Thesis Proposal:
Distributed Optimization Beyond Worst-Case Topologies

Goran Zuzic

April 15, 2019

Abstract

We propose to develop an algorithmic toolbox for designing distributed optimization algorithms which drastically outperform state-of-the-art algorithms on non-worst-case topologies.

The modern computation and information processing systems shaping our world have become massively distributed and a fundamental understanding of distributed algorithmics has never been more important. This shift towards distributed systems has resulted in increased interest and fast acceleration in our theoretical understanding of distributed optimization problems. At the same time, extremely general lower bounds uncovered that any distributed optimization requires $\tilde{\Omega}(\sqrt{n})$ rounds on worst-case topologies, even if the diameter of the diameter of the network is small. Many fundamental optimization problems, including MST, shortest paths, and cut/flow problems, now have “optimal” algorithms matching this worst-case performance bound.

Real world networks, however, are never worst-case and no network of interest shares the limiting bottleneck characteristics of the lower bound topology. In fact, there is no known barrier for ultra-fast polylogarithmic round distributed algorithms on any network of interest. This exponential gap between worst-case-optimal $\tilde{\Theta}(\sqrt{n})$ algorithms and the $O(\log^c n)$ performance which is likely possible in many, if not all, interesting small-diameter networks motivates this proposal and shows clearly that further studies going beyond worst-case topologies are necessary.

Our prior work shows that $o(\sqrt{n})$ round algorithms are possible for several important problems such as the MST and min-cut when the network is planar, genus-bounded, treewidth-bounded or, more generally, has excluded minors. We first ask the following ambitious questions: are the techniques used to achieve the above results sufficient to construct uniform algorithms that optimal across each possible instance, i.e., are instance-optimal?

Barring the question of instance-optimality, another direction we propose is adapting our techniques to novel problems for which no $o(\sqrt{n})$ algorithms currently exist (such as the shortest path problem), or for alternative models (e.g., faulty models or $O(n)$-message complexity in KT1).
1 Introduction

We propose to develop an algorithmic toolbox for distributed optimization algorithms that go beyond worst-case topologies. Such algorithms potentially have exponentially better performance guarantees on topologies of interest than state-of-the-art algorithms which aim to provide performance guarantees that match lower bounds for worst-case topologies.

Decentralization and distributed computing are becoming the computing paradigms of the future. In fact, modern computation and information processing systems are already massively distributed for various reasons: Moore’s law is approaching the limits of physics, data to be processed is vastly exceeding what can be stored on a single machine, and modern services are increasingly decentralized. A deep and fundamental understanding of distributed algorithmics is a prerequisite to build the efficient and reliable distributed systems of tomorrow and has never been more important than today.

The CONGEST model \cite{Pei00} has been the standard mathematical model to study distributed algorithmics. In this model the network topology is abstracted as a graph with \( n \) nodes and diameter \( D \). Communication occurs in synchronous rounds in which each node can send a bounded amount of information to each neighbor, typically \( O(\log n) \) bits. The complexity measure is the number of rounds required to solve an optimization problem. While this classic model is almost two decades old, it remains highly relevant and influential because it allows for a clean mathematical study of communication bottlenecks in distributed optimization algorithms. Indeed, communication remains the most crucial bottleneck in any practical distributed system. Consequently, many algorithms for massively parallel processing frameworks, like Hadoop \cite{Whi12} and Spark \cite{ZCFKN+10}, are influenced by algorithms designed for the CONGEST model.

This relevance has motivated a recent, broad, concentrated, and highly successful effort to advance our understanding of distributed algorithmics with a strong focus on algorithms for fundamental network optimization problems, such as MST \cite{KP08,Elk17a}, shortest paths \cite{Nan14,FW12,HW12,LP13,LPS15,IL13,HKN16,Elk17a,HNS17}, Flows \cite{GKKJ15}, cuts \cite{NS14,GK13}, etc.. As a result of these efforts, many fundamental optimization problems now have (near) optimal distributed algorithms with provable worst-case performance guarantees that match unconditional lower bounds for general topologies.

