorithm is (near-)optimal for each network topology G, when measured in terms of worst-case inputs on G.

An immediate benefit of universal optimality is that its definition is independent of any parameterization. Moreover, if a universally-optimal algorithm for a problem Π terminates in times $X_\Pi(G)$ for graph G, this implies that $X_\Pi(\cdot)$ is the fundamental graph parameter that inherently characterizes the hardest barrier in G that prevents faster algorithms from being achievable. The term *universal-network optimality* would be somewhat more descriptive, given that we are optimal for every network. However, throughout this paper we chose to keep the original term "universal optimality" coined in [20] which states the problem of identifying "inherent graph parameters" associated with different problems, and referred to the algorithms matching those bounds as universally optimal.

It is not clear at all whether universal (or instance) optimality can be achieved for a non-trivial problem. Indeed, each network (or instance) could have a single tailor-made algorithm which solves it in record time on this specific network (or instance), at the cost of being much slower on other networks (or instances). Requiring

[4] The bounds slightly change when $D = o(\log n)$, e.g., see [7, 46, 52].

a single uniform algorithm to compete with each of these exponentially many fine-tuned algorithms on every single network (or instance) simultaneously seems almost impossibly hard. Indeed, we show that this barrier prevents instance-optimal distributed MST algorithms from existing both in CONGEST and the supported CONGEST model. Surprisingly, we show that universal optimality for distributed MST and many other problems is achievable.

3.2 Formal Definitions

In this section we give formal definitions of instance and universal optimality. To our knowledge, this is the first paper that clearly separates these two notions.

For some problem Π, we say that an algorithm $\mathcal{A}$ is *always correct* if it terminates with a correct answer on every instance (i.e., every network G and every input I), and let $\mathbb{A}_\Pi$ be the class of always-correct algorithms for Π. Denote by $T_{\mathcal{A}}(G,I)$ the running time of an algorithm $\mathcal{A} \in \mathbb{A}_\Pi$ on graph G and problem-specific input I, and let $\max_I T_{\mathcal{A}}(G,I)$ be the worst-case running time of algorithm $\mathcal{A} \in \mathbb{A}_\Pi$ over all inputs supported on graph G. We define universal optimality and instance optimality for algorithms in model $\mathcal{M}$ (e.g., CONGEST, supported CONGEST, LOCAL, etc...) as follows:

Definition 3.1. *(Instance Optimality) An always-correct model-$\mathcal{M}$ algorithm $\mathcal{A}$ for problem Π is **instance optimal** if $\mathcal{A}$ is $\tilde{O}(1)$-competitive with every always-correct model-$\mathcal{M}$ algorithm $\mathcal{A}'$ for Π on every graph G, and every input I, i.e.,*

$$\forall \mathcal{A}' \in \mathbb{A}_\Pi,\ \forall G,\ \forall I \quad T_{\mathcal{A}}(G,I) = \tilde{O}(1) \cdot T_{\mathcal{A}'}(G,I).$$

Definition 3.2. *(Universal Optimality) An always-correct model-$\mathcal{M}$ algorithm $\mathcal{A}$ for problem Π is **universally optimal** if the worst-case running time of $\mathcal{A}$ on G is $\tilde{O}(1)$-competitive with that of every always-correct model-$\mathcal{M}$ algorithm $\mathcal{A}'$ for Π, i.e.,*

$$\forall \mathcal{A}' \in \mathbb{A}_\Pi,\ \forall G \quad \max_I T_{\mathcal{A}}(G,I) = \tilde{O}(1) \cdot \max_I T_{\mathcal{A}'}(G,I).$$

While similar on a surface level, these two notions have vastly different properties. Notably, in CONGEST, any instance-optimal algorithm would need to compute an answer in $\tilde{O}(D)$ rounds.

Lemma 3.3. *For every problem Π in CONGEST or supported CONGEST, every instance-optimal algorithm $\mathcal{A}$ must terminate in $\tilde{O}(D)$ rounds for every instance on a graph of diameter D.*

Proof. For any instance (G,I) given by a graph G and an input I we define an algorithm $\mathcal{A}'_{(G,I)}$ which is always-correct for Π and also very fast when run on the instance (G,I). The algorithm $\mathcal{A}'_{(G,I)}$, when run on an instance (G',I'), first computes the diameter $D(G')$ of G' together with a BFS-tree of G'. This can be done in $\Theta(D(G'))$ rounds using a standard guess/double parameter search for $D(G')$ together with a simple flooding process. Every node then checks whether a node in G has its ID and, if so, whether its input and local neighborhood looks like the one of the node with its ID in (G,I). This step requires only a single round. Furthermore, if $\mathcal{A}'_{(G,I)}$ is run on an instance (G',I') different from (G,I) at least one node in G' knows. In another $\Theta(D(G'))$ rounds the algorithm $\mathcal{A}'_{(G,I)}$ uses the BFS-tree to aggregate this information, i.e., to inform every node in G' whether the instance the algorithm is run on is identical to (G,I) or not. If the instance the algorithm $\mathcal{A}'_{(G,I)}$ is run on is the target instance (G,I) then each node terminates with a correct output after $\Theta(D(G'))$ rounds. Otherwise the algorithm $\mathcal{A}'_{(G,I)}$ runs any arbitrarily slow algorithm for Π, e.g., $\mathcal{A}'_{(G,I)}$ could use $\tilde{\Theta}(m(G'))$ rounds to aggregate all information about (G',I') along the BFS-tree and then have every node terminate with a correct answer to the instance (G',I'). Note that $\mathcal{A}'_{(G,I)}$ has a very fast running time of $\Theta(D(G')) = \Theta(D(G'))$ when run on instance $(G',I') = (G,I)$ but an incredibly slow running time of $\tilde{\Theta}(m(G'))$ when run on any other instance $(G',I') \neq (G,I)$. Still, any algorithm $\mathcal{A}'_{(G,I)}$ of this kind is always-correct for Π, i.e., $\{\mathcal{A}'_{(G,I)} | G,I\} \subseteq \mathbb{A}_\Pi$. Hence, for any instance (G',I'), the fastest always-correct algorithm terminates in at most $O(D(G'))$ rounds. The running time of any instance optimal algorithm for Π has to be at most $\tilde{O}(1) \cdot \min_{\mathcal{A}' \in \mathbb{A}_\Pi} T_{\mathcal{A}'}(G,I) = \tilde{O}(D(G))$. □

Lemma 3.3 implies that instance optimality is unattainable in the CONGEST or supported CONGEST model for all but very simple problems that can always be solved in $\tilde{O}(D)$ rounds on any topology with diameter D. In particular, instance optimality is impossible to achieve for the MST problem and all the other problems we study in this paper since, due to [8], any always-correct MST algorithm in supported CONGEST requires $\tilde{\Omega}(\sqrt{n})$ rounds on some instance supported on a network of diameter $D = O(\log n)$. So, while the notion of instance optimality has merit for other problems or models, for the problems studied here in supported CONGEST, this notion is unachievable. On the other hand, we show that universal optimality can be achieved for the problems we study in supported CONGEST.

