Minor Excluded Network Families Admit Fast Distributed Algorithms*

Bernhard Haeupler
Carnegie Mellon University
Pittsburgh, PA
haeupler@cs.cmu.edu

Jason Li
Carnegie Mellon University
Pittsburgh, PA
jmli@cs.cmu.edu

Goran Zuzic
Carnegie Mellon University
Pittsburgh, PA
gzuzic@cs.cmu.edu

ABSTRACT

Distributed network optimization problems, such as minimum spanning tree, minimum cut, and shortest path, are an active research area in distributed computing. This paper presents a fast distributed algorithm for such problems in the CONGEST model, on networks that exclude a fixed minor.

On general graphs, many optimization problems, including the ones mentioned above, require $\tilde{\Omega}(\sqrt{n})$ rounds of communication in the CONGEST model, even if the network graph has a much smaller diameter. Naturally, the next step in algorithm design is to design efficient algorithms which bypass this lower bound on a restricted class of graphs. Currently, the only known method of doing so uses the low-congestion shortcut framework of Ghaffari and Haeupler [SODA'16]. Building off of their work, this paper proves that excluded minor graphs admit high-quality shortcuts, leading to an $\tilde{O}(D^2)$ round algorithm for the aforementioned problems, where D is the diameter of the network graph. To work with excluded minor graph families, we utilize the Graph Structure Theorem of Robertson and Seymour. To the best of our knowledge, this is the first time the Graph Structure Theorem has been used for an algorithmic result in the distributed setting.

Even though the proof is involved, merely showing the existence of good shortcuts is sufficient to obtain simple, efficient distributed algorithms. In particular, the shortcut framework can efficiently construct near-optimal shortcuts and then use them to solve the optimization problems. This, combined with the very general family of excluded minor graphs, which includes most other important graph classes, makes this result of significant interest.

CCS CONCEPTS

• **Mathematics of computing** → **Graphs and surfaces**; • **Theory of computation** → *Distributed computing models*; *Distributed algorithms*;

*Supported in part by NSF grants CCF-1527110, CCF-1618280 and NSF CAREER award CCF-1750808. The full version of the paper is available at https://arxiv.org/abs/1801.06237

PODC '18, July 23–27, 2018, Egham, United Kingdom

ACM ISBN 978-1-4503-5795-1/18/07…$15.00
https://doi.org/10.1145/3212734.3212776

KEYWORDS

Distributed Computing; CONGEST; Graph Structure Theorem; Minor Excluded Graphs; Shortcuts; Low-congestion Shortcuts

ACM Reference Format:
Bernhard Haeupler, Jason Li, and Goran Zuzic. 2018. Minor Excluded Network Families Admit Fast Distributed Algorithms. In *PODC '18: ACM Symposium on Principles of Distributed Computing, July 23–27, 2018, Egham, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3212734.3212776

1 INTRODUCTION

Network optimization problems in the CONGEST model, such as Minimum Spanning Tree, Min-Cut or Shortest Path are an active research area in theoretical distributed computing [8, 11, 22, 27]. This paper provides a fast distributed algorithm for such problems in excluded minor graphs in the CONGEST model.

Recently, lower bounds have been established on many distributed network optimization problems, including all of the mentioned ones [5]. More specifically, each of these problems in the CONGEST model require $\tilde{\Omega}(\sqrt{n})$[1] rounds of communication to solve. This holds even on graphs with small diameter, for example, when the diameter is logarithmic in the number of nodes n. The result is surprising since it is not immediately clear why a network of small diameter requires such a large number of communication rounds when solving an optimization problem.

On the positive side, the next major question in algorithmic design is to determine whether one can bypass this barrier by restricting the class of network graphs. One immediate question that arises is, what family of graphs should one consider? Ideally, such a class of graphs should be inclusive enough to admit most "realistic" networks, yet be restrictive enough to disallow the pathological lower bound instances.