1.1 When Optimal Is Not Good Enough

While matching upper and lower bounds for distributed algorithmic problems are a remarkable achievement this proposal emphatically argues that one cannot settle for such results, particularly when the provable performance guarantees are too weak to be practically relevant. The justification of optimality simply does not match any real barriers that are observed in practice. Understanding this claim and its importance, especially in the case of distributed optimization algorithms, requires a more detailed look at different notions of optimality.

A key abstraction in the theoretical study of algorithms is the focus on worst-case running times expressed as asymptotic functions of the instance size, e.g., the number of nodes \( n \) for graph problems. This perspective successfully abstracts away differences between concrete computational models/implementations, enables definitions of complexity classes like P and NP,
and provides a way to compare the efficiency and scalability of different algorithms. The theory of distributed computing has historically followed this paradigm. The running time of a distributed algorithm is measured against all possible inputs and all possible network topologies. For example, one of the first distributed MST algorithms \[\text{[GIS83]}\] runs in \(O(n \log n)\) rounds. This was later improved to an “optimal” \(O(n)\) round algorithm by Awerbuch \[\text{[Awe87]}\].

The precise way in which the algorithm of \[\text{[Awe87]}\] is optimal is the following: There exists a (pathological worst-case) topology on \(n\) nodes on which no algorithm can do better, namely the \(n\)-node cycle. Indeed, it is easy to see that for most network optimization problems nodes require some knowledge about inputs that are far away from them in order to determine a solution. This leads to a trivial \(\Omega(D)\) lower bound on topologies with diameter \(D\), which is \(\Omega(n)\) in the pathological cycle network. An algorithm for which the asymptotic worst-case running time, over all inputs and networks of size \(n\), matches the worst-case running time of the best possible algorithm is called optimal or more precisely existentially optimal with respect to \(n\).

Existential optimality, however, says nothing about how the performance of an algorithm compares to what is achievable on non-worst-case topologies. Topologies of practical interest in particular might allow for drastically faster running times compared to a pathological worst-case instance. Indeed, essentially all real-world topologies have diameters which are (poly-)logarithmic in the size of the network and the trivial \(\Omega(D)\) lower bound for global optimization algorithms on such networks is merely \(\Omega(\log^c n)\). This is exponentially faster than the algorithm of Awerbuch, which requires \(\Theta(n)\) rounds on any topology.

Researchers soon realized that existentially optimal algorithms in terms of \(n\) are far from satisfactory and that a finer grained way to analyze and specify distributed running times was required. Kutten and Peleg \[\text{[KP95]}\] made progress in this direction by giving their celebrated \(\tilde{O}(D + \sqrt{n})\) round MST algorithm, achieving a quadratic performance improvement on any topology of interest. Later, a series of works that started with Peleg and Rubinovich \[\text{[PR99]}\] and culminated in the work of Das Sarma et al. \[\text{[DSHK+11]}\] gave strong unconditional lower bounds using communication complexity. They constructed a certain \(n\)-node networks with logarithmic diameter in which solving any non-trivial global optimization problem requires \(\tilde{\Omega}(\sqrt{n})\) rounds. This result makes the Kutten-Peleg MST algorithm existentially optimal with respect to \(n\) and \(D\) because for any \(n\) and \(D\) there exists a pathological worst-case network on which no algorithm can run faster. It is this notion of existential optimality with respect to \(n\) and \(D\) which applies to most state-of-the-art distributed optimization algorithms. I.e., on any input and low-diameter topology these algorithms achieve a \(\tilde{O}(\sqrt{n})\) running time, which is the best possible—at least on the pathological worst-case network of \[\text{[DSHK+11]}\].