To illustrate some differences between instance and universal optimality, we note that the latter is not directly ruled out by the $\tilde{\Omega}(\sqrt{n})$ lower bound of [8]: Consider the worst-case network G_{WC} from [8]. For this network the $\tilde{\Omega}(\sqrt{n})$ supported CONGEST lower bound applies for some input, hence for any always-correct $\mathcal{A}'$ we have that $\max_I T_{\mathcal{A}'}(G_{WC},I) \geq \tilde{\Omega}(\sqrt{n})$. When presented with G_{WC} it is therefore sufficient for a universally-optimal algorithm $\mathcal{A}$ to terminate in $\tilde{O}(\sqrt{n})$ rounds on G_{WC}.

However any universally optimal MST algorithm still has to simultaneously compete with a large collection of algorithms, each of which can be fine-tuned for a different specific network topology (while being arbitrarily slow on others). For example: suppose that $\mathcal{M}$ is CONGEST and some problem Π allows for a fast (e.g., $O(D)$ or even $O(1)$ round) CONGEST algorithm when the underlying network is promised to be a specific topology G (e.g., computing the edges of an MST can be done instantly if the topology is promised to be a (specific) tree). Then any universally-optimal algorithm for Π must complete in $\tilde{O}(D)$ rounds on any such topology G since there exists again a fine-tuned always-correct algorithms which checks for G in $O(D)$ rounds. The trouble of course is that each of the specialized algorithms only needs to run a single check,

whether the topology matches its specialty, while a universally optimal algorithm cannot run all of these checks simultaneously to be competitive, as this would be equivalent to learning the topology.

Next, we show that our definition of universal optimality (unlike the definition of instance optimality) does not require to beat the folk-lore diameter lower bound which applies to *global* problems like the minimum spanning tree requires $\Omega(D)$ rounds in CONGEST. Indeed the following lemma shows that a *universal lower bound* of $\Omega(D)$ applies to any always-correct MST algorithm. This also grants any universally-optimal algorithm MST algorithm at least $\Omega(D)$ rounds. Note that for this to hold it is crucial that we compare our running time only to always-correct algorithms and not just algorithms that are correct on the topology G they are evaluated on (or fast for).

Lemma 3.4. *For every always-correct CONGEST MST algorithm $\mathcal{A}$ and for any graph G with diameter D, the worst-case running time of $\mathcal{A}$ on instances supported on G is $\Omega(D)$, i.e., $\max_I T_{\mathcal{A}}(G, I) \geq \Omega(D)$.*

Proof. If $n \leq 2$, or more generally $D = O(1)$, the observation is trivial. Suppose therefore that $n \geq 3$ and $D \geq 3$. Let s and t be two vertices of maximum hop distance in G. That is, $d_G(s,t) = D$. Consider a vertex v at distance at least $\frac{D}{2} - 1$ from both s and t (such a vertex must exist, else $d_G(s,t) \leq D - 2$). Fix a simple path $p : s \rightsquigarrow v \rightsquigarrow t$, and let $e = (u,v)$ be some edge in p incident on v. We next consider two instances in two different graphs, G and G', obtained from G by adding edge (s,t). (Note that (s,t) is not an edge in G, else $d_G(s,t) = 1$.) The instances I in G which we consider assigns weights

$$w_{e'} = \begin{cases} 1 & e' \in p \setminus \{e\} \\ 2 & e' = e \\ n & e' \notin p, \end{cases}$$

while the instance I' in G' assigns the same weights to edges which also belong to G and weight $w_{(s,t)} = 1$ to (s,t). By application of Kruskal's MST algorithm, it is easy to show that $e = (u,v)$ is in every MST of the instance I in graph G, while it belongs to no MST of instance I' in graph G'. However, after $\frac{D}{2} - 2$ CONGEST rounds, no messages which are functions of the input of nodes s and t may reach v. Therefore, after $\frac{D}{2} - 2$ rounds, node v cannot distinguish whether the underlying topology is G or G' and it cannot determine whether $e = (u,v)$ is in the MST or not. Consequently, any always-correct MST CONGEST algorithm must spend at least $\frac{D}{2} - 1$ rounds on any diameter D graph. □

We briefly discuss various aspects of our notions of optimality.

Universal optimality in supported CONGEST. When $\mathcal{M}$ is supported CONGEST, a universally-optimal algorithm $\mathcal{A}$ can perform arbitrary computations on the network topology before the problem-specific input is revealed to it. This enlarges the space of possible universally-optimal algorithms compared to the (classic) CONGEST. On the other hand, the relative power of the "competitor algorithm" $\mathcal{A}'$ is not significantly impacted between the two models; the argument behind the proof of Lemma 3.3 implies that the running time of the best always-correct CONGEST and supported CONGEST algorithm on any input I supported on G never differ by more than an (often insignificant) $O(D)$ term.

Coarsening instance-optimal MST in LOCAL. Elkin [11] defines the class of "coarsening will-maintaining" protocols as those that, in each round, maintain a set of edges which contain an MST, and eventually converge to the correct solution. The paper considers the LOCAL model (i.e., CONGEST with *unlimited* message sizes) and concludes that one can construct instance-optimal (coarsening will-maintaining) algorithms. Specifically, the paper defines the so-called *MST-radius* $\mu(G,I)$ (a function of both the network G and the input I) and argues it is a lower bound for any always-correct LOCAL algorithm in the above class; the upper bound of $O(\mu(G,I))$ can also be achieved. The results also extend to the CONGEST model, giving a $\tilde{O}(\mu(G,I) + \sqrt{n})$, and can be argued that this algorithm is instance optimal up to an additive $\tilde{O}(\sqrt{n})$. A few notable differences between the definitions in Elkin's paper [11] and ours are imminent: in the former, protocols do not need to detect when to terminate, but rather converge towards the answer. This change of the model makes the results incomparable to ours—every always-correct CONGEST algorithm has a universal lower bound of $\Omega(D)$, which can often be significantly larger than the MST-radius.

Is the diameter always a universal lower bound? The results in this paper typically ignore additive $\tilde{O}(D)$ terms. For interesting models and problems for the scope of this paper, this choice can be formally justified with the universal lower bound of Lemma 3.4. However, our universal and instance optimality formalism is interesting even in settings where sub-diameter results are possible, i.e., where the $\Omega(D)$ lower bound does not hold. For example, suppose that we define the MST problem to be solved when each node incident to an edge e knows whether e is part of the MST or not. In the known-topology setting (i.e., supported CONGEST), when T is a tree, a universally-optimal algorithm takes $1 \ll D$ round, since each edge must be in the MST. This MST problem in supported CONGEST has the maximum diameter of a biconnected component, rather than the diameter of G, as a universal lower bound—see the full paper for an exploration of such issues. We also note that this MST problem in (non-supported) CONGEST still has an $\Omega(D)$ universal lower bound.