In our search for a restricted graph family to study, we focus on three criteria. First, we desire a family with a *rich and rigorous mathematical theory*, so that our result is technically meaningful. Second, the family should capture many networks in practice. And finally, we want *robustness*: a graph with a few added or perturbed edges and vertices should still remain in the family. Robustness is an important goal, since we want our graph family to be resistant to noise. For example, planar graphs satisfy the first two criteria, but fail to be robust since often adding a single random edge will make the graph non-planar. Indeed, most algorithms on planar graphs fail completely when run on a planar graph with a few perturbed edges and vertices. Next, one might try genus-bounded graphs, but they

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

also suffer from similar problems since adding a single randomly connected vertex can arbitrarily increase the genus.

A candidate graph family that fulfills all three conditions is the family of *excluded minor* graphs, namely the graphs which do not have a fixed graph H as a minor. This family encompasses several classes of naturally occuring networks. For example, trees which exclude K_3, planar graphs that capture the structure of two-dimensional maps exclude K_5 and $K_{3,3}$, and, series-parallel graphs that capture many network backbones exclude K_4 [1, 9]. Excluded minor graphs also have a history of deep results, including the series of Graph Minor papers by Robertson and Seymour.

In this paper, we provide efficient distributed algorithms for the class of excluded minor graphs which break the $\tilde{\Omega}(\sqrt{n} + D)$ lower bound for general graphs, giving evidence that most practical networks admit efficient distributed algorithms. We show an $\tilde{O}(D^2)$ algorithm for MST and $(1 + \epsilon)$ approximate min-cut, among other results. For networks having low diameter, such as $D = \text{poly}(\log n)$ or $D = n^{o(1)}$, our algorithms are optimal up to $\text{poly}(\log n)$ or $n^{o(1)}$ factors, respectively. This is a significant improvement over previous MST and min-cut algorithms, which run in $\Omega(\sqrt{n})$ time even on an excluded minor graph with $D = \text{poly}(\log n)$, such as a planar graph with an added vertex attached to every other node.

Our results use the framework of **low-congestion shortcuts**, introduced by [12] and built on by [15], which is a combinatorial abstraction to designing distributed algorithms. It introduces a simple, combinatorial problem involving **shortcuts** on a graph, and guarantees that a good **quality** solution to this combinatorial problem automatically translates to a simple, efficient distributed algorithm for MST and $(1 + \epsilon)$ approximate min-cut, among other problems; the concepts of shortcuts and quality will be defined later. Actually, the algorithm is the same regardless of the network or the graph family; the purpose of the combinatorial shortcuts problem is to prove that the algorithm runs *efficiently* on the graph or family.

To solve the shortcuts problem on excluded minor graphs, we appeal to the Graph Structure Theorem of Robertson and Seymour [29, 30]. At a high level, the Graph Structure Theorem decomposes every excluded minor graph into a set of almost-planar graphs connected in a tree-like fashion. Our solution to the shortcuts problem is in fact a series of results, one for each step in the structure decomposition. We remark that our result is, to the best of our knowledge, the first in distributed computing to make use of the Graph Structure Theorem to claim a distributed algorithm is fast. The absence of such a preceding result in distributed computing is unsurprising, since algorithms working with the Graph Structure Theorem generally require computing the required decomposition beforehand, and no efficient distributed algorithm to do so is known. Even the best classical algorithm still takes $O(n^3)$ time [21], so even a sublinear distributed algorithm is still out of reach. However, our result is unique in that we merely show the *existence* of a solution to the shortcuts problem in excluded minor graphs, and as a consequence, the simple algorithm of [15]—which does not look at any structure in the network graph, let alone compute a decomposition—is proven to run efficiently on excluded minor graphs.