Unfortunately, this form of optimality and the \(\tilde{O}(D + \sqrt{n})\) performance guarantees of these algorithms suffer from the exact same drawbacks as before: Real-world networks are never worst-case and no network of interest comes close to exhibiting any of the limiting bottleneck characteristics of the pathological worst-case topology which is used to demonstrate (existential) optimality. In fact, there is no plausible barrier known for ultra-fast polylogarithmic round CONGEST algorithms on any network of interest for most fundamental network optimization problems. On the other hand, essentially all optimal state-of-the-art \(\tilde{O}(\sqrt{n} + D)\) algorithms are

\[\text{1Throughout this proposal we mostly ignore polylogarithmic factors in } n. We use } \tilde{\Omega} \text{ and } \tilde{O} \text{ notation to hide such factors. For example, } \tilde{O}(f(n)) = O(f(n) \log^{O(1)} n).\]
specifically designed to achieve a $\Theta(\sqrt{n})$ running time and always require $\Theta(\sqrt{n})$ many rounds, even when no communication bottlenecks are present and faster algorithms are possible. This exponential gap between optimal worst-case algorithms which exhibit $\tilde{\Theta}(\sqrt{n})$ round complexities on every topology and the $O(\log^c n)$ performance which is likely possible in most, if not all, real-world settings forms the starting point for the studies of this proposal.

1.2 Overreaching goals

Given the clear need for efficient distributed algorithms that go beyond worst-case topologies, the overarching goal of this proposal can be summarized as follows.

**Objective 1.** Develop general tools for designing distributed optimization algorithms that provably achieve ultra-fast polylogarithmic running times on practical networks.

The key ingredient of our past work on this problem has been the *low-congestion shortcut framework*. This framework was originally devised to show that the $\tilde{\Omega}(\sqrt{n})$ lower bound of [DSHK+11] can be circumvented on planar networks. However, it has since shown great promise as a unified description of the common communication bottlenecks underlying most distributed optimization problems. Indeed, it is an intriguing question of whether one can generalize this technique to provide an instance-optimal distributed algorithm for some interesting optimization problem. An *instance-optimal* algorithm is an algorithm competitive with the best possible algorithm on any given topology [GKP93]. As such, it constitute the strongest possible form of algorithmically adjusting to non-worst-case topologies.

**Objective 2.** Study the possibility of instance-optimal distributed algorithms. Specifically, design an instance-optimal algorithm for a natural distributed optimization problem, such as MST, via the low-congestion shortcut framework.

Section 2 provides a detailed three-step program toward achieving Objective 2. This section also presents several open questions related to instance optimality that are interesting in their own right.

Low congestion shortcuts provide clean and general abstractions for a wide variety of distributed optimization problems, including MST, min-cut [GH16b], and shortest path problems [HL18]. The second part of this project, described in Section 3, aims at further developing the use of these abstractions as a general framework:

**Objective 3.** Further generalize and expand the low-congestion shortcut framework by incorporating powerful general-purpose algorithmic tools, such as continuous optimization and graph separators and applying low congestion shortcuts to new optimization problems, complexity measures, and distributed models.

We aim to improve our understanding of what structure and features in a network topology facilitate distributed algorithms or form communication bottlenecks for them. This objective remains relevant even once the holy-grail of designing instance optimal algorithms is achieved. After all, even if one has an algorithm that is provably as fast as it can possibly be, it is useful
to understand how fast one should expect this algorithm to be on a given topology and what
topologies, in general, admit fast distributed algorithms.

We believe that even a failure to achieve instance-optimality would lead to a deeper under-
standing of distributed optimization. For instance, a counter-example that showcases how our
low-congestion shortcuts fail to achieve instance optimality will lead to new insights on how
specialized algorithms can conform to networks. This understanding will go beyond worst-case
topologies and provide a useful toolbox for designing practically-relevant algorithms.

1.3 The Low-Congestion Shortcut Framework

Next, we briefly provide some background and details on the low-congestion shortcuts framework.
Broadly speaking, the shortcut framework shows how many distributed problems can be reduced
to solving a simple and natural communication problem. It also defines low-congestion shortcuts
as natural routing structures which can be used to algorithmically solve this communication
problem. Lastly, it shows how the topological structure of a network influences the quality of
shortcuts in the network and how this quality is directly coupled to the efficiency of shortcut-
based distributed algorithms.

