Distributed Computing (2021) 34:79–90
https://doi.org/10.1007/s00446-020-00383-2

Low-Congestion shortcuts without embedding

Bernhard Haeupler[1] · **Taisuke Izumi**[2] · **Goran Zuzic**[1]

Received: 11 December 2019 / Accepted: 3 July 2020 / Published online: 24 July 2020

Abstract

Distributed optimization algorithms are frequently faced with solving sub-problems on disjoint connected parts of a network. Unfortunately, the diameter of these parts can be significantly larger than the diameter of the underlying network, leading to slow running times. This phenomenon can be seen as the broad underlying reason for the pervasive $\widetilde{\Omega}(\sqrt{n}+D)$ lower bounds that apply to most optimization problems in the CONGEST model. On the positive side, [Ghaffari and Hauepler; SODA'16] introduced low-congestion shortcuts as an elegant solution to circumvent this problem in certain topologies of interest. Particularly, they showed that there exist good shortcuts for any planar network and more generally any bounded genus network. This directly leads to fast $O(D\log^{O(1)} n)$ distributed algorithms for MST and Min-Cut approximation, given that one can efficiently construct these shortcuts in a distributed manner. Unfortunately, the shortcut construction of [Ghaffari and Hauepler; SODA'16] relies heavily on having access to a genus embedding of the network. Computing such an embedding distributedly, however, is a hard problem—even for planar networks. No distributed embedding algorithm for bounded genus graphs is in sight. In this work, we side-step this problem by defining tree-restricted shortcuts: a more structured and restricted form of shortcuts. We give a novel construction algorithm which efficiently finds such shortcuts that are, up to a logarithmic factor, as good as the best restricted shortcuts that exist for a given network. This new construction algorithm directly leads to an $O(D\log^{O(1)} n)$-round algorithm for solving optimization problems like MST for any topology for which good restricted shortcuts exist—without the need to compute any embedding. This greatly simplifies the existing planar algorithms and includes the first efficient algorithm for bounded genus graphs.

1 Introduction

1.1 Background and motivation

Consider the problem of finding the Minimum Spanning Tree (MST) on a distributed network with n independent processing nodes. The network is abstracted as a graph $G=(V, E_G)$ with n nodes and diameter D. The nodes communicate by synchronously passing $O(\log n)$-bit messages to each of its direct neighbors. The goal is to design algorithms (protocols) that minimize the number of synchronous message passing rounds before the nodes collaboratively solve the optimization problem.

The message-passing setting we just described is a model called CONGEST [25]. The MST problem can be solved in such a setting using $O(\sqrt{n}\log^* n + D)$ rounds of communication [18][1]. Moreover, and perhaps more surprisingly, this bound was shown to be the best possible (up to polylogarithmic factors). Specifically, there are graphs in which one cannot do any better than $\widetilde{\Omega}(\sqrt{n}+D)$ [1,3,26][2]. While clearly no algorithm can solve any global network optimization problem faster than $\Omega(D)$, the $\widetilde{\Omega}(\sqrt{n})$ factor is harder to discern. To make matters worse, the $\widetilde{\Omega}(\sqrt{n}+D)$ lower

This work was supported in part by KAKENHI No. 15H00852 and 16H02878 as well as NSF grants CCF-1527110, CCF-1618280, CCF-1814603, CCF-1910588, NSF CAREER award CCF-1750808, a Sloan Research Fellowship and the 2018 DFINITY fellowship.

✉ Goran Zuzic
gzuzic@cs.cmu.edu

Bernhard Haeupler
haeupler@cs.cmu.edu

Taisuke Izumi
t-izumi@nitech.ac.jp

[1] Carnegie Mellon University, Pittsburgh, PA, USA

[2] Nagoya Institute of Technology, Gokiso-cho, Showa-ku, Nagoya, Aichi, Japan

[1] The algorithm can be easily modified to run in $O(\sqrt{n}\log^* n + D)$ rounds of communication by growing components to size $\sqrt{n/\log^* n}$ in the first phase of the algorithm.

[2] Throughout this paper, $\widetilde{O}(\cdot)$, $\widetilde{\Theta}(\cdot)$ and $\widetilde{\Omega}(\cdot)$ hide polylogarithmic factors in n, the number of nodes in the network.

bound was shown to be far reaching. It applies to a multitude of important network optimization problems including MST, minimum-cut, weighted shortest-path, connectivity verification and so on [1]

While this bound precludes the existence of more efficient algorithms in the general case, it was not clear whether it holds for special families of graphs. This question is especially important because any real-world application on huge networks should exploit the special structure that the network provides. The mere existence of "hard" networks for which one cannot design any fast algorithm might not be a limiting factor.

In the first result that utilizes network topology to circumvent the lower bound, Haeupler and Ghaffari designed an $\widetilde{O}(D)$-round distributed MST algorithm for planar graphs [6]. Note that this algorithm offers a huge advantage over older results for planar graphs with small diameters.

They achieve this by introducing an elegant abstraction for designing distributed algorithms named **low-congestion shortcuts**. Their methods could in principle be used to achieve a similar result for genus-bounded graphs, but their presented algorithms have a major technical obstacle: they require a surface embedding of the planar/genus bounded graph to construct the low-congestion shortcuts. While computing a distributed embedding for planar graphs has a complex $\widetilde{O}(D)$-round solution [5], this remains an open problem for genus-bounded graphs [6].

This paper side-steps the issue by vastly simplifying the construction of low-congestion shortcuts. We define a more structured version of low-congestion shortcuts called **tree-restricted shortcuts** and propose a simple and general distributed algorithm for finding them. On many graphs of interest these shortcuts are as powerful as the general ones (see the discussion in Sect. 1.4 for a short comparison). Moreover, the algorithm is completely oblivious to any intricacies of the underlying topology and finds universally near-optimal tree-restricted shortcuts. As a simple consequence of our construction technique we get an $\widetilde{O}(gD)$-round algorithm for genus g graphs, a result that was not known before the conference version of this paper was published. We believe that this simplicity makes the algorithm usable even in practice.

1.2 A brief overview of low-congestion shortcuts

We now give a short introduction to the general low-congestion shortcut framework, as defined in [6]. Consider the following recurring scenario throughout many distributed optimization problems:

Definition 1 *(Part-wise aggregation)* Let $G = (V, E_G)$ be a graph. Given disjoint and internally-connected **parts** $P_1, P_2, \ldots, P_N \subseteq V$ we want to distributedly compute some simple part-wise aggregate (e.g., sum or max) of nodes' private values. Specifically, each node is initially given its part ID (or $\perp$ if none) and a private value x_v; at the end of the computation each node v belonging to some part P_i should know the aggregate value of $\{x_v \mid v \in P_i\}$.

A classical example for such a scenario is the 1926 algorithm of Boruvka [23] for computing the MST: We start with a trivial partition of singleton parts for each node. For $O(\log n)$ iterations each part computes the minimum-weighted outgoing edge, adds it to the MST, and merges with the other part incident to this edge.

A key concern in designing a distributed version of Boruvka's algorithm is finding good communication schemes that allow the nodes of some part to collaborate without interfering with other parts. While a natural solution would be to allow communication only inside the same part (which is feasible since the parts are internally connected), this could take a long time. The problem appears when the diameter of a part in isolation is much larger than the diameter D of the original graph G.

Low-congestion shortcuts [6] were introduced to overcome this issue: each part P_i is allowed to use a set of extra edges $H_i \subseteq E_G$ to more efficiently communicate with other nodes in the same part. More precisely, part P_i is permitted to use the edges $E_G[P_i] \cup H_i$ for communication, where $E_G[P_i]$ are edges with both endpoints in P_i.

We evaluate the quality of the shortcut with **congestion** and **dilation**. A shortcut has dilation d if the diameter of $E_G[P_i] \cup H_i$ is at most d for all parts; the congestion is c when each edge is assigned to at most c different parts. If one can efficiently construct shortcuts with congestion c and dilation d, we can solve problems such as MST and Min-Cut approximation in $\widetilde{O}(c + d)$ rounds [6]. In other words, designing many distributed algorithms can be reduced to constructing good-quality shortcuts.