In the following section we outline our approach for proving the existence (and design) of universally-optimal algorithms for the problems studied in this paper.

4 TECHNICAL OVERVIEW

In this section we outline the key steps for obtaining universally-optimal algorithms in the supported CONGEST model, and highlight additional results implied by our work.

The problem we use as our running example (and as the core of our lower bounds) is the *spanning connected subgraph* verification problem (defined in Section 2). By known reductions from spanning connected subgraph verification Das Sarma et al. [8], lower bound for the above problem extend to lower bounds for MST, cut, min-cut, s-source distance, shallow light trees, min-routing cost trees and many other problems as well as to any non-trivial approximations for these problems. In order to provide universal lower bounds for these problems, we therefore prove such universal lower bounds for this verification problem.

4.1 Generalizing the Existential Lower Bound to General Topologies

To achieve our results we first give a robust definition of a *worst-case subnetwork*, which generalizes the pathological worst-case topology of the existential lower bound of Das Sarma et al. [8] to subnetworks in general graphs. This generalization builds on insights and crucial definitions from a recent work of the authors [36], which connects the CONGEST lower bound of Das Sarma et al. [8] to network coding gaps for multiple unicasts. Once the new definition is in place it is easy to verify that the proof of [8] generalizes to our worst-case subnetworks. Defining $\text{WCSubnetwork}(G)$ to be the size of the largest such worst-case subnetwork in G then gives a lower bounds for *any* network, instead of just a single graph that is carefully chosen to facilitate the lower bound proof.[5] One particularly nice aspect of this universal lower bound is that it brings the full strength and generality of the lower bound of [8] to general topologies. In particular, it applies to a myriad of different optimization and verification problems, holds for deterministic and randomized algorithm alike, holds for known topologies, and extends in full strength to any non-trivial approximations.

Lemma 4.1. *For any graph G, any always-correct supported CONGEST spanning connected subgraph verification algorithm $\mathcal{A}_G$ takes $\tilde{\Omega}(\text{WCSubnetwork}(G)+D)$ rounds on some input supported on G. This holds even if $\mathcal{A}_G$ is randomized and knows G.*

4.2 Shortcut Quality Is a Universal Lower Bound

A priori, it is not clear how strong or interesting the lower bound of $\text{WCSubnetwork}(G)$ is. By definition, it only applies to networks with subnetworks displaying similar characteristics to the pathological worst-case topology from [8], which seems very specific. Surprisingly, however, we prove an equivalence (up to polylog terms) between this graph parameter and several other graph parameters, including and most importantly a universal lower bound of $\text{ShortcutQuality}(G)$. (We elaborate on these in Section 4.4.)

Lemma 4.2. *For any graph G,*

$$\text{ShortcutQuality}(G) = \tilde{\Theta}(\text{WCSubnetwork}(G) + D).$$

From this equivalence and our universal lower bound in terms of the worst-case subnetwork, we obtain our main result: a universal lower bound in terms of the graph's shortcut quality.

Theorem 4.3. *For any graph G, any always-correct message-passing supported CONGEST algorithm $\mathcal{A}_G$ for spanning connected subgraph verification takes $\tilde{\Omega}(\text{ShortcutQuality}(G))$ rounds on some input supported on G. This holds even if $\mathcal{A}_G$ is randomized and knows G.*

[5] Indeed, Das Sarma et al. [8] state concerning their existential lower bound that "The choice of graph G is critical."

By standard reductions presented it Das Sarma et al. [8], we deduce that $\text{ShortcutQuality}(G)$ serves as a lower bound for various distributed optimization and verification problems.

Corollary 4.4. *Let G be a graph, Π be either MST, $(1+\varepsilon)$-min-cut, or approximate shortest paths, and $\mathcal{A}_G$ an always-correct supported-CONGEST algorithm for Π. Then, $\mathcal{A}_G$ takes $\tilde{\Omega}(\text{ShortcutQuality}(G))$ rounds on at least one input supported on G.*

As a corollary of Theorem 4.3 and the aforementioned reductions of [8], we find that the parameter $\text{ShortcutQuality}(G)$ is also a universal lower bound for the complexity of the very same optimization problems for which the low-congestion framework has already established algorithmic results with running times mostly depending on $\text{ShortcutQuality}(G)$. Indeed, by Lemma 2.3, an algorithm constructing $\tilde{O}(1)$-approximately optimal shortcuts in time $\tilde{O}(\text{ShortcutQuality}(G))$ would result in algorithms with running time $\tilde{O}(\text{ShortcutQuality}(G))$, which would be universally optimal, by Theorem 4.3. We provide precisely such shortcut construction in the known topology setting.

Theorem 4.5. *There exists a supported CONGEST algorithm that, for any k disjoint sets of connected parts $\{P_i \subseteq V\}_{i=1}^{k}$ in a network G, constructs $\tilde{O}(\text{ShortcutQuality}(G))$-quality shortcut on $\{P_i\}_i$ in $\tilde{O}(\text{ShortcutQuality}(G))$ rounds.*

4.3 Universal Optimality in Supported CONGEST

The above results combined directly imply universally-optimal supported CONGEST algorithms for any problem that has a good shortcut-based distributed algorithm.

Theorem 1.1. *[Informal] The problems MST, $(1+\varepsilon)$-minimum cut, sub-graph connectivity, various approximate shortest path problems (and more) admit a universally-optimal supported-CONGEST algorithm based on the low-congestion shortcut framework.*

Proof. Fix a graph G, and let $Q := \text{ShortcutQuality}(G)$. By Theorem 4.5 , there exists a supported CONGEST algorithm which for any connected parts computes $\tilde{O}(Q)$-quality shortcuts in $\tilde{O}(Q)$ time. But then, by Lemma 2.3, there exists an algorithm for computing MST, $(1+\varepsilon)$-min cut, approximate shortest paths, and spanning connected subgraph verification, all in $\tilde{O}(Q)$ rounds. Call the obtained algorithm $\mathcal{A}$. That is,

$$\max_I T_{\mathcal{A}}(G, I) = \tilde{O}(\text{ShortcutQuality}(G)).$$

On the other hand, by Theorem 4.3 and its Corollary 4.4, for any algorithm $\mathcal{A}'$, we have that

$$\max_I T_{\mathcal{A}'}(G, I) = \tilde{\Omega}(\text{ShortcutQuality}(G)).$$

Combining the above bounds, we find that indeed $\max_I T_{\mathcal{A}}(G, I) = \tilde{O}(1) \cdot \max_I T_{\mathcal{A}'}(G, I)$. As the same holds for all graphs G, we conclude that $\mathcal{A}$ is universally optimal. □

4.4 Other Characterizations of a Topology's Inherent Distributed Complexity

In this work we show that ShortcutQuality(G) is a tight universal lower bound for our problems. However, as mentioned before, identifying, understanding, and characterizing the aspects of a topology that influence and determine the complexity of distributed optimization problems is in itself a worthwhile goal. Indeed, there are a multitude of reasons why a detailed understanding of the relationship between topology and complexity is important. Among other reasons, it (a) can be important for the design of good networks, (b) might give important leads for understanding the structure of existing natural and artificial networks occurring in society, biology, and other areas, and (c) is necessary to provide quantitative and provable running time guarantees for universally-optimal algorithms run on a known topology G, beyond a simple "it runs as fast as possible".