The fact that this algorithm does not actually compute the Graph Structure Theorem should be stressed further. Since the framework of [15] computes a shortcut competitive to the optimal one, a consequence is that the running time of this algorithm rarely depends on the (large) constants appearing in the Graph Structure Theorem. In other words, while we can only *prove* that the constants in the running time are bounded by (some functions of) the constants in the Graph Structure Theorem, the actual running time of the algorithm is likely to be much smaller. In fact, for most excluded minor networks, we expect the running time to be $\tilde{O}(D^2)$ with a small constant, or even $\tilde{O}(D)$. In contrast, algorithms that explicitly compute a Graph Structure Theorem decomposition have an inherent bottleneck in the form of the potentially huge constants of the Graph Structure Theorem.

This paper is structured as follows. After the introduction, we begin with introducing the two main tools necessary for our main result, namely the Graph Structure Theorem and the low-congestion shortcuts framework. Then, we prove the existence of good shortcuts one step at a time, following the step-by-step construction in the Graph Structure Theorem.

1.1 Outline of the Proof

The goal of this section is to outline the proof of our main result, without delving into the technical details. Some concepts will be left undefined (e.g., shortcut quality, tree-restricted shortcuts), since the definitions are technical and require a lot of motivation beforehand. However, one can think of shortcuts as a combinatorial construction on a graph, and think of quality as a metric with which to measure a solution.

Theorem 1. *[Haeupler et al. [15, 16]] Suppose that a graph with diameter D admits tree-restricted shortcuts of quality $q : \mathbb{N} \to \mathbb{N}$. Then, there is an $\tilde{O}(q(D))$-round distributed algorithm for MST and $(1 + \epsilon)$-approximate min-cut for that graph.*

Our main technical result is showing the existence of good tree-restricted shortcuts in excluded minor graphs.

Theorem 2. *[Main Theorem] Every graph in a graph family excluding a fixed minor H admits tree-restricted shortcuts of quality $q(d) = \tilde{O}(d^2)$. The constants in the big-O depend only on the minor H.*

Combining the above Theorem 1 and Theorem 2, we get out main result.

Corollary 1. *There exists an $\tilde{O}(D^2)$-round distributed algorithm for MST and $(1 + \epsilon)$-approximate min-cut, for any $\epsilon > 0$, on graph networks excluding a fixed minor.*

For excluded minor graphs of diameter $n^{o(1)}$, as is the case for many practical networks, our algorithms also run in $n^{o(1)}$ time, which is optimal up to lower order terms. This is a significant improvement over the previously known $\tilde{\Omega}(\sqrt{n})$ time algorithms and it avoids the $\tilde{\Omega}(\sqrt{n})$ lower bound for general graphs, even when they have $n^{o(1)}$ diameter.

Corollary 2. *There exists an $n^{o(1)}$-round distributed algorithm for MST and $(1 + \epsilon)$-approximate min-cut, for any $\epsilon > 0$, on graph networks with diameter $n^{o(1)}$ and excluding a fixed minor.*

In order to work with excluded minor families, we appeal to the Robertson-Seymour Graph Structure Theorem. At a high level,

this theorem states that every graph in an excluded minor family can be decomposed into a set of graph **almost embeddable** in a bounded genus surface that are glued together in a tree-like fashion. Naturally, our approach is to first construct good-quality shortcuts for the entire family of almost embeddable graphs, and then modify them in a robust manner as they are patched together in the composition. While this approach works in general, the patching required is very involved because of various interactions between the many ingredients involved in the decomposition. For example, one step in the construction of an almost embeddable graph is the addition of an "apex" vertex that connects arbitrarily to all previous vertices. While the addition of only one vertex appears harmless at first glance, observe that the diameter can shrink arbitrarily, e.g., to 2 if the apex is connected to all other vertices. If the graph without the apex has large diameter D, and its shortcuts solution leads to an $\tilde{O}(D^2)$-round algorithm, this same algorithm will not suffice on the graph with the apex, which can have diameter 2. A lot of technical effort goes into reconstructing shortcuts upon the addition of an apex, in order to handle the arbitrary decrease in graph diameter. Hence, as a consequence of all these difficulties, we settle for $\tilde{O}(d^2)$-quality shortcuts, and leave the improvement to $\tilde{O}(d)$-quality shortcuts as an open problem.