Definition 2 Let $G = (V, E_G)$ be an undirected graph with vertices subdivided into **disjoint and connected** subsets $\mathcal{P} = (P_1, P_2, \ldots, P_N)$, $P_i \subseteq V$. In other words, $E_G[P_i]$ is connected and $P_i \cap P_j = \emptyset$ for $i \neq j$. The subsets P_i are called **parts**. We define a **shortcut** $\mathcal{H}$ as $(H_1, H_2, \ldots, H_N)$, $H_i \subseteq E_G$. A shortcut is characterized by the following parameters:

i) $\mathcal{H}$ has congestion c if each edge $e \in E_G$ is used in at most c different sets $E_G[P_i] \cup H_i$, i.e., $\forall e \in E_G : |\{i : e \in E_G[P_i] \cup H_i\}| \leq c$. Note that the sets $\{E_G[P_i]\}_{i=1}^{N}$ are disjoint.

ii) $\mathcal{H}$ has dilation d if for each $i \in [N]$ the diameter of $E_G[P_i] \cup H_i$ is at most d.

While the pervasive $\widetilde{\Omega}(\sqrt{n} + D)$ lower bound clearly implies we cannot find shortcuts with congestion+dilation $= \widetilde{O}(D)$ on general graphs, this might not be the case on specific families of graphs. For example, planar graphs always offer $\widetilde{O}(D)$ congestion and dilation shortcuts, thus bypassing the $\widetilde{\Omega}(\sqrt{n} + D)$ lower bound. Finally, we note that congestion and dilation are traditional parameters that are extensively used in routing (e.g., [20]).

1.3 Our contribution

Roughly speaking, there are two challenges in the design of shortcut-based algorithms. Let $\mathcal{G}$ be the target class of graphs we want to design distributed algorithms. The first challenge is to identify the (small) values of c and d such that $\mathcal{G}$ has shortcuts with congestion c and dilation d. This is purely a graph-theoretic problem. The second challenge is to convert the existential result proved by the first challenge to the constructive result, i.e., we must design a distributed algorithm constructing efficient shortcuts for that class. This is a distributed computing problem that might be distinctively harder than the former one. Indeed, while one can prove that bounded genus graphs have good shortcuts, the proof is not constructive because it requires access to an embedding [6]; this is the primary reason why fast algorithms for bounded genus graphs were not known. Even in the planar case, distributedly constructing such an embedding is known, but complicated.

A natural idea to simplify algorithm design would be to come up with a generic procedure which finds a congestion c and dilation d shortcut for the best (or approximately best) c and d. Such a result would automatically lift a purely existential result to a constructive one and that is the primary contribution of this paper.

We present a simple algorithm for constructing shortcuts that resolves the issues mentioned above. We introduce a more structured definition of shortcuts called **tree-restricted shortcuts** and give a constructive algorithm that finds the nearly optimal tree-restricuted shorcuts in any graph that contains them. While the new shortcut definition is a strict subset of the old definition, for many graphs of interest one does not lose any power due to this restriction.

The details of our contribution are summarized as follows:

- In Sect. 3, we introduce tree-restricted shortcuts, which can only use edges of some fixed spanning tree $T \subseteq G$. We substitute the classic dilation parameter with a new **block parameter** more appropriate for tree-restricted shortcuts due to their highly-structured nature: in particular, the new parameter is stronger in the sense that it implies an upper bound on the dilation. The block parameter (upper-)bounds the number of components of P_i, where two nodes are in different components if they cannot reach each other via H_i. In Sect. 3.3 we propose deterministic algorithms for broadcast, convergecast, and leader election (for all parts in parallel) utilizing tree-restricted shortcuts, which are simpler and faster compared with the general-case randomized algorithms shown in [6].
- In Sect. 4, we present a generic algorithm for constructing tree-restricted shortcuts. Let T be a spanning tree of G with depth at most D and assume there exists a tree-restricted shortcut on $T \subseteq G$ with congestion c and block parameter b. We describe an algorithm that constructs a tree-restricted shortcut with congestion $O(c \log N)$ and block parameter $O(b)$ in $O(D \log n \log N + bD \log N + bc \log N)$ CONGEST rounds.
- An important consequence of our algorithm is to provide the first distributed algorithm constructing a good shortcut for genus-g graphs, and by extension, fast MST and Min-Cut approximations. Fortunately, one can reinterpret known results to conclude that genus-g graphs exhibit tree-restricted shortcuts with congestion $O(gD \log D)$ and block parameter $O(\log D)$ with respect to an arbitrary tree T of depth $O(D)$ (e.g., BFS tree). In Sect. 3.4, we can obtain a distributed algorithm that constructs a tree-restricted shortcut with congestion $O(gD \log D \log N)$ and block parameter $O(\log D)$ for graphs with genus at most g. For bounded genus graphs (i.e. $g = O(1)$), the algorithms based on our shortcut construction achieves near-optimal time complexity (up to a polylogarithmic factor).

1.4 Subsequent work: a short survey

Significant progress has been made since the initial conference version of this paper was published [12]. Subsequent work has expanded on the utility of the framework by extending it to new graph classes, new problems, and provided better construction guarantees. We intend this section to serve as a short and convenient survey of the tree-restricted shortcut framework.

Tree-restricted shortcut quality and construction. We define the **quality** q of a T-restricted shortcut as $q = b \cdot \text{depth}(T) + c$. Quality combines the congestion and the block parameter into a single value that sufficiently describes the shortcut construction and routing performance.

Definition 3 A graph $G = (V, E_G)$ of diameter D **admits** tree-restricted shortcuts of quality q if for each spanning tree T of depth $O(D)$ and each set of disjoint and connected parts $(P_i \subseteq V)_{i=1}^{N}$ there exists a T-restricted shortcut of congestion c and block parameter b satisfying $b \cdot D + c \leq q$.

The following theorem asserts that tree-restricted shortcuts of quality q can be constructed and used in $\tilde{O}(q)$ CONGEST rounds. A particularly appealing property of tree-restricted shortcuts is that one does not need to know the optimal shortcut quality q_* upfront. This can often yield much better shortcuts than guaranteed by the theoretical bound, a property often desired in practical applications.

Theorem 1 (Theorem 1.2 of [11]) *Suppose that a graph $G = (V, E_G)$ admits tree-restricted shortcuts of quality q. Given any set of disjoint and connected parts $(P_i \subseteq V)_{i=1}^{N}$ there exists a (randomized) distributed algorithm that solves the part-wise aggregation problem in CONGEST using $\tilde{O}(q)$ rounds and sends at most $\tilde{O}(m)$ messages during its execution with high probability. Moreover, the algorithm does not need to know the value of q upfront.*

Note: we slightly reworded the main Theorem of [11]. While the paper typically assumes the algorithm knows the congestion c and block parameter b, one can circumvent this issue with a simple exponential parameter search. At iteration i one can try to tree-restricted construct shortcuts with parameters $b_i \leftarrow 2^i, c_i \leftarrow D \cdot 2^i$ and verify if the shortcuts were successfully constructed (which can be done in $\tilde{O}(b_i \cdot D + c_i)$ rounds and $\tilde{O}(m)$ messages). Assuming the graph admits tree-restricted shortcuts of quality q_*, the search will terminate in $\tilde{O}(q_*)$ rounds and will use $\tilde{O}(m)$ messages.

Graph families. Various graph families admit good-quality tree-restricted shortcuts. Table 1 lists the known results, all of which fall within the extensive family of excluded minor graphs. Reducing the shortcut quality of excluded minor graphs from $\widetilde{O}(D^2)$ to $\widetilde{O}(D)$ is an interesting open problem; the other results are tight up to polylogarithmic factors.

Applications. Numerous distributed optimization tasks can be simplified and optimized by utilizing the part-wise aggregation primitive as a black-box subroutine. Applications include the MST, approximate Min-Cut, and approximate single-source shortest path (SSSP) [6,11,14].