As such, another important contribution of the tight lower bounds proven in this paper consists of giving different characterizations and ways to think about what makes a topology hard (or easy). For example, while WCSubnetwork(G) and ShortcutQuality (G) are both quantitatively equal, the fact that they both characterize the complexity of distributed optimization lends itself to very different interpretations and conclusions.

Indeed, ShortcutQuality(G) can be seen as the best routing schedules for the partwise aggregation problem, which is the very natural communication primitive underlying distributed divide-and-conquer style algorithms (see, e.g., [32]). Shortcut quality being a tight universal lower bound further demonstrates the key role partwise aggregation plays for distributed optimization algorithms, even to the extent that the complexity of many very different optimization tasks is dominated by how fast this simple aggregation procedure can be performed on a given topology.

The tightness of WCSubnetwork(G) as a lower bound, on the other hand, points to the pathological network structure identified by Peleg and Rubinovich [8, 62] as indeed the only way in which a topology can be hard for optimization. Put otherwise, a topology is exactly as hard as the worst obstruction of this type within a network.

As part of our proof of Theorem 4.2 we identify, define, and expose several other graph parameters which similarly characterize the complexity of a topology G, such as, MovingCut(G), Routing (G) and others. Many of these parameters have very different flavors. For example the MovingCut(G) parameter can be seen as identifying crucial communication bottlenecks within a topology via a sequence of cuts. It is also known [36] to characterize the time needed to solve a simple multiple unicast communication problem which requires information to be sent between different sender-receiver pairs in the network. Routing(G) relates to the same communication problem, but with the restriction that information is routed (without any coding) which, by Leighton, Maggs, and Rao [48], is equivalent to the best congestion and dilation of paths connecting the sender-receiver pairs. We give precise definitions and further explanations for these and other equivalent universal lower bound parameters in the technical sections of this paper. We hope that they will help to further illuminate different aspects of the topology-complexity interplay.

5 SHORTCUT QUALITY IS A UNIVERSAL LOWER BOUND

In this section we present our proof of our universal lower bounds in terms of shortcut quality. In particular, this section is dedicated to proving the following theorem.

Theorem 5.1. *Let $\mathcal{A}$ be any always-correct algorithm for spanning connected subgraph and let $T_{conn}(G) = \max_I T_{\mathcal{A}}(G, I)$ denote the worst-case running time of $\mathcal{A}$ on the network G. Then we have that:*

$$T_{conn}(G) = \tilde{\Omega}(\textsc{ShortcutQuality}(G)).$$

We defer most proofs of this section to the full version, focusing only on a high-level overview here. We start by introducing *disjointness gadgets*, which are pathological sub-graphs for distributed optimization, and outline their use in proving distributed lower bounds, in Section 5.1. In order to obtain informative lower bounds from these gadgets, we then relate the worst such subgraph to the highest distance of any moving cut in G, MovingCut(G), in Section 5.2. This is the technical meat of this work, and a non-negligible fraction of the full version of this paper is dedicated to proving this relation. We then relate the obtained lower bounds to shortcut quality in Section 5.3. Finally, we conclude with the proof of Theorem 5.1, as well as discussions of its implications to other distributed problems, in Section 5.4.

5.1 Lower Bound Witnesses

In this section we define *β-disjointness gadgets*, a structure that connects together information-theoretic bounds with higher-level distributed optimization problems like MST. The structure can be seen as a generalization of previous *existential* lower bounds that show many distributed problems cannot be solved faster than $\tilde{\Omega}(D + \sqrt{n})$ on a specific graph family [8, 12, 62]. We argue that β-disjointness gadgets are the "right" way to generalize their approaches to arbitrary graphs.

Definition 5.2. *A β-*disjointness gadget (P, T, ℓ) *in graph G consists of a set of vertex-disjoint paths $P \neq \emptyset$, each of length at least three; a tree $T \subseteq G$ which intersects each path in P exactly at its endpoint vertices; and a moving cut of capacity strictly less than $|P|$ and distance β with respect to the pairs $\{(s_i, t_i)\}_{i=1}^{|P|}$ of endpoints of paths $p_i \in P$.*

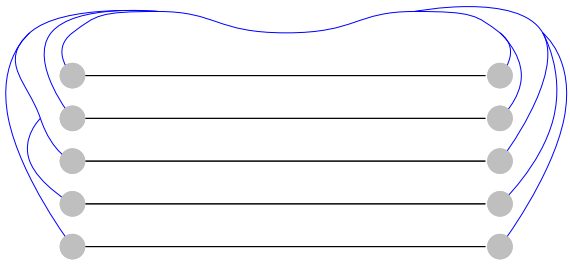

Figure 1: A disjointness gadget's path and tree, given by straight and rounded blue lines, respectively.

As we show, such disjointness gadgets are precisely the worst-case subgraphs which cause distributed verification (and optimization) to be hard. In particular, denoting by WCSubnetwork(G) the

highest value of β for which there exists a β-disjointness gadget in G (or zero, if none exists). This quantifies the most pathological subgraph in G. We prove the following.

Lemma 5.3. *Let $\mathcal{A}$ be any always-correct algorithm for spanning connected subgraph and let $T_{conn}(G) = \max_I T_{\mathcal{A}}(G, I)$ denote the worst-case running time of $\mathcal{A}$ on the network G. Then we have that:*

$$T_{conn}(G) = \tilde{\Omega}(\textit{WCSubnetwork}(G) + D).$$

The non-trivial part of this lemma is the lower bound $T_{conn}(G) = \tilde{\Omega}(\text{WCSubnetwork}(G))$. Our proof of this bound (see full version) follows the approach implicit in [8]. Broadly, we use a disjointness gadget (P, T, ℓ) to construct a subgraph H determined by private $|P|$-bit inputs x, y for the endpoints of the paths, such that H is a spanning and connected subgraph of G if and only if $\text{disj}(x, y) = 1$. Combined with Lemma 2.5, this equivalence and the moving cut ℓ yield a lower bound on subgraph connectivity in a graph G containing a β-disjointness gadget. In following sections we show how to use this bound to prove lower bounds for this problem in *any* graph G.