More formally, consider the following part-wise aggregation problem: Given disjoint, con-
nected subsets of vertices, called parts, and input values at each vertex, compute within each
part in parallel a simple aggregate function. For instance, each vertex may want to compute the
minimum value among vertices in its part.

This problem arises naturally in many divide-and-conquer algorithms in which a network is
sub-divided and simple distributed computations need to be solved in each part. A prominent
example is Boruvka’s MST algorithm [NMN01] in which the MST is constructed across $O(\log n)$
iterations in which each connected component of edges selected so far computes and selects the
smallest weight edge leaving it. Up to a factor of $O(\log n)$ in the running time the classic MST
problem therefore reduces to solving the part-wise aggregation problem. Moreover, all known
MST distributed algorithms can be seen as simply finding efficient ways to solve the part-wise
aggregation problem. Maybe more surprisingly, recent related work has shown that many other,
seemingly unrelated, distributed network optimization problems like finding approximate min-
cuts [GH16b] or shortest paths [HL18] similarly reduce to solving the part-wise aggregation
problem.

Ideally one would like to solve this part-wise aggregation problem in $D$ time, where $D$ is
the diameter of the network. This would lead to essentially optimal $\tilde{O}(D)$ round optimization
algorithms. Unfortunately, the lower bound of [DSHK+11] shows that this is not possible, at
least not on the pathological worst-case network they construct. The reason for this, however,
is somewhat subtle. In particular, since parts are disjoint and connected one can easily solve
the aggregation problem by having nodes repeatedly forward the minimum value seen so far to
neighbors in their part. It is easy to see that the number of rounds needed for this trivial flooding
strategy to converge is equal to the strong diameter of any part. Unfortunately, the strong
diameter of a connected subgraph can be much larger than the diameter $D$ of the underlying
network graph. This demonstrates that efficient part-wise aggregation solutions require parts
to utilize edges not spanned by them to shortcut their communications. However, if too many
parts try to communicate using the same network edge they cause congestion which slows down communication. Overall, solving part-wise aggregation efficiently requires communicating over short paths whose edges have low congestion.

With this in mind, a $d$-dilation $c$-congestion shortcut for a set of parts is naturally defined as set of shortcut edges for every part which if added make the diameter of a part at most $d$ while each edge is used by at most $c$ parts. Classic routing results by Leighton, Maggs and Rao \cite{LMR94} show that such a shortcut allows for the part-wise aggregation to be solved in $\tilde{O}(c + d)$ rounds, even distributedly. We therefore call $Q = c + d$ the quality of a shortcut and 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. Putting all this together leads to the following theorem:

**Theorem 4.** Suppose a network $G$ admits low-congestion shortcuts of quality $Q$ and that these shortcuts can be efficiently computed. Then there exists $\tilde{O}(Q)$ time distributed algorithms on $G$ for the MST, min-cut, and shortest paths problems.

While providing clean mathematical models that capture the structure of real-world networks is a notoriously hard and imprecise endeavor, we believe that any practical network topology admits shortcuts of optimal quality $Q = \tilde{O}(D)$. If such shortcuts could furthermore be efficiently constructed than this would directly imply instance optimal $\tilde{O}(D)$ distributed algorithms for any network of interest, which is $O(\log^c n)$ for typical polylogarithmic network diameters.

Our related work, made significant progress towards this vision. In particular, large natural classes of network topologies including planar networks \cite{GH16b} and networks with bounded genus, pathwidth or treewidth were shown to admit optimal $\tilde{O}(D)$-quality shortcuts \cite{HIZ16b}. We have also shown that all these shortcuts, and in fact any quality-$Q$ shortcut satisfying a simple additional property of being “tree-restricted”, can furthermore be deterministically constructed in a distributed manner using only $\tilde{O}(Q)$ rounds \cite{HIZ16a}.

In this way low congestion shortcuts give the first widely applicable framework for designing distributed network optimization algorithms that go beyond worst-case topologies and drastically outperform state-of-the-art distributed algorithms on many network topologies of interest.