Corollary 1 *Suppose that a graph G admits tree-restricted shortcuts of quality q. One can compute an (exact) MST in $\tilde{O}(q)$ rounds and $\tilde{O}(m)$ messages with high probability.*

As a reminder, in the Min-Cut problem one is given a graph $G = (V, E_G)$ with integer weights $w : E_G \to [1, \text{poly}(n)]$ and needs to compute a set of edges $F \subseteq E_G$ that disconnect G into at least 2 components while minimizing the sum $\sum_{e \in F} w_e$. An α-approximation to Min-Cut finds a set of edges that disconnects the graph whose aggregate weight is at most a multiplicative α factor larger than the optimal value.

Corollary 2 *Suppose that a graph G admits tree-restricted shortcuts of quality q. One can compute an $(1+\varepsilon)$-approximate (weighted) Min-Cut in $\tilde{O}(q) \cdot \text{poly}(1/\varepsilon)$ rounds and $\tilde{O}(m) \cdot \text{poly}(1/\varepsilon)$ messages with high probability.*

In the Single-Source Shortest Path (SSSP), one is given a graph $G = (V, E_G)$ with integer weights $w : E_G \to [1, \text{poly}(n)]$ and a source $s \in V$, and needs to compute a spanning tree $T \subseteq E_G$ such that for each node u we have that $d_T(s, u) = d(s, t)$ where $d(u, v)$ is the distance between $u, v \in V$ in G with respect to the weight w, and $d_T(u, v)$ is their distance in the tree (with respect to w). An α-approximation to SSSP requires the tree to satisfy $d_T(u, v) \leq \alpha \cdot d(u, v)$ (note that the inequality $d_T(u, v) \geq d(u, v)$ is always satisfied).

Corollary 3 *Suppose that a graph $G = (V, E_G)$ admits tree-restricted shortcuts of quality q. Each edge $e \in E_G$ has a weight w_e, and let L be the weight-diameter of G. For any $\beta = (\log n)^{-\Omega(1)}$ one can compute an $L^{O(\log\log n)/\log(1/\beta)}$-approximate SSSP in $\tilde{O}(q/\beta)$ rounds and $\tilde{O}(m/\beta)$ messages with high probability.*

For instance, in the above corollary, setting $\beta = n^{-\varepsilon}$, $\beta = 2^{-\Theta(\sqrt{n})}$, and $\beta = \log^{-\Theta(1/\varepsilon)} n$ for a constant $\varepsilon > 0$ one obtains a $\log^{O(1)} n$, $2^{\sqrt{\log n}}$ and L^{ε} approximations to SSSP, respectively. [14]

General shortcuts vs. tree-restricted shortcuts. One can easily construct pathological graph examples that admit good-quality *general* shortcuts, but do not admit good-quality *tree-restricted* shortcuts. For example, one can take the lower bound graph of [1] which requires $\widetilde{\Omega}(\sqrt{n})$ rounds to solve MST and replace each edge with $\sqrt{n}$ parallel multi-edge copies. This immediately yields a $\widetilde{O}(D) = \widetilde{O}(1)$ MST solution via general shortcuts, whereas tree-restricted shortcuts are constrained by the original $\widetilde{\Omega}(\sqrt{n})$ lower bound. Moreover, general shortcuts allow faster algorithms for several important graph families. For example, expander graphs and Erdős-Rényi random graphs admit general shortcuts of dilation + congestion $= \widetilde{O}(1)$ for any set of parts; no such result is possible in the tree-restricted setting. However, it seems that distributed construction of general shortcuts is a burdensome task even in highly structured graphs. The best-known result for shortcut construction and part-wise aggregation in expander graphs has round complexity $2^{O(\sqrt{\log n})} = n^{o(1)}$, significantly worse than the best existential result [10].

1.5 Related work

The complexity-theoretic issues in the design of distributed graph algorithms for the CONGEST model have received

Table 1 Upper and lower bounds for tree-restricted shortcuts

Graph family	Tree-restricted shorcut parameters				Lower bound
		Block	Congestion	Quality	$\Omega(d+c)$
General	[6]	1 [3]	$O(\sqrt{n})$	$O(D+\sqrt{n})$	$\widetilde{\Omega}(D+\sqrt{n})$
Pathwidth k	[13]	$O(k)$	$O(k)$	$O(kD)$	$\Omega(kD)$
Treewidth k	[13]	$O(k)$	$O(k\log n)$	$O(kD+k\log n)$	$\Omega(kD)$
Genus g	[13]	$O(\sqrt{g})$	$O(\sqrt{g}D\log D)$	$O(\sqrt{g}D\log D)$	$\Omega(\frac{\sqrt{g}D}{\log g})$
Planar	[6]	$O(\log D)$	$O(D\log D)$	$O(D\log D)$	$\Omega(D\frac{\log D}{\log\log D})$
Minor-excluded	[15]	$O(D)$	$O(D\log n+\log^2 n)$	$\widetilde{O}(D^2)$	trivial $\Omega(D)$

[3]For general graphs, each part of size $|P_i|\geq\sqrt{n}$ is assigned the entire tree; giving them a block param. of 1 and congestion of at most $\sqrt{n}$. Smaller parts can be handled separately in $\tilde{O}(\sqrt{n})$ rounds by using intra-part edges

much attention in the last decade. Researchers have studied many problems in-depth: Minimum-Spanning Tree [9,18,19,26], Maximum flow [8], Minimum Cut [7,24], Shortest Paths and Diameter [4,16,17,21,22], and so on. Most of those problems have $\widetilde{\Theta}(\sqrt{n}+D)$-round upper and lower bounds for some sort of approximation guarantee [1,2,7,21,26]. The guarantee of exact results sometimes yield a nearly liner-time bound [4]. Note that almost all lower bounds above hold for graphs of small diameter (e.g., polylogarithmic in n). In such graphs we have that $\sqrt{n}\gg D$, making $\widetilde{O}(D)$ algorithms strictly better than those requiring $\widetilde{O}(D+\sqrt{n})$ rounds.

2 Preliminary: CONGEST model

We work in the classical CONGEST model [25]. In this setting, a network is given as a connected undirected graph $G=(V,E_G)$ with diameter D. Initially, nodes only know their immediate neighbors and they collaborate to compute some global function of the graph like the MST. Communication occurs in synchronous rounds; during a round, each node can send $O(\log n)$ bits to each of its neighbors. The nodes always correctly follow the protocol and never fail. The goal is to design protocols that minimize the resource of time - the number of rounds before the nodes compute the solution.

We now precisely formalize the notion of solving a problem in this model, e.g., how is the input and output given. While the formalization is specifically given for the MST, any other problem is completely analogous. All nodes synchronously wake up in the first round and start executing some given protocol. Every node initially only knows its immediate neighbors and the weight of each of its incident edges. After a specific number of rounds, all nodes must simultaneously output (i) the weight of the computed MST τ (ii) for each edge e incident to it, a 0/1 bit indicating if $e\in\tau$.

3 Tree-restricted shortcuts

In this section we define tree-restricted shortcuts: a restricted version of low-congestion (i.e., general) shortcuts that are (i) simpler to work with, (ii) often equally powerful as the general shortcuts, (iii) offer deterministic routing schemes and, most importantly, (iv) can be efficiently constructed on any graph that contains them. Following the definitions, we rephrase the relevant prior work in our new term, showcase an efficient deterministic routing scheme, and finally state our main result and applications.

3.1 Definition

Tree-restricted shortcuts are low-congestion shortcuts with the additional property that H_i is restricted to (the edges of) some spanning tree T. The running time of algorithms will depend on the depth of T, hence we will assume throughout the paper that T is some tree of depth $O(D)$ (e.g., a BFS tree); the user of the framework is otherwise free to choose any tree T.