5.2 Disjointness Gadgets in Any Graph

The first challenge in deriving an informative lower bound on the time for spanning connected subgraph verification from Lemma 5.3 is that graphs need not contain disjointness gadgets. For example, as disjointness gadgets induce cycles, trivially no such gadgets exist in a tree. Consequently, for trees Lemma 5.3 only recreates the trivial lower bound of $\tilde{\Omega}(D)$.

The following theorem implies that for any graphs where the parameter $\text{MovingCut}(G)$ is sufficiently larger than D, disjointness gadgets *do* exist. More precisely, we prove the following theorem.

Theorem 5.4. *For any graph G,*

$$\textit{WCSubnetwork}(G) + D = \tilde{\Theta}(\textit{MovingCut}(G)).$$

Theorem 5.4 is the technical core of this paper, and much of the paper's full version is dedicated to its proof. At a (very) high level, what we prove there is that, while disjointness gadgets do not always exist, some relaxation of them always does. In particular, we show that for any graph G and set of connectable pairs S in G, some relaxed notion of disjointness gadgets always exists for a subset $S' \subseteq S$ of size $|S'| = \Omega(|S|)$. We then show how to extend a moving cut of distance $\beta \geq 9D$ on S to (strict) disjointness gadgets: construct a relaxed disjointness gadget on a large subset of S (since S are connectable), then clean-up the structure using $\beta \geq 9D$ to transform it to a (strict) β-disjointness gadget.

5.3 Relating $\text{MovingCut}(G)$ to ShortcutQuality(G)

So far we have shown that (up to polylog multiplicative terms and additive $O(D)$ terms), the time to solve subgraph connectivity is at least the length of the worst moving cut in G, which we denote by MovingCut(G). More precisely, so far we proved that

$$T_{conn}(G) \geq \tilde{\Theta}(\text{WCSubnetwork}(G) + D) = \tilde{\Theta}(\text{MovingCut}(G)).$$

In this section we show that the above terms we have proven to be equivalent (up to polylog factors) are in turn equivalent to the graph's shortcut quality.

Indeed, by lemmas 2.6 and 2.7, we have that $\text{MovingCut}(G) = \tilde{\Theta}(\text{ShortcutQuality}_2(G))$. The following lemma proves an equivalence (up to polylog factors) between shortcut quality for pairs to the graph's shortcut quality (for parts).

Lemma 5.5. *For any graph G,*

$$\textit{ShortcutQuality}(G) = \tilde{\Theta}(\textit{ShortcutQuality}_2(G)).$$

Broadly, we use heavy-light decompositions [66] of spanning trees of parts, to show how to obtain shortcuts for parts by gluing together a polylogarithmic number of shortcuts for connected pairs.[6] The overall dilation and congestion of the obtained shortcuts for the parts are at most polylogarithmically worse than those of the shortcuts for the pairs. (See the full version for a proof.)

5.4 Putting It All Together

In this section we review our main result, whereby shortcut quality serves as a universal lower bound for the spanning connected subgraph problem, as well as numerous other problems.

Theorem 5.1. *Let $\mathcal{A}$ be any always-correct algorithm for spanning connected subgraph and let $T_{conn}(G) = \max_I T_{\mathcal{A}}(G, I)$ denote the worst-case running time of $\mathcal{A}$ on the network G. Then we have that:*

$$T_{conn}(G) = \tilde{\Omega}(\textit{ShortcutQuality}(G)).$$

Proof. Putting all the lemmas above together, we have

$$\begin{aligned}
T_{conn}(G) &\geq \tilde{\Theta}(\text{WCSubnetwork}(G) + D) && \textit{Lemma 5.3}\\
&= \tilde{\Theta}(\text{MovingCut}(G)) && \textit{Theorem 5.4}\\
&= \tilde{\Theta}(\text{Communicating}(G)) && \textit{Lemma 2.8}\\
&= \tilde{\Theta}(\text{Routing}(G)) && \textit{Lemma 2.8}\\
&= \tilde{\Theta}(\text{ShortcutQuality}_2(G)) && \textit{Lemma 2.8}\\
&= \tilde{\Theta}(\text{ShortcutQuality}(G)) && \textit{Lemma 5.5}
\end{aligned}$$ □

We note that the above proof entails a proof of Lemma 4.2, as well as the equivalence between the number of tight universal lower bounds for our problems discussed in Section 4.4.

Known reductions presented in Das Sarma et al. [8] extend the same universal lower bounds of Theorem 5.1 to numerous problems such as the MST, shallow-light tree, SSSP, min-cut and others. The reductions hold for both non-trivial approximation factors as well as randomized algorithms.

Since MST (and all above problems, for some approximation ratios) can be solved using $\tilde{O}(1)$ applications of partwise aggregation, Theorem 5.1 implies a similar $\tilde{\Omega}(\text{ShortcutQuality}(G))$ lower bound for the partwise aggregation problem. In Section 6 we present a polytime algorithm matching this lower bound, resulting in polytime universally-optimal supported CONGEST algorithms for all problems studied in this paper.

[6] See [37] for a similar application of heavy-light decompositions to the reduction of multicast routing to unicast routing.

6 MATCHING THE LOWER BOUND: ALGORITHMS FOR SHORTCUT CONSTRUCTION

In this section we give our shortcut construction for the supported CONGEST model which efficiently constructs shortcuts of quality $\tilde{O}(Q)$ in $\tilde{O}(Q)$ rounds, where $Q = \textsc{ShortcutQuality}(G)$ is the best possible shortcut quality.

Theorem 4.5. *There exists a supported CONGEST algorithm that, for any k disjoint sets of connected parts $\{P_i \subseteq V\}_{i=1}^k$ in a network G, constructs $\tilde{O}(\textsc{ShortcutQuality}(G))$-quality shortcut on $\{P_i\}_i$ in $\tilde{O}(\textsc{ShortcutQuality}(G))$ rounds.*

By Lemma 2.3, numerous problems, including MST, approximate min-cut, approximate shortest path problems, and verification problems can therefore be solved in $\tilde{O}(\textsc{ShortcutQuality}(G))$ rounds when the topology is known. On the other hand, from Section 5 we know that all these problems are harder than the subgraph connectivity problem, which requires at least $\tilde{\Omega}(\textsc{ShortcutQuality}(G))$ rounds on any network G, by Theorem 5.1. These matching bounds together give Theorem 1.1.