## 2 Towards Instance Optimality

This section gives a detailed description of our three-step program to achieve the instance-optimal algorithm of Objective 2. For the sake of concreteness and simplicity we focus on the MST problem. Due to the generality of the shortcut framework, ideas described here would likely transfer relatively easily to a wide variety of other fundamental optimization problems.

We start by discussing our formal definition of an instance-optimal MST algorithm in Section 2.1. Step one of our program is given in Section 2.2. The step aims to show that the MST problem and the problem of part-wise aggregation are reducible to one another up to a negligible poly-logarithmic overhead. The second step, described in Section 2.3, consists of developing a proof that every part-wise aggregation problem can be optimally solved using the simple shortcut based routing algorithm. The final step, elaborated on in Section 2.4, consists of giving an efficient distributed algorithm for constructing approximately optimal quality shortcuts.
Success in all three steps would indeed yield an instance-optimal distributed MST algorithm for any network topology $G$: The existence of an $O(T)$ round MST algorithm on $G$ would, by step one, imply an $O(T)$ round part-wise aggregation algorithm on $G$. By step two, this aggregation algorithm can be assumed to simply route along a quality $O(T)$ shortcut in $G$. The shortcut construction algorithm from step three would then be able to find such a $O(T)$-quality shortcut in $G$ in $O(T)$ rounds. Together with Theorem this would imply that the natural shortcut-based distributed MST algorithm runs in $O(T)$ rounds on $G$.

2.1 Formal Definition of Instance Optimality

It turns out that defining a plausibly tractable notion of an instance optimal MST algorithm is non-trivial. For this reason, we give our formal definition here.

Suppose that a distributed problem instance consists of a network topology $G$, a (distributed) input $I$, and a (distributed) output $O$. We demand that all nodes “terminate” simultaneously. This is without loss of generality and can be enforced for every algorithm with only an additive $O(D)$ round overhead. Now let $T_A(G, I)$ be the number of rounds until $A$ terminates on $G$ with input $I$. We say that $A$ is an instance optimal algorithm for this problem if (i) for any $I, G$ it produces a correct output $O$, and (ii) if there exists a constant $C \geq 0$ such that for all topologies $G$ and algorithms $A'$ $G$ that produce a correct output $O$ for any $I$ when run on $G$ we have that $\max_I T_A(G, I) \leq \max_I T_{A'}(G, I) \cdot \log^C |G|$. That is, for any given topology $G$, which is unknown to the nodes running $A$, the worst-case running time over all inputs is (up to a polylogarithmic factor) the same as the worst-case running of any algorithm $A'$ which is allowed to non-uniformly depend on $G$. The latter setting is equivalent to all nodes running $A'$ knowing the complete topology $G$ but not the complete input $I$.

In our definition of the MST problem the input consists of a distinct integer $O(\log n)$-bit weight for each of the edges in $G$ which is initially known only to the nodes incident to it and the output consists of every node stating (i) the weight of the unique MST $T^*$ and (ii) which of its adjacent edges are in $T^*$. For the part-wise aggregation problem, the input consists of a $O(\log n)$-bit value $x_v$ and part ID $p_v$ for each node $v$, where part IDs are distinct for every part. The output of a node is the minimum $x$-value among all nodes in its part.

To illustrate the subtleties of giving a reasonable definition of instance optimality, consider the following slight modification to our definition. Suppose that in MST we only demand that nodes know their incident edges in the MST $T^*$. That is, we drop the requirement that nodes have to output the weight of $T^*$. Such a modification precludes an instance optimal algorithm. In particular, on any tree network topology $G$ the trivial algorithm $A'$ which terminates immediately and simply has every node declare all of its edges to be part of the MST correctly reports $T^*$ on $G$ for any input $I$. Algorithm $A$, however, must run correctly on any topology which means that when it runs on $G$ it still has to determine that $G$ is tree. This necessarily requires at least $D$ rounds, where $D$ is the network diameter of $G$. Hence no instance-optimal MST algorithm $A$ can exist for this slightly different notion of instance optimality.
2.2 Interreducibility of MST and Part-wise Aggregation

The first step in our program is to prove that that the part-wise aggregation and the MST problem as defined above can be reduced to one another up to a logarithmic factors. This implication is well-known in one direction: Computing an MST can be reduced to logarithmically many part-wise aggregation computations. While non-trivial to prove formally, the opposite direction seems likely to hold since all known approaches to solving the MST distributedly can be seen to either implicitly or explicitly solve the part-wise aggregation problem on the way.