1.2 Related Work

Work on global network problems in the distributed setting was started by Gallager, Humblet and Spira [10] who gave a $O(n \log n)$-round algorithm to compute the MST. The algorithm was subsequently improved by Awerbuch [2] to an "optimal" $O(n)$ rounds. However, Peleg and Awerbuch [3] noted that a more useful notion of round complexity would parametrize on both the number of nodes n and the diameter D since $D \ll n$ for many practical networks. This influenced a substantial amount of work that followed, culminating in an $\tilde{O}(D + \sqrt{n})$ distributed algorithm for many optimizations problems in the CONGEST setting. Examples of such problems include the MST [8, 11, 22], $(1 + \epsilon)$-approximate Maximum Flow [13], Minimum Cut [14, 27], Shortest Paths, and Diameter [7, 18–20, 23, 24, 26]. The mentioned $\tilde{O}(D + \sqrt{n})$ round complexity is **existentially optimal** for all of these problems, as there exists a family of graphs for which any algorithm must take a matching $\tilde{\Omega}(D + \sqrt{n})$ rounds [5, 6].

Some early work that tried to circumvent the $\Omega(\sqrt{n})$ bound was done by Khan and Pandurangan[22] who argued that their MST algorithm performed in an **universally optimal** manner on some restricted classes of graphs. In particular, they gave a $\tilde{O}(D)$ round MST algorithm for (1) unit disk networks where weights match the distances, and (2) networks with IID random weights. Their approach used a novel parameter called "local shortest path diameter" that takes the edge weights into account. This makes their approach unsuitable for arbitrary weights and topologically constrained networks such as planar graphs.

However, an alternative approach has recently made progress for global optimization problems on restricted graph classes. Ghaffari and Haeupler showcased a distributed MST and $(1+\epsilon)$-approximate min-cut algorithm that runs in $\tilde{O}(D)$ rounds on planar graphs [12], which was later simplified and generalized to bounded treewidth graphs and bounded genus graphs [16]. All of these results use the aforementioned low-congestion shortcuts framework.

1.3 Preliminaries

For a graph G, let $V(G)$ and $E(G)$ denote the vertices and edges, respectively. Given $P \subseteq V(G)$, $G[P]$ denotes the induced subgraph, namely, the one obtained by removing $V(G) \setminus P$ from G. Finally, when G is the underlying network graph, we always assume that G is connected and contains no self-loops (which can be ignored in the distributed setting anyway).

Due to space constraints, we move the preliminary concerning the Graph Structure Theorem to Section 2; the reader is encouraged to review it to become familiar with the used terminology.

1.3.1 CONGEST model. While we only use the classical CONGEST model indirectly, via Theorem 1 of [15], we will state its assumptions for context. A network is given by a connected graph $G = (V, E)$ with n nodes and diameter D. Communication proceeds in synchronous rounds. In each round, each node can send a different $O(\log n)$ bit message to each of its neighbors. Local computations are free and require no time. Nodes have no initial knowledge of the topology G, except that we assume that they know n and D up to constants. One can easily remove these assumptions by distributively computing these parameters in $O(D)$ time, which is negligible in our context.

2 PRELIMINARY: GRAPH STRUCTURE THEOREM

In this section we introduce Robertson and Seymour's Graph Structure Theorem, following the survey of Lovász [25]. This theorem is instrumental in our shortcut construction, since it provides structure for all graphs that are H-free, for any minor H. At a high level, the theorem says that every H-free graph can be glued together in a tree-like fashion from graphs that can be "almost" embedded in a fixed surface. To elaborate on this statement, we need a few definitions. The first definition, k-clique-sum, captures the tree-like structure of the graph.