6.1 Constructing Shortcuts for Pairs

In order to construct shortcuts for parts, we will rely on the ability to construct shortcuts for pairs. In particular, we will require such shortcut construction for pairs of nodes which are oblivious of each other. To this end, we rely on the existence of hop-constrained oblivious routing to construct such shortcuts between pairs of nodes, yielding the following lemma.

Lemma 6.1. *There exists a supported CONGEST algorithm that, given k disjoint node pairs $S = \{(s_i, t_i)\}_i$ (each s_i knows t_i and vice versa, but not other pairs) in a graph G, constructs an $\tilde{O}(Q)$-quality shortcut for S in $\tilde{O}(Q)$ rounds, where $Q = \textsc{ShortcutQuality}_2(G)$.*

Proof. Knowing the topology, for $h = 2^1, 2^2, \ldots, 2^{\lceil \log_2 n \rceil}$, all nodes internally compute (the same) h-hop oblivious routing with $\tilde{O}(1)$ hop stretch and $\tilde{O}(1)$ congestion approximation for all values h, denoted by $\{R^h_{s,t}\}_{s,t,h}$. Sampling these distributions then gives paths $p^h_{s,t} \sim R^h_{s,t}$ for each such triple (s, t, h). By Lemma 2.11 and Theorem 2.12, for $h \in [Q, 2Q)$, the set of paths $p^h_{s_i,t_i}$ form shortcuts for S of quality $q := O(Q \cdot \log^7 n) = O(h \cdot \log^7 n)$.

During the shortcut construction stage, we appeal to the random-delay-based routing protocol which implies that $\textsc{Routing}(S) = \tilde{\Theta}(\textsc{ShortcutQuality}_2(S))$ (i.e., Lemma 2.7). In particular, for $h = 2^1, 2^2, \ldots, 2^{\lceil \log_2 n \rceil}$, for each pair (s_i, t_i), we send a message between s_i and t_i via the path p_i, starting at a uniformly randomly chosen time in $[q]$, and send this message during $O(q \cdot \log n)$ rounds. We let this message contain the identifiers of s_i and t_i, and so intermediary nodes, which all know $p^h_{s_i,t_i}$, can forward this message along the path. By standard random-delay arguments [48], if the paths $p^h_{s_i,t_i}$ are shortcuts of quality q, then all pairs (s_i, t_i) will have both of their messages delivered w.h.p. To verify whether or not all sinks t_i receive their message, after these $O(q \cdot \log n)$ rounds, all sinks t_i flood a single-bit message through the system, indicating whether any sink has not received its designated message from s_i. This step takes $O(D)$ rounds. Now, since by Lemma 2.11 we know that for $h \in [Q, 2Q]$, the sampled shortcuts are $q = \tilde{O}(h)$-quality shortcuts w.h.p., and since $D \le Q = \textsc{ShortcutQuality}_2(G)$, this algorithm terminates successfully after

$$\sum_{k=1}^{\log_2 Q+1} (\tilde{O}(1) \cdot 2^i + D) = \tilde{O}(Q)$$

rounds w.h.p. □

Using this $\tilde{O}(\textsc{ShortcutQuality}_2(G))$-quality shortcut construction algorithm for pairs, we tackle the more challenging problem of $\tilde{O}(\textsc{ShortcutQuality}(G))$-quality shortcuts construction for parts.

6.2 Lifting Pair Shortcuts to Part Shortcuts

In the full version we describe how to use the shortcut construction for pairs to construct general low-congestion shortcuts (using heavy-light tree decompositions), giving the following lemma.

Lemma 6.2. *Suppose there exists an algorithm that for input connectable pairs $\mathcal{S} = \{(s_i, t_i)\}_{i=1}^k$ (each s_i knows t_i and vice versa, but not other pairs) outputs a shortcut for $\mathcal{S}$ with quality $\tilde{O}(Q)$ in T rounds, where $Q = \textsc{ShortcutQuality}_2(G)$. Then there exists a randomized CONGEST algorithm that on input disjoint and connected parts $\mathcal{P} = (P_1, \ldots, P_k)$ (with each node only knowing the i for which $v \in P_i$, if any), constructs a shortcut for $\mathcal{P}$ of quality $\tilde{O}(Q)$ in $\tilde{O}(T)$ rounds w.h.p.*

Finally, invoking Lemma 6.2 with Lemma 6.1 as its pairwise shortcut algorithm, we obtain a supported CONGEST algorithm which constructs $\tilde{O}(Q)$-quality shortcuts for any disjoint connected parts in $\tilde{O}(Q)$, for $Q = \textsc{ShortcutQuality}(G)$. That is, we have proved Theorem 4.5.

7 CONCLUSIONS AND OPEN QUESTIONS

In this work we give the first non-trivial universally-optimal distributed algorithms for a number of optimization and verification problems in the supported CONGEST model of communication. In particular, we show that the low-congestion framework yields such universally-optimal algorithms. This framework has since shown great promise as a basis for a unified description of the common communication bottlenecks underlying many distributed network optimization problems.

This work suggests a number of natural follow-up questions, of which we mention a few here.

Universally-optimal CONGEST algorithms. Our algorithmic results require efficient computation of shortcuts, in time at most $\tilde{O}(\textsc{ShortcutQuality}(G))$. Using the recent oblivious routing result of [29], we showed how to achieve this efficiently in the supported CONGEST model. Can such shortcuts be computed in the CONGEST model, i.e., without pre-processing or knowledge of G? A positive answer to this question would yield universally-optimal CONGEST algorithms for the problems tackled by the low-congestion shortcut framework.

Universally-optimal algorithms for more problems. Our work proves that the low-congestion framework, which yields existentially-optimal $\tilde{O}(D + \sqrt{n})$-time algorithms for numerous distributed optimization problems (see [8, 25]), in fact yields universally-optimal

algorithms for these problems. However, some fundamental problems remain for which shortcut quality serves as a universal lower bound (by our work), yet shortcut-based algorithms are not currently known. One such problem is *exact* min-cut, for which an existentially-optimal algorithm was recently given in [9]. Another such problem is (better) approximate SSSP, for which the best approximation guarantee of a shortcut-time algorithm is polynomial [31], while the best existentially-optimal algorithm yields a $(1+\varepsilon)$ approximation [4]. Our work motivates the study of shortcut-based (and shortcut-quality time) algorithms for these and other problems, as such algorithms would be universally optimal, by our work.

Universally-Optimal Round and Message Complexity. Another well-studied complexity measure of message-passing algorithms is their *message complexity*, i.e., the number of messages they send during their execution. Pandurangan et al. [60] showed that existentially-optimal time and message complexities are achievable simultaneously, resolving a longstanding open problem. This was then shown to be achievable deterministically, by Elkin [14], and then shown to be achievable within the shortcut framework, in [34]. We note, however, that [34] relied specifically on tree-restricted shortcuts. Can one remove this restriction? A positive resolution to this question would yield algorithms with *universally-optimal* time complexity, and optimal message complexity.