A natural approach for solving the part-wise aggregation problem using an MST algorithm as a sub-routine (which is the direction of the reduction needed), could look like this: Assign very cheap edge costs to edges between all nodes that are in the same part; select a (mostly independent) subset of all parts and assign to each edge leaving such a part $x$-value the node in the part incident to it; assign all other inter-part edges a very high weight; and compute the MST. The MST cut-property guarantees that for each selected part the MST contains the edge corresponding to the minimum $x$-value among all nodes in a part which have a neighbor outside the part. One can furthermore ensure that $x$-values of nodes that only have neighbors in their own part get taken into account as well by first flooding the minimum $x$-value observed so far within a part for $O(D)$ rounds. We have been playing with instantiations of these ideas for which hopefully only polylogarithmically many such flooding and MST computations are needed. This would lead to the desired $\Theta(D + MST(G)) \cdot \log c n = \tilde{\Theta}(MST(G))$ running time for the part-wise aggregation problem, where $MST(G)$ is the running time of the MST algorithm used in the reduction.

2.3 Part-wise Aggregation Implies High-Quality Shortcuts

The second step shows that shortcuts are always the optimal solution to any part-wise aggregation problem.

Objective 5. For any topology $G$ and any set of disjoint connected parts prove that if there exists an algorithm which solves the partwise aggregation algorithm in $T$ rounds for any input then there also exists a $\tilde{O}(T)$-quality shortcut for these parts in $G$.

If one treats the nodes’ inputs to the partwise aggregation problem as atomic packets that can only be routed through the network, the above objective is immediate because tracing packets in a valid partwise aggregation communication induces the desired low-congestion shortcuts. Due to standard (LP) rounding techniques the same still holds if one treats the information in the inputs as a “liquids” which can be sent fractionally along different routes as long as an edge does not exceed its $O(\log n)$ bit capacity per round. Both approaches, however, exclude the possibility of the communication algorithm employing network coding wherein different packets are coded together within the network. The extend to which network coding can improve communications compared to the more intuitive routing approaches is called the network coding gap. Indeed, Objective 5 essentially asks to prove a polylogarithmic upper bound on the network coding gap for the completion time of a multi-session multicast.

Coding gaps for similar communication problems have been extensively studied both in computer science and coding and information theory, Also, while surprising, coding gaps can
easily be polynomially large \cite{LL04,ACLY00,BKL11} even for simple and natural network communication problems. In the context of distributed optimization algorithms the power of even simple coding approaches can be seen in the MST algorithm of King et al. \cite{KKT15} which achieves a $\tilde{O}(n)$ message complexity by cleverly communicating XORs of node-IDs. On dense networks this quadratically beats the $\Omega(m)$ message complexity lower bound of Awerbuch et al. \cite{AGVP90}, which albeit only holds under the assumption that no coding is used.

Despite these cautionary tales we are confident that Objective 5 is true and provable. Indeed, we have unpublished upper and lower bounds which show that the coding gap for the completion time of a multi-session unicast is $\log^{O(1)} n$. This is a slightly simpler special case of Objective 5. Our results is proven via an interesting combination of an integrality gaps for a depth-bounded variant of the concurrent flow LP and a moving-cut argument similar to the one employed by \cite{DSHK+11}, which is surprising and novel in this context. We are currently working on extending this result to multi-casts, which would also imply Objective 5.

### 2.4 Constructing Shortcuts Instance-Optimally

The third and last step towards instance optimality is to design an efficient distributed algorithm to compute near-optimal shortcuts.