Definition 4 Let $\mathcal{H}=(H_1,H_2,\ldots,H_N)$ be a (general) shortcut on the graph $G=(V,E_G)$ with respect to the parts $\mathcal{P}=(P_i)_{i=1}^N$. Given a rooted spanning tree $T=(V,E_T)\subseteq G$ we say that a shortcut $\mathcal{H}$ is *tree-restricted* or **T-restricted** if for each $i\in[N]$, $H_i\subseteq E_T$ i.e., every edge of H_i is a tree edge of T.

Congestion and dilation are still well-defined for tree-restricted shortcuts. However, it is more convenient to use an alternative **block parameter**, which in turn also bounds the dilation. The block parameter (upper-)bounds the number of components of P_i, where two nodes $u,v\in P_i$ are in different components if they cannot reach each other via H_i.

Definition 5 Let $\mathcal{H}=(H_1,H_2,\ldots,H_N)$ be a T-restricted shortcut on the graph $G=(V,E_G)$ with respect to the parts $\mathcal{P}=(P_i)_{i=1}^N$. Fix a part P_i and consider the connected components of the subgraph (V,H_i). If a component contains at

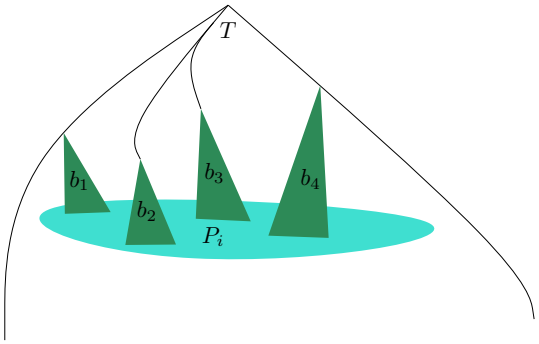

Fig. 1 Illustration of a T-restricted shortcut subgraph for a part P_i, composed of block components b_1, b_2, b_3 and b_4

least one node of P_i, we call it a **block component** (e.g., an isolated $v \in P_i$ is a block component). Furthermore, we say $\mathcal{H}$ has **block parameter** b if the number of block components associated with each part is at most b.

Note that a connected component of (V, H_i) without nodes in P_i does not need to be counted; it does not need any information from the part-wise aggregation of part i. On the other hand, an isolated vertex $\{v\}$ where $v \in P_i$ must be counted. Lemma 1 argues that a block parameter of b implies the dilation of $b(2 \cdot \text{depth}(T) + 1)$. From now on, we will assume that T is chosen to have depth $O(D)$, which is asymptotically minimal and achievable via a BFS tree. We note that distributedly computing a BFS tree is a classic problem with a simple $O(D)$ rounds CONGEST algorithm [25].

Lemma 1 *Let T be a spanning tree with depth at most D and let $\mathcal{H} = (H_i : i \in [N])$ be a T-restricted shortcut with congestion c and block parameter b with respect to parts $\mathcal{P} = (P_i : i \in [N])$. Then the dilation of $\mathcal{H}$ is at most $b(2D + 1)$.*

Proof Fix $i \in [N]$. Contract every block component of H_i into a supernode and remove all other nodes. This supergraph will contain $b' \leq b$ supernodes and will be connected (because $E_G[P_i]$ is connected). Hence its diameter is $b' - 1 \leq b - 1$. Every supernode corresponds to a block component of diameter $2D$, implying the diameter of $E_G[P_i] \cup H_i$ is at most $2bD + b - 1 < b(2D + 1)$. □

3.2 Shortcuts on genus-bounded and planar graphs

Tree-restricted shortcuts are particularly useful on genus-bounded (e.g., planar) graphs. In particular, we can reinterpret the low-congestion result of Haeupler and Ghaffari [6] using our notation.

Theorem 2 (Haeupler and Ghaffari [6]) *Let G be a graph with genus g and diameter D, and let T be any tree with depth $O(D)$ (e.g., a BFS tree). There exists a T-restricted shortcut with congestion $O(gD \log D)$ and block parameter $O(\log D)$.*

We note that the paper [6] proves the analogous claim about general shortcuts and does not explicitly talk about tree-restricted shortcuts. However, their proof implicitly argues precisely about the congestion and block parameter of tree-restricted shortcuts without explicitly referring to them. In particular, their $O(D \log D)$ dilation bound is implicitly derived by arguing about the block parameter being $O(\log D)$ and using Lemma 1. However, note that their theorem proves only the existence of such shortcuts. While the original paper does describe an algorithm that can in principle be used to compute them, it requires an embedding of G on a surface of genus g. It is an open problem to compute such an embedding efficiently in the CONGEST model.

3.3 Deterministic routing on tree-restricted shortcuts

In this section, we show how the structure of tree-restricted shortcuts can be useful in facilitating communication within parts. From a high level, the tree-like structure allows for fast, deterministic and simultaneous broadcasting/convergecasting on block components; this can be easily extended to true part-wise aggregation. For clarity, broadcast is defined as an operation on a rooted (sub)tree that floods some value from the root down to all other nodes; convergecast is defined as an aggregation of nodes' private values starting from the leaves and towards the root (ending in the root knowing the final aggregate). Lemma 2 gives a way how to simultaneously perform these primitives on subtrees.

Lemma 2 (Routing on subtrees) *Let T be a rooted tree of depth $O(D)$ and let $T_1, T_2, \ldots, T_k \subseteq T$ be a family of subtrees where each edge of T is contained in at most c subtrees, i.e., $|\{i \mid e \in T_i, i \in [k]\}| \leq c$. There is a simple deterministic algorithm that can perform a convergecast/broadcast on all of the subtrees in $O(D + c)$ CONGEST rounds.*

Proof We describe the convergecast algorithm. Each message sent during the algorithm will have a subtree-ID i associated with it. Suppose that a node v is in a subtree T_i (a node can be contained in multiple subtrees). We say (v, i) is active when v receives a message associated with i from all of its T_i-children (if v is a leaf in T_i, then (v, i) is immediately active). When (v, i) becomes active, it will schedule an ID-i message to be sent along its T-parent edge; note that two messages scheduled along the same edge cannot have the same ID. Each round, if multiple messages are scheduled over the same T-edge, the algorithm sends the message associated with the ID i that minimizes $\text{depth}_T(\text{root}(T_i))$. Here, $\text{depth}_T(v)$ is the length of the unique path between $\text{root}(T_i)$ and v in T. Ties are broken by the ID i itself. The converge-

cast and broadcasts operations are symmetric, so we will only prove the lemma for convergecasts.

We now analyze the algorithm. Fix a node v. It is sufficient to prove that no message gets transmitted along v's parent edge after $\text{height}_T(v) + c = O(D + c)$ rounds where $\text{height}_T(v)$ is the maximum distance between v and any leaf in T that is a descendant of v (the unique path between the T-root and the leaf goes through v).

Note that any message that gets transmitted along v's parent edge must belong to a subtree T_i that contains that edge. Let $I = (i_1, i_2, \ldots, i_k)$ be the IDs of subtrees that contain v's parent edge, ordered by their priority (as described). In particular, we say that T_{i_p} has priority p. The congestion condition stipulates that $k \leq c$.

We will prove by induction that for $p \in [k]$ the message associated with i_p will be transmitted no later than round $\text{height}_T(v) + p$. The claim clearly holds for the leafs of T. Note that (i) the relative priority-ordering between I is unchanged with respect to any node of T (other than v), (ii) any subtree T_i that is contained in the set of descendants of v, but does not contain the parent edge of v will have lower priority than any subtree in I.

Fix i_p. By the induction hypothesis, messages corresponding to $\{i_1, \ldots, i_{p-1}\}$ will be sent strictly before round $\text{height}_T(v) + p$. It is sufficient to argue that v has received messages corresponding to i_p from all of its T_{i_p}-children before round $\text{height}_T(v) + p$. However, this can be directly argued from the induction: for any child $w \in T_{i_p}$ we have $\text{height}_T(w) \leq \text{height}_T(v) - 1$, hence the priority of i_p is at most p with respect to w. Hence v will send the message corresponding to i_p no later than round $\text{height}_T(v) + p$ and we are done. □

Convergecast and broadcast are used to facilitate routing in tree-restricted shortcuts. We can intuitively envision the shortcut edges H_i as a family of subtrees (in our notation: block components). Aggregation of values within each block component can be exactly achieved by simultaneously convergecasting and broadcasting in all block components. We extend this result to true part-wise aggregation.