Other universal barriers to distributed computation. Our work shows that for a wide family of problems for which $\tilde{\Theta}(D+\sqrt{n})$ serves as a tight existential bound, shortcut quality serves as a tight universal bound. Can similar tight universal bounds be proven for problems outside this "complexity class"?

REFERENCES

[1] Amir Abboud, Keren Censor-Hillel, Seri Khoury, and Ami Paz. Smaller cuts, higher lower bounds. *arXiv preprint arXiv:1901.01630*, 2019.

[2] Peyman Afshani, Jérémy Barbay, and Timothy M Chan. Instance-optimal geometric algorithms. *Journal of the ACM (JACM)*, 64(1):1–38, 2017.

[3] Baruch Awerbuch. Optimal distributed algorithms for minimum weight spanning tree, counting, leader election, and related problems. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC)*, pages 230–240, 1987.

[4] Ruben Becker, Andreas Karrenbauer, Sebastian Krinninger, and Christoph Lenzen. Near-optimal approximate shortest paths and transshipment in distributed and streaming models. In *Proceedings of the 31st International Symposium on Distributed Computing (DISC)*, 2017.

[5] Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. cambridge university press, 2005.

[6] Avery Ching, Sergey Edunov, Maja Kabiljo, Dionysios Logothetis, and Sambavi Muthukrishnan. One trillion edges: Graph processing at facebook-scale. *Proceedings of the VLDB Endowment*, 8(12):1804–1815, 2015.

[7] Julia Chuzhoy, Merav Parter, and Zihan Tan. On packing low-diameter spanning trees. In *Proceedings of the 47th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 33:1–33:18, 2020.

[8] Atish Das Sarma, Stephan Holzer, Liah Kor, Amos Korman, Danupon Nanongkai, Gopal Pandurangan, David Peleg, and Roger Wattenhofer. Distributed verification and hardness of distributed approximation. *SIAM Journal on Computing (SICOMP)*, 41(5):1235–1265, 2012.

[9] Michal Dory, Yuval Efron, Sagnik Mukhopadhyay, and Danupon Nanongkai. Distributed weighted min-cut in nearly-optimal time. In *Proceedings of the 53rd Annual ACM Symposium on Theory of Computing (STOC)*, page To appear, 2021.

[10] Cynthia Dwork and Yoram Moses. Knowledge and common knowledge in a byzantine environment i: crash failures. In *Theoretical Aspects of Reasoning about Knowledge*, pages 149–169, 1986.

[11] Michael Elkin. A faster distributed protocol for constructing a minimum spanning tree. *Journal of Computer and System Sciences*, 72(8):1282–1308, 2006.

[12] Michael Elkin. An unconditional lower bound on the time-approximation tradeoff for the distributed minimum spanning tree problem. *SIAM Journal on Computing (SICOMP)*, 36(2):433–456, 2006.

[13] Michael Elkin. Distributed exact shortest paths in sublinear time. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 757–770, 2017.

[14] Michael Elkin. A simple deterministic distributed MST algorithm, with near-optimal time and message complexities. In *Proceedings of the 36th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 157–163, 2017.

[15] Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. *Journal of computer and system sciences*, 66(4):614–656, 2003.

[16] Klaus-Tycho Foerster, Juho Hirvonen, Stefan Schmid, and Jukka Suomela. On the power of preprocessing in decentralized network optimization. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 1450–1458, 2019.

[17] Klaus-Tycho Foerster, Janne H Korhonen, Joel Rybicki, and Stefan Schmid. Does preprocessing help under congestion? In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 259–261, 2019.

[18] Silvio Frischknecht, Stephan Holzer, and Roger Wattenhofer. Networks cannot compute their diameter in sublinear time. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1150–1162, 2012.

[19] Robert G. Gallager, Pierre A. Humblet, and Philip M. Spira. A distributed algorithm for minimum-weight spanning trees. *ACM Transactions on Programming Languages and Systems*, 5(1):66–77, 1983.

[20] Juan A Garay, Shay Kutten, and David Peleg. A sublinear time distributed algorithm for minimum-weight spanning trees. In *Proceedings of the 34th Symposium on Foundations of Computer Science (FOCS)*, pages 659–668, 1993.

[21] Juan A Garay, Shay Kutten, and David Peleg. A sublinear time distributed algorithm for minimum-weight spanning trees. *SIAM Journal on Computing (SICOMP)*, 27(1):302–316, 1998.

[22] Mohsen Ghaffari. Distributed broadcast revisited: Towards universal optimality. In *Proceedings of the 42nd International Colloquium on Automata, Languages and Programming (ICALP)*, pages 638–649, 2015.

[23] Mohsen Ghaffari and Bernhard Haeupler. Distributed algorithms for planar networks II: Low-congestion shortcuts, mst, and min-cut. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 202–219, 2016.

[24] Mohsen Ghaffari and Jason Li. New distributed algorithms in almost mixing time via transformations from parallel algorithms. In *Proceedings of the 32nd International Symposium on Distributed Computing (DISC)*, pages 31:1–31:16, 2018.

[25] Mohsen Ghaffari and Merav Parter. MST in log-star rounds of congested clique. In *Proceedings of the 35th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 19–28, 2016.

[26] Mohsen Ghaffari and Merav Parter. Near-optimal distributed DFS in planar graphs. *Proceedings of the 31st International Symposium on Distributed Computing (DISC)*, 91:21, 2017.

[27] Mohsen Ghaffari, Andreas Karrenbauer, Fabian Kuhn, Christoph Lenzen, and Boaz Patt-Shamir. Near-optimal distributed maximum flow. In *Proceedings of the 34th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 81–90, 2015.

[28] Mohsen Ghaffari, Fabian Kuhn, and Hsin-Hao Su. Distributed MST and routing in almost mixing time. In *Proceedings of the 38th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 131–140, 2017.

[29] Mohsen Ghaffari, Bernhard Haeupler, and Goran Zuzic. Hop-constrained oblivious routing. page To appear, 2021.

[30] Joseph E Gonzalez, Reynold S Xin, Ankur Dave, Daniel Crankshaw, Michael J Franklin, and Ion Stoica. Graphx: Graph processing in a distributed dataflow framework. In *Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 599–613, 2014.

[31] Bernhard Haeupler and Jason Li. Faster distributed shortest path approximations via shortcuts. In *Proceedings of the 32nd International Symposium on Distributed Computing (DISC)*, pages 33:1–33:14, 2018.

[32] Bernhard Haeupler, Taisuke Izumi, and Goran Zuzic. Low-congestion shortcuts without embedding. In *Proceedings of the 35th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 451–460, 2016.