**Objective 6.** Give a distributed algorithm which for any network $G$ and any disjoint connected parts in $G$ that are given to the algorithm as an input computes a shortcut of quality $\tilde{O}(Q)$ in $\tilde{O}(Q)$ rounds, where $Q$ is the best shortcut quality admitted by $G$.

We are currently investigating different ideas for solving Objective 6, such as, taking centralized algorithms for the problem as an inspiration. In particular, one can obtain a centralized $O(\log^2 n)$-approximation algorithm for shortcuts by boosting any approximation algorithm for shallow-light trees using the multiplicative weights framework. It seems true that one can obtain sufficiently good distributed algorithms for shallow-light trees, which is an interesting result in its own right. Adapting the multiplicative weight paradigm to work sufficiently fast in the distributed setting seems challenging however, due to the rather sequential nature of the method. We are currently investigating different ideas for accomplishing the above objective, such as, taking centralized algorithms for the problem as an inspiration. In particular, one can obtain a centralized $O(\log^2 n)$-approximation algorithm for shortcuts by boosting any approximation algorithm for shallow-light trees using the multiplicative weights framework. It seems true that one can obtain sufficiently good distributed algorithms for shallow-light trees, which is an interesting result in its own right. Adapting the multiplicative weight paradigm to work sufficiently fast in the distributed setting seems challenging however, due to the rather sequential nature of the method.

Lastly, we remark that initially results like Objective 6 may look circular and too good to be true because the efficiency of the construction algorithm is supposed to be related to the quality of the near-optimal shortcut it is supposed to find: How can the algorithm profit from merely the existence of the unknown high-quality shortcut it is supposed to find? This perceived barrier, however, has already been overcome by the recursive approach of our shortcut construction algorithm in \cite{HIZ16a}. Indeed this algorithm solves the exact same problem as Objective 6 albeit for the case of tree-restricted shortcuts, i.e., it finds an approximately optimal tree-restricted
shortcut in time that is linear in the quality of this shortcut. This algorithm does not solve Objective 6 directly because there are topologies where the quality of the best tree-restricted shortcut is not comparable to the best unrestricted shortcut. Nonetheless, several of the ideas and key insights from [HIZ16a] seem to carry over and seem useful for proving Objective 6.

3 Generalizing the Shortcuts Framework

3.1 Incorporating Modern Optimization Techniques into Shortcuts

Many of the algorithmic tools that have had a tremendous impact in the centralized setting do not have counter-parts in the distributed world. We want to investigate how to incorporate such general-purpose tools into the low-congestion framework to increase its applicability even further. Since such a goal is quite open-ended, we choose two exemplary paradigms that have shown recent promise in the distributed setting.

**Continuous Optimization.** In the sequential world, continuous optimization techniques have recently led to remarkable breakthroughs for classical optimization problems like maximum flows and negative weighted shortest path, which have traditionally been approached with purely combinatorial methods. We believe that such modern optimization techniques will be equally transformative in the area of distributed algorithms, a currently still very combinatorial area. One noteworthy example which points in this direction is the beautiful work of Becker et al. [BKKL16] which uses a preconditioned gradient descent algorithm to solve the $1 + \varepsilon$ approximate shortest path problem in $\tilde{O}(\varepsilon^{-O(1)}(D + \sqrt{n}))$ rounds in the distributed setting. We feel this is a great starting point for investigating how modern optimization techniques can be brought into the low-congestion framework.

**Objective 7.** Adapt the approximate shortest path algorithm of Becker et al. to the shortcuts framework. Obtain a $\tilde{O}(\varepsilon^{-O(1)}Q)$ round distributed $(1 + \varepsilon)$-approximate shortest algorithm for any network with an efficient quality-$Q$ shortcut construction.

**Separators.** Graph separators are fundamental tools for graph algorithms. Roughly speaking, a separator breaks the graph into two disjoint pieces of roughly the same size by removing only a small number of vertices or edges. Separators form the basis of many centralized divide-and-conquer algorithms for near-planar [Epp03] and excluded minor networks [AST90a, AST90b]. Separators are often applied recursively. The initial work low-congestion shortcut papers [GH16a, GH16b] already implied that separators could be efficiently computed, at least on planar graphs. However, it is not clear how useful such classic graph separators are in the distributed setting. The reason why it is hard to utilize separators is that the recursive application splits the graph into components of small size, but potentially uncontrollably large diameter. A boundary of size $O(\sqrt{n})$ is also not interesting, at least if one needs to essentially sequentially process this boundary. Overall, there is a need for better understanding how and what kind of separators are appropriate and helpful.