Theorem 3 (Routing on tree-restricted shortcuts) *Given a T-restricted shortcut with congestion c and block parameter b, there are deterministic distributed algorithms that terminate in $O(b(D + c))$ rounds for the following problems.*

1. *Electing a leader for each of the parts in parallel.*
2. *Convergecasting $O(\log n)$-bit messages to the leader of each part in parallel.*
3. *Broadcasting a $O(\log n)$-bit message from the leader of each part in parallel.*

Proof All of these algorithms have a common flavor: for each part we perceive its shortcut edges H_i as a supergraph of at most b supernodes where each supernode corresponds to a block component. We proceed to describe each of the algorithms on the supergraph and implicitly assume that intra-block communication happens after each step of the algorithm.

Communication within block components can be done in parallel using Lemma 2: all the nodes of a block component convergecast the relevant information to the block-root and subsequently the block-root broadcasts the result back.

Electing a leader for each part is performed by electing a leader for each supernode (block component) and broadcasting the leader to all neighborhood supernodes for b steps. Every supernode keeps the smallest leader ID ever seen as its current leader. After b rounds all the supernodes have the same leader. The algorithm requires $O(b(D + c))$ rounds as each of the b broadcasting steps is followed by an $O(D + c)$ intra-block communication step.

Broadcasting/convergecasting from/to the leader for each part can be done by building a BFS tree from the leader-supernode. We can utilize the standard distributed BFS algorithm on the supergraph requiring $O(b)$ steps. The algorithm similarly requires $O(b(D + c))$ rounds as each of the $O(b)$ BFS steps is followed by an $O(D + c)$-round intra-block communication step. □

We also state a simple technical lemma we use for the construction of tree-restricted shortcuts.

Lemma 3 *Given a T-restricted shortcut with congestion c, a deterministic distributed algorithm can identify all parts with at most b' block components. Specifically, after the algorithm terminates each node within a part i knows if P_i is composed of more than b' block components. The algorithm executes in $O(b'(D + c))$ rounds.*

Proof Similarly to the proof of Theorem 3, for each part P_i we consider the (connected) supergraph where each supernode corresponds to a block component of H_i. We need to find all parts whose supergraphs have at most b' supernodes.

Each supernode broadcasts its leader for exactly b' rounds and every supernode keeps the minimum ID as their current leader. Subsequently, each leader r (there may be multiple ones as we have not bounded the block parameter) tries to build a BFS tree comprised of all the nodes that believe r is the leader. We can detect the existence of multiple leaders as in that case each BFS tree will contain two neighboring supernodes in different BFS trees and report failure. If this is not the case (all the supernodes of a part belong to the same BFS tree), we can convergecast the number of supernodes back to the root and subsequently broadcast their count back. □

Comparison with routing on general shortcuts: Ghaffari and Haeupler [6] give a method for routing on general

shortcuts in $O(\text{dilation} \cdot \log n + \text{congestion})$ rounds that is randomized and assumes a leader is already elected for each part. They describe a process of leader election via a complicated randomized bootstrapping process that takes $O(\text{dilation} \cdot \log^2 n + \text{congestion} \cdot \log n)$ rounds. We contrast those results with our current tree-restricted shortcut routing where leader election is simple, deterministic and essentially no more difficult than a single convergecast+broadcast. The downside is that non-tree-restricted shortcuts sometimes offer better quality guarantees and therefore better performance.

3.4 Main result and applications

The main contribution of the paper is to introduce a general framework for finding near-optimal tree-restricted shortcuts in graphs where the only assurance is that they exist.

Theorem 4 *Let G be a graph with a spanning tree $T \subseteq G$ such that there exists a T-restricted shortcut with congestion c and block parameter b. There exists a distributed algorithm that finds a T-restricted shortcut with congestion $O(c \log N)$ and block parameter $3b$ with high probability (with probability at least $1 - n^{-O(1)}$, where any constant can be chosen in the exponent). The shortcut can be found in $O(D \log n \log N + bD \log N + bc \log N)$ rounds.*

We note that the Theorems 2 and 4 immediately give a novel result: an algorithm for constructing shortcuts on bounded genus graphs.

Corollary 4 *Given a genus-g graph with diameter D and N parts there is a (randomized) distributed algorithm that computes a tree-restricted shortcut with congestion $O(gD \log D \log N)$ and block parameter $O(\log D)$ in $O(gD \log^2 D \log N)$ rounds with high probability.*

Next, we explain how to use tree-restricted shortcuts to distributedly compute the MST on genus-g graphs. Similarly to [6], we incorporate the shortcuts into the classic 1926 algorithm of Boruvka [23].

Lemma 4 *Given a genus-g graph with n nodes and diameter D, there is a (randomized) distributed algorithm that computes the Minimum Spanning Tree in $O(gD \log^2 D \log^2 n)$ rounds with high probability.*

For completeness we give a brief proof outline:

Proof Boruvka's algorithm runs in $O(\log n)$ phases. Each phase starts with a partition of the graph into connected parts; each part has previously computed the MST on the subgraph induced by the part. Initially, the algorithm starts with the trivial partition in which each node is in its own part. During each phase, each part P_i suggests a merge along the minimum-weighted edge going out of P_i. It is well-known that all such edges belong to some MST. By computing a tree-restricted shortcut for each part in $O(gD \log^2 D \log n)$ rounds and using our convergecast algorithm on it in $O(gD \log^2 D)$ rounds we can compute the min-weight outgoing edge from each part. A slight difficulty remains: many parts could chain together to form a new part, making the assignment of part IDs in the newly merged part difficulty. This can be avoided by restricting the merge shapes to be star graphs: each part can independently mark itself as a **head** or **tail** with probability $\frac{1}{2}$; we are only allowed to merge tails to heads. The number of phases remains $O(\log n)$ as every minimum-weighted outgoing edge will be used for merging with probability at least $\frac{1}{4}$, thus reducing the expected number of parts by a constant.□

4 Constructing tree restricted shortcuts

In this section, we describe an algorithmic framework that solves the problem of finding near-optimal tree-restricted shortcuts.

4.1 Overview of the algorithmic framework

Our algorithm `FindShortcut` uses two separate subroutines:

- **Core:** This subroutine finds a good-quality shortcut with respect to at least a constant fraction of the parts. As a prerequisite, we assume we constructed a tree T with depth $O(D)$ such there exists a T-restricted shortcut with congestion c and block parameter b. Note that we only assume the tree-restricted shortcut's existence.

Lemma 5 *Let T be a spanning tree with depth $O(D)$ and assume there exists a T-restricted shortcut with congestion c and block parameter b. The subroutine `CoreFast` finds a T-restricted shortcut $\mathcal{H}' = (H'_i)_{i=1}^N$ with the following properties:*

1. *The congestion of $\mathcal{H}'$ is at most $8c$ with high probability.*
2. *There exists a subset of parts $\mathcal{P}' \subseteq \mathcal{P}$ with size at least $|\mathcal{P}'| \geq \frac{|\mathcal{P}|}{2}$ such that each part in $\mathcal{P}'$ has at most $3b$ block components.*

The subroutine takes $O(D \log n + c)$ CONGEST rounds to execute with high probability. Upon completion, each node knows for each of its incident edges which parts are they assigned to in $\mathcal{H}'$.

We present two versions of the core subroutine for purposes of exposition. We present a deterministic and simper `CoreSlow` requiring $O(D \cdot c)$ rounds and a randomized `CoreFast` requiring $O(D \log n + c)$ rounds. We note that

the CoreFast subroutine is the only randomized building block of our framework. Therefore, we can replace it with a deterministic (albeit slower) version at a cost of an addition $\frac{c}{\log n}$ factor.