Definition 1 (k-clique-sum)**.** *Let G_1 and G_2 be two graphs, and let $S_i \subseteq G_i$ be a k-clique for $i = 1, 2$. Let G be obtained by identifying S_1 with S_2 and deleting some (possibly none, possibly all) edges between the nodes in $S_1 = S_2$. We say that G is a k-**clique-sum** of G_1 and G_2. More generally, G is a k-clique-sum of $G_1, G_2, \ldots, G_\ell$ if G is formed by starting with G_1 and iteratively taking the k-clique-sum of the current graph with G_i, for $i = 2, \ldots, \ell$ in that order.*

The next few definitions classify the graphs which are almost embeddable on a surface. We start with the three main ingredients in constructing such a graph, and then define what it means to be almost-embeddable.

Definition 2 (Apex)**.** *Define **adding an apex to graph** G as follows: create a new vertex called the **apex**, and connect it to an arbitrary subset of the vertices in G.*

Definition 3 (Surface of genus g)**.** *A graph G has genus g if there is a **2-cell embedding** in a surface of genus g. In other words, this means: (i) there exists an oriented or unoriented surface (i.e., 2-manifold) Σ of genus g, (ii) vertices of G are mapped to distinct points of Σ, (iii) edges are mapped to simple paths whose interiors do not contain any vertices of G nor do path interiors mutually intersect, and (iv) each face defined by such embedding is homeomorphic to a unit disk, i.e., contains no holes or handles in it.*

Definition 4 (Vortex [25]). *Let G be a graph with a 2-cell embedding. Let C be a cycle in G that corresponds to a face on the surface. Call a continuous interval on the cycle an **arc**. Select a family of arcs on C so that each node is contained in at most k of these arcs. For each arc A, create a new node v_A and connect v_A to a subset of the vertices in C that lie on arc A. Such nodes v_A are called **internal vortex nodes**. Finally, for any two arcs A and B sharing a common vertex in C, we may add the edge $\{v_A, v_B\}$. We call this operation **adding a vortex of depth k to cycle** C.*

Definition 5. *A graph G is (q, g, k, ℓ)-**almost-embeddable** if it can be constructed according to the three steps below.*

(i) Start with a graph G' embedded on a surface of genus at most g.
(ii) We select at most ℓ faces of G' and add a vortex of depth at most k to each of them. Call the result G''.
(iii) We add q apices to G'', connected arbitrarily to vertices in G'' and to each other, and obtain the desired graph G.

*For simpler notation, we say a graph is h-**almost-embeddable** if it is (h, h, h, h)-almost embeddable.*

By this definition, the planar graphs are precisely the $(0, 0, 0, 0)$-almost-embeddable graphs, and the genus-g graphs are precisely the $(0, g, 0, 0)$-almost-embeddable graphs. Later on, we will study the planar graphs with added vortices, in particular the $(0, 0, k, \ell)$-almost-embeddable graphs for constants k and ℓ.

As a final ingredient to the Graph Structure Theorem, we construct a graph family $\mathcal{L}_k$ as follows.

Definition 6. *Let $\mathcal{L}_k$ denote all graphs that can be represented as a k-clique-sum of k-almost-embeddable graphs. That is, a graph G is in $\mathcal{L}_k$ if there exist k-almost-embeddable graphs $G_1, G_2, \ldots, G_\ell$ such that G is a k-clique-sum of $G_1, G_2, \ldots, G_\ell$.*

In other words, take any set of k-almost-embeddable graphs $G_1, G_2, \ldots, G_\ell$ for $\ell \geq 1$, and let G be their k-clique-sum. Construct a graph G by repeatedly taking a k-clique-sum operation between multiple G_i's constructed using step (i)–(iii) and connect them in a tree-like fashion. Define $\mathcal{L}_k$ as precisely all graphs G that can be constructed in this way.

Finally, we present the Graph Structure Theorem, which states that for any H, there is a k such that $\mathcal{L}_m$ includes (but does not exactly characterize) all graphs that are H-free [25].