**Objective 8.** Investigate appropriate notions and distributed constructions of separators that are appropriate for the distributed setting and that can be used in conjunction with the low-congestion shortcuts framework.
A recent result of Ghaffari and Parter [GP17] is encouraging on this front. This work computes a DFS ordering in any planar graphs in $\tilde{O}(D)$ rounds by utilizing path separators. Various issues, including the increased diameter from repeated-application of separators, are side-stepped in this work by using shortcuts.

3.2 Applying Shortcuts to New Complexity Measures, Models, and Problems

New Complexity Measures. Past work in distributed algorithms has focused on the number of rounds required to solve a problem as the primary complexity measure for designing and comparing distributed algorithms. However, there are other natural complexity measures which have been considered and which seem worth studying. We want to think about some of these and investigate if they integrate well into the shortcut framework.

One such alternative complexity measures is the message complexity of an algorithm, i.e., the total number of messages sent. In an exciting recent series of works [Elk17b, PRS17], it was shown that there is a single MST algorithm which simultaneously achieves an existentially optimal round complexity of $\tilde{O}(\sqrt{n} + D)$ and a near linear message complexity of $\tilde{O}(m)$. This was recently generalized [HHW18] to show that an optimal round complexity of $\tilde{O}(Q)$ simultaneously linear message complexity is achievable for the shortcut based partwise aggregation algorithm and shortcut construction algorithm from [HIZ16a] as well. Given that every network has trivial quality-$Q = O(\sqrt{n} + D)$ shortcuts this implies that simultaneous and generalized beyond-worst-case running times of $\tilde{O}(Q) \leq \tilde{O}(\sqrt{n} + D)$ and a linear message complexity is achievable by any current and future problem that can be shown to reduces partwise aggregation, e.g., MST, min-cuts, approximate shortest paths, etc. Here again the power of the widely applicable framework is very apparent. Even if one is only interested in worst-case running times using the low-congestion shortcut framework gives an easy way to prove something about a large number of very different algorithmic problems by merely showing it for the very simple partwise aggregation problem.

Recent work of King et al. [KKT15] shows that if nodes initially also know the IDs of their neighbors. An MST can be solved in only $\tilde{O}(n)$ messages as well, albeit at the cost of a longer running time and [GP18] shows how to create achieve various time-message tradeoffs for message complexities between $O(n)$ and $O(m)$ for the MST problem. We believe that the same tradeoffs should also apply to shortcuts / partwise aggregation and therefore many other distributed problems, too. Whether a good round complexity can be simultaneously obtained with an optimal $O(n)$ message complexity and whether such a result can then extended to the shortcut framework are interesting questions we intend to investigate.

Objective 9. Determine if there exists a $\tilde{O}(Q)$-round algorithm with message complexity $\tilde{O}(n)$ for the partwise aggregation problem on any network that admits $\tilde{O}(Q)$-quality (tree-restricted) shortcuts.

New Models. We are also interested in applying shortcuts to alternative models of distributed computation.

Objective 10. Determine the extent to which the low-congestion framework can give efficient and beyond-worst-case algorithms in alternative models for distributed computations such as asynchronous or faulty models.
New Problems. Lastly, it is natural to further investigate which other fundamental optimization problems are amenable to the shortcut framework.

Objective 11. Determine which other fundamental distributed network optimization problems can be reduced to a small number of partwise aggregation computations. Determine if more classic network optimization problems—especially max flow—reduce to part-wise aggregation.

The classical maximum flow problem, which was recently shown to have a worst-case optimal $\tilde{O}(\sqrt{n} + D)$ round algorithm [GKK+15], is merely one natural candidate for this objective.

References