- **Verification:** This subroutine is used to identify the parts i for which the shortcut edges H_i have a sufficiently small number of block components. The following result follows directly from Lemma 3.

Corollary 5 *Given a tree T with depth at most D and a tentative T-restricted shortcut $\mathcal{H}'$ with congestion c, the deterministic subroutine Verification finds all parts $\mathcal{P}' \subseteq \mathcal{P}$ whose designated shortcuts have at most b' block components. The subroutine takes $O(b'(D+c))$ CONGEST rounds to execute. Upon completion, each node knows whether its part is in the set $\mathcal{P}'$ or not.*

We use the subroutines in FindShortcut that implements the construction of Theorem 4.

Algorithm FindShortcut: We run the CoreFast subroutine that computes a shortcut $\mathcal{H}' = \{H'_1, \ldots, H'_N\}$ with congestion $8c$, but possibly an unacceptably large block parameter. The next step is to run the Verification subroutine that finds all parts whose computed shortcut edges H'_i have at most $3b$ block components. We call those parts **good** and fix their computed shortcut edges and discard the rest. The subroutine is iteratively repeated for $O(\log N)$ rounds at which point the parts have been marked as good.

Proof of Theorem 4 By Lemma 5, in each iteration we find a shortcut with congestion $8c$ and block parameter $3b$ for at least a half of the parts that have not yet been marked as good, w.h.p. This implies that after $O(\log N)$ iterations all the parts are marked as good. This further implies that the congestion of $\mathcal{H}'$ is $O(c \log N)$ as the congestion of the union of partial shortcuts is at most the sum of congestion of individual partial shortcuts.

Finally, the number of rounds is at most $O(\log N)$ times the combined number of rounds of the CoreFast and Verification subroutines, namely $O(\log N \cdot (D \log n + c + bD + bc)) = O(D \log N \log n + bD \log N + bc \log N)$ w.h.p. □

4.2 Warm-up: an $O(D \cdot c)$-round version of the core subroutine

In this section, we explain a simple and deterministic, but slower version of the core subroutine named CoreSlow that terminates in $O(D \cdot c)$ CONGEST rounds. We improve its round complexity to $O(D \log n + c)$ in the following section.

On a high level, the subroutine takes each part P_i and tries to assign the T-ancestors of nodes in P_i to its shortcut edges H'_i. However, this might lead to a large congestion on some edges. We address this issue by declaring an edge **unusable** if more than $2c$ different parts try to use it. This ensures the congestion is at most $2c$. We show the process provably leads to a constant fraction parts having small congestion and a small block parameter.

Preliminaries: As standard, assume we fix a spanning tree $T = (V, E_T)$ of depth $O(D)$ such that G has a T-restricted shortcut with congestion c and block parameter b. During the execution of the algorithm some of the edges will be marked as **unusable**. Furthermore, we say that a tree edge $e \in E_T$ **can see** a node $v \in V$ if v is in the subtree of e and no edge on the unique path between the lower endpoint of e and v is unusable. Analogously, an edge can see a part P_i if it can see any node in P_i. **Outline of the CoreSlowsubroutine:** Initially, no edge is unusable. We process the (tree) edges of T in order of decreasing depth (bottom to top). An edge e is assigned to all parts P_i that e can see. If an edge is assigned to more than $2c$ different part, we mark this edge e as **unusable** disallow e from being used at all by any part.

Detailed description of the CoreSlow subroutine: Each node v maintains a list L_v of part IDs that v's T-parent edge can see. The lists L_v are initially empty. The subroutine runs in depth(T) phases where in phase k each node v at depth depth$(T) - k$ updates L_v simultaneously and send the entire list L_v to its (v's) T-parent. Consider a node v that receives $L_{v'}$ for all its T-children v'. We assign the union of all received lists and the singleton part ID of v (if any) to L_v. If $|L_v| \leq 2c$, we assign the parent edge of v to all the parts in L_v and transmit L_v to its parent (potentially requiring $2c$ rounds). Otherwise, if $|L_v| > 2c$, we declare the parent edge as unusable. A direct implementation of this would lead to a subroutine that takes $O(D \cdot c)$ rounds in the CONGEST model. Each of the $O(D)$ levels of T must propagate at most $2c$ part IDs to their parent nodes. However, this bottleneck can be improved by random sampling, as we show in the next section with the subroutine CoreFast.

Algorithm 1 CoreSlow

1. At time k each node v at depth $depth(T) - k$ does the following in parallel:
   (a) if v is an element of P_i, set $L_v \leftarrow \{i\}$, otherwise $L_v \leftarrow \emptyset$
   (b) receive all the part IDs from v's children and assign their union to L'
   (c) $L_v \leftarrow L_v \cup L'$
   (d) if $|L_v| > 2c$, mark v's parent edge as unusable
   (e) otherwise (serially) send all the part IDs of L_v up to v's parent node
2. For each node v:
   (a) if the parent edge e of v is marked as unusable, e will not be assigned to any part
   (b) otherwise e will be assigned to all $H_i, \forall i \in L_v$

Lemma 6 *Let T be a spanning tree of depth $O(D)$ and assume there exists a T-restricted shortcut with congestion c and block parameter b. The subroutine* `CoreSlow` *finds a T-restricted shortcut $\mathcal{H}' = (H'_1, H'_2, \ldots, H'_N)$ with the following properties:*

1. *The congestion of $\mathcal{H}'$ is at most $2c$.*
2. *There exists a subset of parts $\mathcal{P}' \subseteq \mathcal{P}$ with size at least $|\mathcal{P}'| \geq \frac{|\mathcal{P}|}{2}$ such that each part in $\mathcal{P}'$ has at most $3b$ block components.*

The subroutine is deterministic and takes $O(D \cdot c)$ CONGEST rounds to execute. Upon completion, each node knows for each of its incident edges which parts are they assigned to in $\mathcal{H}'$.

Proof Let $\mathcal{H} = (H_i)$ be any T-restricted shortcut with congestion c and block parameter b and let $\mathcal{H}' = (H'_i)$ be the shortcut computed by `CoreSlow`. We call $\mathcal{H}$ the **canonical** shortcut and $\mathcal{H}'$ the **computed** shortcut.

By construction, the congestion of $\mathcal{H}'$ is $2c$ as any edge that would be assigned to more than $2c$ parts is marked as unusable. Hence we proved property 6.

Let $U \subseteq E_T$ be the set of unusable edges marked by the subroutine. In this paragraph we find an upper bound for $|U|$. Consider **blaming** a part P_i for congesting an unusable edge $e \in U$ when $e \notin E_G[P_i] \cup H_i$ and e can see P_i, i.e., edge e was not in the canonical shortcut H_i, but e was congested by part P_i (and ultimately declared unusable). Each part can be blamed at most b times because each block component can only be blamed for the first unusable edge in his T-tree path towards the T-root. Furthermore, if e is unusable, it takes at least $2c - c$ different block components (from different parts) to be blamed for congesting e. Therefore $|U| \leq N\frac{b}{c}$.

We say that a part P_i **missed** an edge e when $e \in E_G[P_i] \cup H_i$ and $e \in U$ (consequently, $e \notin H'_i$). Furthermore, call a part **bad** if it missed at least $2b$ edges and **good** otherwise. Note that if a part P_i is good, the block parameter of H'_i is at most $2b + \text{blockParameter}(\mathcal{H}) = 3b$. This is because each missed edge induces a new block component in $\mathcal{H}'$ (more precisely, we can identify each block component of $\mathcal{H}'$ with either an unique block component of $\mathcal{H}$ or an unique missed edge $e \in U$). Consequently, it is sufficient to prove that the subroutine finds at least $\frac{1}{2}N$ good parts.

As any unusable edge is assigned to at most c parts in the canonical shortcut, and for a part to be bad we need at least $2b$ edges to be missed, we have that the number of bad parts is at most $|U|\frac{c}{2b} \leq \frac{1}{2}N$. Hence, the subroutine finds at least $\frac{1}{2}N$ good shortcuts, proving property 6.