Theorem 3 (Graph Structure Theorem). *For every graph H there is a fixed integer $k = k(H)$ such that any H-free graph G is contained in $\mathcal{L}_k$.*

Below, we include additional terminology on clique-sums and vortices used in the shortcut construction.

Definition 7 (Vortex terminology). *Let C be a cycle of G, and add a vortex of depth k to C, following Definition 4. Let $v_1, v_2, \ldots$ be the vertices created when adding a vortex of depth k to C. The vertices in C form the **vortex boundary**, and the added vertices $v_1, v_2, \ldots$ are called **inside the vortex** and **internal vortex** nodes. Moreover, suppose an internal vertex v_i corresponds to **arc** A_i of C in the vortex construction. Define the **vortex decomposition** to be the map $\mathcal{P}$ satisfying $\mathcal{P}(v_i) = A_i$. Finally, if G is embedded on a closed surface such that C forms a face in the embedding, then that face is called the **vortex face**.*

Definition 8 (k-Clique-sum decomposition tree). *Let the graph G be constructed as the k-clique-sum of subgraphs $B_1, B_2, \ldots, B_\ell$. The subgraphs $B_i \subseteq G$ are denoted as **bags**. A k-**clique-sum decomposition tree** of G is a tree $\mathbb{DT}$ whose vertices $V(\mathbb{DT})$ are identified with bags B_i. The edges of the decomposition $f \in E(\mathbb{DT})$ correspond to a clique in two of the bags, with possibly some removed edges. Therefore, we refer to them as **partial** k-**cliques** C_f. The decomposition satisfies the following properties:*

(1) $\bigcup_{i \in \mathbb{DT}} V(B_i) = V(G)$.
(2) For all $i \in V(\mathbb{DT})$, $B_i \subseteq G$.
(3) For all $f = \{i, j\} \in E(\mathbb{DT})$, $B_i \cap B_j = C_f$.
(4) For all $v \in V(G)$, the set $\{i \in V(\mathbb{DT}) \mid v \in V(B_i)\}$ is connected in $\mathbb{DT}$.
(5) For all $e \in E(G)$, there exists $i \in V(\mathbb{DT})$ with $e \in E(B_i)$.

We conclude with a statement that the above clique-sum decomposition tree captures all possible ways to take clique-sums of graphs.

Fact 1. *Let a graph G be the k-clique-sum of graphs from a family $\mathcal{F}$. Then, G has a k-clique-sum decomposition tree whose bags are graphs in $\mathcal{F}$.*

2.0.1 Tree-restricted Shortcuts. This section introduces the shortcut framework from [12] and [15]. We state the definitions and concisely repeat the motivations behind them.

Imagine solving the following problem in a distributed manner: *Each node in the network is assigned a number x_v. The network is partitioned into a number of vertex disjoint individually-connected parts and each node wants to compute the minimum/maximum/sum of x_v between all nodes in its own part.* The above subproblem often occurs in algorithm design. Notably, it appears in Boruvka's Minimum Spanning Tree algorithm [28]. A naive solution would spread the information about x_v independently inside each part, because the problem statement basically makes them independent. However, this will incur performance penalties if the diameter of the parts in isolation is much larger than the diameter of the entire graph. For instance, a wheel graph has $\Theta(1)$ diameter, while a part containing all the outer nodes has $\Theta(n)$ diameter. This leads us to the idea of "helping" a part by assigning it additional edges that it can use to spread information. We call these edges "shortcuts".

Definition 9 (General Shortcuts). *Let $G = (V, E_G)$ be an undirected graph with vertices subdivided into **pairwise disjoint and connected** subsets $\mathcal{P} = (P_1, P_2, ..., P_{|\mathcal{P}|}), P_i \subseteq V$. In other words, $G[P_i]$ 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 a tuple of **shortcut edges** $(H_1, H_2, \ldots, H_{|\mathcal{P}|}), H_i \subseteq E(G)$.*

The shortcut framework has particularly nice properties if the shortcut edges are restricted to some tree T, typically the BFS tree. In particular, they can be near-optimally constructed in a distributed and uniform manner on any network. We first define this structured version of shortcuts, enumerate the quality measures for them, and finally state the relevant theorems.