[33] Bernhard Haeupler, Taisuke Izumi, and Goran Zuzic. Near-optimal low-congestion shortcuts on bounded parameter graphs. In *Proceedings of the 30th International Symposium on Distributed Computing (DISC)*, pages 158–172, 2016.

[34] Bernhard Haeupler, D Ellis Hershkowitz, and David Wajc. Round-and message-optimal distributed graph algorithms. In *Proceedings of the 37th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 119–128, 2018.

[35] Bernhard Haeupler, Jason Li, and Goran Zuzic. Minor excluded network families admit fast distributed algorithms. In *Proceedings of the 39th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 465–474, 2018.

[36] Bernhard Haeupler, David Wajc, and Goran Zuzic. Network coding gaps for completion times of multiple unicasts. In *Proceedings of the 61st Symposium on Foundations of Computer Science (FOCS)*, pages 494–505, 2020.

[37] Bernhard Haeupler, D Ellis Hershkowitz, and David Wajc. Near-optimal schedules for simultaneous multicasts. In *Proceedings of the 48th International Colloquium on Automata, Languages and Programming (ICALP)*, page To appear, 2021.

[38] Bernhard Haeupler, David Wajc, and Goran Zuzic. Universally-optimal distributed algorithms for known topologies. *arXiv preprint arXiv:2104.03932*, 2021.

[39] Minyang Han and Khuzaima Daudjee. Giraph unchained: Barrierless asynchronous parallel execution in pregel-like graph processing systems. *Proceedings of the VLDB Endowment*, 8(9):950–961, 2015.

[40] James W Hegeman, Gopal Pandurangan, Sriram V Pemmaraju, Vivek B Sardeshmukh, and Michele Scquizzato. Toward optimal bounds in the congested clique: Graph connectivity and MST. In *Proceedings of the 34th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 91–100, 2015.

[41] Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. An almost-tight distributed algorithm for computing single-source shortest paths. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, 2016.

[42] Stephan Holzer and Roger Wattenhofer. Optimal distributed all pairs shortest paths and applications. In *Proceedings of the ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, pages 355–364, 2012.

[43] C. Huang, D. Nanongkai, and T. Saranurak. Distributed exact weighted all-pairs shortest paths in $\tilde{O}(n^{5/4})$ rounds. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 168–179, 2017.

[44] Tomasz Jurdziński and Krzysztof Nowicki. MST in $O(1)$ rounds of congested clique. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2620–2632, 2018.

[45] Maleq Khan and Gopal Pandurangan. A fast distributed approximation algorithm for minimum spanning trees. *Distributed Computing*, 20(6):391–402, 2008.

[46] Naoki Kitamura, Hirotaka Kitagawa, Yota Otachi, and Taisuke Izumi. Low-congestion shortcut and graph parameters. In *Proceedings of the 33rd International Symposium on Distributed Computing (DISC)*, pages 25:1–25:17, 2019.

[47] Shay Kutten and David Peleg. Fast distributed construction of smallk-dominating sets and applications. *Journal of Algorithms*, 28(1):40–66, 1998.

[48] Frank Thomson Leighton, Bruce M Maggs, and Satish B Rao. Packet routing and job-shop scheduling in O(congestion+ dilation) steps. *Combinatorica*, 14(2): 167–186, 1994.

[49] Christoph Lenzen and Boaz Patt-Shamir. Fast partial distance estimation and applications. *Proceedings of the ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, pages 153–162, 2015.

[50] Christoph Lenzen and David Peleg. Efficient distributed source detection with limited bandwidth. *Proceedings of the ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, pages 375–382, 2013.

[51] Zvi Lotker, Elan Pavlov, Boaz Patt-Shamir, and David Peleg. MST construction in $O(\log \log n)$ communication rounds. *SIAM Journal on Computing (SICOMP)*, 35(1):120–131, 2005.

[52] Zvi Lotker, Boaz Patt-Shamir, and David Peleg. Distributed MST for constant diameter graphs. *Distributed Computing*, 18(6):453–460, 2006.

[53] Grzegorz Malewicz, Matthew H Austern, Aart JC Bik, James C Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. Pregel: a system for large-scale graph processing. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 135–146, 2010.

[54] Mark S Manasse, Lyle A McGeoch, and Daniel D Sleator. Competitive algorithms for server problems. *Journal of Algorithms*, 11(2):208–230, 1990.

[55] Danupon Nanongkai. Distributed approximation algorithms for weighted shortest paths. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*, pages 565–573, 2014.

[56] Danupon Nanongkai and Hsin-Hao Su. Almost-tight distributed minimum cut algorithms. In *Proceedings of the 28th International Symposium on Distributed Computing (DISC)*, pages 439–453, 2014.

[57] Jaroslav Nešetřil, Eva Milková, and Helena Nešetřilová. Otakar boruvka on minimum spanning tree problem translation of both the 1926 papers, comments, history. *Discrete Mathematics*, 233(1):3–36, 2001.

[58] Mark EJ Newman. The structure and function of complex networks. *SIAM review*, 45(2):167–256, 2003.

[59] Krzysztof Nowicki. A deterministic algorithm for the mst problem in constant rounds of congested clique. page To appear, 2021.

[60] Gopal Pandurangan, Peter Robinson, and Michele Scquizzato. A time- and message-optimal distributed algorithm for minimum spanning trees. In *Proceedings of the 49th Annual ACM Symposium on Theory of Computing (STOC)*, pages 743–756, 2017.

[61] David Peleg. *Distributed computing: A Locality-Sensitive Approach*, volume 5. SIAM, 2000.

[62] David Peleg and Vitaly Rubinovich. A near-tight lower bound on the time complexity of distributed minimum-weight spanning tree construction. *SIAM Journal on Computing (SICOMP)*, 30(5):1427–1442, May 2000.

[63] Harald Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 255–264, 2008.

[64] Alexander A Razborov. On the distributional complexity of disjointness. In *Proceedings of the 17th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 249–253, 1990.

[65] Stefan Schmid and Jukka Suomela. Exploiting locality in distributed sdn control. In *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pages 121–126, 2013.

[66] Daniel D Sleator and Robert Endre Tarjan. A data structure for dynamic trees. *Journal of Computer and System Sciences*, 26(3):362–391, 1983.

[67] Daniel Dominic Sleator and Robert Endre Tarjan. Self-adjusting binary search trees. *Journal of the ACM (JACM)*, 32(3):652–686, 1985.

[68] Gregory Valiant and Paul Valiant. Instance optimal learning of discrete distributions. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing (STOC)*, pages 142–155, 2016.

[69] Gregory Valiant and Paul Valiant. An automatic inequality prover and instance optimal identity testing. *SIAM Journal on Computing (SICOMP)*, 46(1):429–455, 2017.