The subroutine terminates in $O(D \cdot c)$ rounds: on each of the $O(D)$ levels of the tree T all the nodes in parallel must send the part IDs trying to use its parent edge up the tree. A node can send up to $2c$ IDs, each requiring a round for its transmission. □

4.3 A faster $O(D \log n + c)$-round version of the core subroutine

In this section, we describe a faster version of the core subroutine named `CoreFast`. On a high level, we lower the running time of `CoreSlow` by estimating the number of parts trying to use an edge by random sampling. In particular, each part becomes **active** with probability p and we declare an edge unusable when $\Omega(c \cdot p)$ active parts try to use that edge.

Preliminaries: In addition to the preliminaries of `CoreSlow` we need shared randomness between all the nodes within a part. In other words, all the nodes of the same part must have access to the same seeds for a pseudorandom generator. This can be done by sharing $O(\log^2 n)$ random bits among all the nodes of G in $O(D + \log n)$ rounds, as described in [6].

Outline of the `CoreFast` subroutine: Each part becomes **active** with probability $p = \frac{\gamma \log n}{2c}$ where $\gamma > 0$ is sufficiently large constant. We basically follow the `CoreSlow` subroutine, but instead of propagating all $O(c)$ part IDs of L_v, we propagate only the active ones. An edge is declared **unusable** if at least $4c \cdot p = \Omega(\log n)$ (active) part IDs want to use it. Hence, by a standard Chernoff bound argument we can claim with high probability that (i) we never propagate more than $O(\log n)$ part IDs through an edge, (ii) each unusable edge has at least $2c$ part IDs trying to use that edge, and (iii) each usable (non-congested) edge has at most $8c$ part IDs. After determining which edges are unusable in $O(D \log n)$ rounds, `CoreFast` must nevertheless find the complete set of part IDs that can use each edge. This is a tree routing problem where each message (part ID) has to be routed up the tree T until the first unusable edge. No message needs to travel more than D edges and no edge needs to transmit more than $8c$ different part IDs w.h.p. Hence this routing can be done in $O(D + c)$ using Lemma 2.

Detailed description of the `CoreFast` subroutine: Due to shared randomness, each part independently becomes **active** with probability $p = \frac{\gamma \log n}{2c}$ (all the nodes within the part agree on this label). Similarly as in `CoreSlow`, each node v maintains a list $\tilde{L}_v$ of active part IDs that its (T) parent edge can see. The lists $\tilde{L}_v$ are initially empty. The subroutine runs in depth(T) phases where in phase k all the nodes at depth depth$(T) - k$ try to update $\tilde{L}_v$ in parallel and send $\tilde{L}_v$ to its T-parent. Consider a node v that receives $L_{v'}$ for all its T-children v'. We assign the union of all received lists and the singleton part ID of v (if any) to L_v. If $|L_v| \leq 4c \cdot p$, we assign the parent edge of v to all the parts in L_v and transmit L_v to its parent (potentially requiring $O(\log n)$ rounds). Otherwise, if $|L_v| > 4 \cdot p$, we declare the parent edge as

unusable. This finalizes the first part of the subroutine where we determine all unusable edges. It remains to forward the complete set of part IDs (and not just the sampled ones) that can use some edge e to the endpoints of e. This is a classic tree routing problem where no route has its length larger than D and no edge intersects more than $8c$ paths w.h.p. Lemma 2 provides a method to route all part IDs in at most $O(D+c)$ rounds. Note that any two part IDs whose routes share an edge have the same endpoint (lowest unusable ancestor edge), so any routing priority between the messages gives the aforementioned $O(D+c)$ bound w.h.p.

Algorithm 2 `CoreFast`

1. Each part becomes active with probability $p = \frac{\gamma \log n}{2c}$
2. At time k each node v at depth $depth(T) - k$ does the following in parallel:
   (a) if v is an element of P_i and P_i is active, set $\tilde{L}_v \leftarrow \{i\}$, otherwise $\tilde{L}_v \leftarrow \emptyset$
   (b) receive all the active part IDs from v's children and assign their union to L'
   (c) $\tilde{L}_v \leftarrow \tilde{L}_v \cup L'$
   (d) if $|\tilde{L}_v| \geq 4c \cdot p$, mark v's parent edge as unusable
   (e) otherwise send all the part IDs $\tilde{L}_v$ up to v's parent node
3. Each node v initializes Q_v with its part ID (or $\emptyset$ if not in any part)
4. Each node v does the following in parallel:
   (a) add all received IDs to the Q_v
   (b) if parent edge of v is not unusable and $\exists i \in Q_v$ that was never forwarded
      i. forward minimum such i along the parent edge
5. Each part ID in Q_v can use the parent edge of v, unless it is unusable

Lemma (Restated Lemma 5) *Let T be a spanning tree with depth $O(D)$ and assume there exists a T-restricted shortcut with congestion c and block parameter b. The subroutine `CoreFast` finds a T-restricted shortcut $\mathcal{H}' = (H'_i)_{i=1}^{N}$ with the following properties:*

1. *The congestion of $\mathcal{H}'$ is at most $8c$ with high probability.*
2. *There exists a subset of parts $\mathcal{P}' \subseteq \mathcal{P}$ with size at least $|\mathcal{P}'| \geq \frac{|\mathcal{P}|}{2}$ such that each part in $\mathcal{P}'$ has at most $3b$ block components.*

The subroutine takes $O(D \log n + c)$ CONGEST rounds to execute with high probability. Upon completion, each node knows for each of its incident edges which parts are they assigned to in $\mathcal{H}'$.

Proof This proof extensively utilizes methods used in the proof of Lemma 6. For completeness, we redefine all of the used terminologies and reprove all of the intermediate results.

Let $\mathcal{H} = (H_i)$ be any T-restricted shortcut with congestion c and block parameter b and let $\mathcal{H}' = (H'_i)$ be the shortcut computed by `CoreFast`. We call $\mathcal{H}$ the **canonical** shortcut and $\mathcal{H}'$ the **computed** shortcut.

Consider any tree edge. Suppose that the edge can see t different part IDs. Denote by $X_1, \ldots, X_t$ whether those t parts are active (in which case $X_i = 1$, otherwise $X_i = 0$). Let $S := X_1 + X_2 + \ldots + X_t$. Due to sampling, we have that the expectation $\mathbb{E}[S] = pt$. Since $X_i \in \{0, 1\}$ and they are independent we can apply a standard Chernoff bound argument giving us that $\Pr[X_1 + \ldots + X_t \leq \frac{1}{2}\mathbb{E}[S]] \leq \exp(-\delta\mathbb{E}[S])$ for some constant $\delta > 0$. Suppose now that $t \geq 8c$, we have that $\Pr[X_1 + \ldots + X_t \leq 4c \cdot p] \leq \exp(-\delta 8pc) = \exp(-\delta 4\gamma \log) = n^{-\gamma'}$ for a sufficiently large constant $\gamma' > 0$ (since we choose $\gamma > 0$ sufficiently large). We conclude that if $t \geq 8c$, the considered edge will become unusable with high probability. Since there are only a polynomial number of different edges, we can use a union bound to conclude that the congestion of $\mathcal{H}'$ is $8c$ (for all edges) with high probability (since the probability of this being violated is at most $n \cdot n^{-\gamma'} = n^{-\gamma+1}$, i.e., with high probability).

Let $U \subseteq E_T$ be the set of unusable edges marked by the subroutine. In this paragraph we find an upper bound for $|U|$. Consider **blaming** a part P_i for congesting an unusable edge $e \in U$ when $e \notin E_G[P_i] \cup H_i$ and e can see P_i, i.e., edge e was not in the canonical shortcut H_i, but e was congested by part P_i (and ultimately declared unusable). We argue via a Chernoff bound that each unusable edge $e \in U$ can see at least $2c$ parts.

The bound is argued in a completely analogous way as proving the congestion being at most $8c$, except the Chernoff bound we use here is the following one. As before, let $S := X_1 + X_2 + \ldots X_t$ be the sum of indicator variables of the part IDs that can see an edge the fixed edge e. Our bound stipulates that for independent $\{0, 1\}$ variables X_i we have that $\Pr[X_1 + X_2 + \ldots + X_t \leq 2\mathbb{E}[S]] \leq \exp(-\delta\mathbb{E}[S])$ for some $\delta > 0$. Using it, we conclude that if $t \leq 2c$ parts can see $e \in U$, then $\Pr[S \geq 2\mathbb{E}[S]] = \Pr[S \geq 4c \cdot p] \leq n^{-\gamma'}$ for some sufficiently large $\gamma' > 0$, giving us that in such a case the would not be declared unusable with high probability. Union bounding, we get the same holds for each $e \in U$.

Since each unusable edge $e \in U$ can see at least $2c$ parts, we blame at least $2c - \text{congestion}(\mathcal{H}) = c$ parts for congesting e. Each part can be blamed at most b times because each block component can only be blamed for the first unusable edge in his T-tree path towards the T-root. Furthermore, if e is unusable, it takes at least $2c - c$ different block components (from different parts) to be blamed for congesting e. Therefore $|U| \leq N\frac{b}{c}$.

We say that a part P_i **missed** an edge e when $e \in E_G[P_i] \cup H_i$ and $e \in U$ (consequently $e \notin H'_i$). Furthermore, call a part **bad** if it missed at least $2b$ edges and **good** otherwise. Note that if a part P_i is good, the block parameter of H'_i is at most $2b + \text{blockParameter}(\mathcal{H}) = 3b$. This is because each missed edge induces a new block component in $\mathcal{H}'$ (more

precisely, we can identify each block component of $\mathcal{H}'$ by either an unique block component of $\mathcal{H}$ or an unique missed edge $e \in U$). Consequently, it is sufficient to prove that the subroutine finds at least $\frac{1}{2}N$ good parts.

As any unusable edge is assigned to at most c parts in the canonical shortcut and for a part to be bad we need at least $2b$ edges to be missed, we have that the number of bad parts is at most $|U|\frac{c}{2b} \leq \frac{1}{2}N$. Hence, the subroutine finds at least $\frac{1}{2}N$ good shortcuts.

The subroutine takes $O(D \log n + c)$ rounds: on each of the $O(D)$ levels of the tree T all the nodes in parallel must send the active part IDs that its parent edge can see. If an edge e is not unusable, we argued via a Chernoff bound that at most $O(c \cdot p) = O(\log n)$ active part IDs can be seen from e, hence the number of rounds for determining unusable edges is $O(D \log n)$, w.h.p. Finally, propagating the part IDs upwards along T described in Lemma 2 takes $O(D + c)$ rounds, bringing the total number of rounds to $O(D \log n + c)$. □

References

1. Das Sarma, A., Holzer, S., Kor, L., Korman, A., Nanongkai, D., Pandurangan, G., Peleg, D., Wattenhofer, R.: Distributed verification and hardness of distributed approximation. In: Proceedings of the Symposium on Theory of Computing (STOC), pp. 363–372 (2011)
2. Elkin, M.: Unconditional lower bounds on the time-approximation tradeoffs for the distributed minimum spanning tree problem. In: Proceedings of the Symposium on Theory of Computing (STOC), pp. 331–340 (2004)
3. Elkin, M.: An unconditional lower bound on the time-approximation trade-off for the distributed minimum spanning tree problem. SIAM J. Comput. **36**(2), 433–456 (2006)
4. Frischknecht, Silvio., Holzer, Stephan., Wattenhofer, Roger.: Networks cannot compute their diameter in sublinear time. In: Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 1150–1162 (2012)
5. Ghaffari, M., Haeupler, B.: Distributed algorithms for planar networks I: Planar embedding. Manuscript, (2015)
6. Ghaffari, M., Haeupler, B.: Distributed algorithms for planar networks II: Low-congestion shortcuts, mst, and min-cut. In: Proceedings of ACM-SIAM Symposium on Discrete Algorithm (SODA), pp. 202–219. SIAM, (2016)
7. Ghaffari, M., Kuhn, F.: Distributed minimum cut approximation. In: Proceedings of the International Symposium on Distributed Computing (DISC), pp. 1–15 (2013)
8. Ghaffari, M., Karrenbauer, A., Kuhn, F., Lenzen, C., Patt-Shamir, B.: Near-optimal distributed maximum flow: Extended abstract. In: The Proceedings of the International Symposim on Principles of Distributed Computing (PODC), pp. 81–90 (2015)
9. Garay, J.A., Kutten, S.., Peleg, D.: A sub-linear time distributed algorithm for minimum-weight spanning trees. In: Proceedings of the Symposium on Foundations of Computer Science (FOCS), (1993)
10. Ghaffari, M., Li, J.: New distributed algorithms in almost mixing time via transformations from parallel algorithms. *arXiv preprint* arXiv:1805.04764, (2018)
11. Haeupler, B., Hershkowitz, D Ellis., Wajc, D.: Round-and message-optimal distributed graph algorithms. In: Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, pp. 119–128. ACM (2018)
12. Haeupler, B., Izumi, T., Zuzic, G.: Low-congestion shortcuts without embedding. In: Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing, pp. 451–460. ACM (2016)
13. Haeupler, B., Izumi, T., Zuzic, G.: Near-optimal low-congestion shortcuts on bounded parameter graphs. In: International Symposium on Distributed Computing, pp. 158–172. Springer (2016)
14. Haeupler, B., Li, J.: Faster distributed shortest path approximations via shortcuts. *arXiv preprint* arXiv:1802.03671 (2018)
15. Haeupler,B., Li, J., Zuzic, G.: Minor excluded network families admit fast distributed algorithms. In: Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, pp. 465–474. ACM (2018)
16. Holzer, S., Wattenhofer, R.: Optimal distributed all pairs shortest paths and applications. In: The Proceedings of the International Symposium on Principles of Distributed Computing (PODC), pp. 355–364 (2012)
17. Izumi, T., Wattenhofer, R.: Time lower bounds for distributed distance oracles. In: Proceedings of the International Conference on Principles of Distributed Systems, pp. 60–75 (2014)
18. Kutten, S., Peleg, D.: Fast distributed construction of k-dominating sets and applications. In: Proceedings of the International Symposium on Principles of Distributed Computing (PODC), pp 238–251 (1995)
19. Khan, M., Pandurangan, G.: A fast distributed approximation algorithm for minimum spanning trees. Distrib. Comput. **20**(6), 391–402 (2008)
20. Frank Thomson, L., Bruce M, M., Satish B, R.: Packet routing and job-shop scheduling in O(congestion+ dilation) steps. Combinatorica **14**(2), 167–186 (1994)
21. Lenzen, C., Patt-Shamir, B., Peleg, D.: Distributed distance computation and routing with small messages. Distrib. Comput. **32**(2), 133–157 (2019)
22. Nnongkai, D.: Distributed approximation algorithms for weighted shortest paths. In: Proceedings of the Symposium on Theory of Computing (STOC), pp. 565–573 (2014)
23. Nešetřil, J., Milková, E., Nešetřilová, H.: Otakar boruvka on minimum spanning tree problem translation of both the 1926 papers, comments, history. Discrete Math. **233**(1), 3–36 (2001)
24. Nanongkai, D., Su, H.-H.: Almost-tight distributed minimum cut algorithms. In: Proceedings of the International Symposium on Distributed Computing (DISC), pp 439–453 (2014)
25. Peleg, D.: Distributed Computing: A Locality-sensitive Approach. Society for Industrial and Applied Mathematics, Philadelphia (2000)
26. Peleg, D., Rubinovich, V.: A near-tight lower bound on the time complexity of distributed MST construction. In: Proceedings of the Symposium on Foundation of Computer Science (FOCS), p 253 (1999)