Definition 10 (Tree-Restricted Shortcuts). *Let $\mathcal{H} = (H_1, H_2, ..., H_{|\mathcal{P}|})$ be a shortcut on the graph $G = (V, E_G)$ with respect to the parts $\mathcal{P} = (P_i)_i$. Given a spanning tree $T = (V, E_T) \subseteq G$ we say that a*

*shortcut $\mathcal{H}$ is T-**restricted** if for each part $P_i \in \mathcal{P}$, its shortcut edges $H_i \subseteq E_T$. In other words, every edge of H_i lies on the tree T.*

Note that the above definitions do not impose any hard condition on H_i with respect to P_i. We introduce these conditions implicitly by defining **congestion, block**, and **quality** parameters to measure how good they are (smaller is better).

Definition 11 (Congestion)**.** *Let $\mathcal{H} = (H_1, H_2, ..., H_{|\mathcal{P}|})$ be a shortcut on $G = (V, E_G)$. Define a congestion over an edge e as $c_e := |\{i : e \in H_i\}|$, the number of shortcuts using an edge. Finally, define the **congestion of the shortcut** to be $\max_{e \in E_G} c_e$, the maximum congestion of any edge.*

Another beneficial property we would like is that the subgraphs $G[P_i]+H_i$ have small diameter. By $G[P_i]+H_i$ we mean the subgraph constructed by taking the induced subgraph $G[P_i]$ and then adding all the edges in H_i as well as any of their endpoints not contained in P_i. We will measure diameter of $G[P_i] + H_i$ indirectly, by defining a **block parameter** b that essentially measures the number of different subtrees (components) that exist in H_i. We use it in the following way: note that (1) $G[P_i]$ is connected by definition, (2) $G[P_i] + H_i$ effectively looks like b_i interconnected subtrees, (3) each subtree has diameter $O(D)$ since T is typically a spanning tree whose height is at most the diameter of G. From these properties we conclude that $G[P_i] + H_i$ has diameter $O(b_i D)$.

Definition 12 (Block parameter)**.** *For a shortcut $\mathcal{H} = (H_1, H_2, ..., H_{|\mathcal{P}|})$, fix a part P_i and consider the connected components of the spanning subgraph (V, H_i). If a connected component contains a node in P_i we call it a **block component**. They correspond to subtrees of T. We define that $\mathcal{H}$ has **block parameter** b if no part has more than b block components.*

*In general, for functions $b, c : \mathbb{N} \to \mathbb{N}$, a graph G is said to **admit** tree-restricted shortcuts with block parameter b and congestion c if for **any** spanning tree T with diameter at most d_T and any family of parts there exists a T-restricted shortcut with block parameter $b(d_T)$ and congestion $c(d_T)$. "Good" tree-restricted shortcuts typically have $\tilde{O}(1)$ block parameter and $\tilde{O}(d_T)$ congestion.*

Similarly, a family of graphs admits tree-restricted shortcuts with block parameter b and congestion c if all graphs in the family individually admit them.

We now define the **quality** of a tree-restricted shortcut. The terminology of admitting tree-restricted shortcuts carries over to quality.

Definition 13 (Quality)**.** *The **quality** of a tree-restricted shortcut is the function $q : \mathbb{N} \to \mathbb{N}$ defined by $q(d) = b(d) \cdot d + c(d)$.*

We now restate the central theorem from the low-congestion shortcuts framework.

Theorem 1. *[Haeupler et al. [15, 16]] Suppose that a graph with diameter D admits tree-restricted shortcuts of quality $q : \mathbb{N} \to \mathbb{N}$. Then, there is an $\tilde{O}(q(D))$-round distributed algorithm for MST and $(1+\epsilon)$-approximate min-cut for that graph.*

Note that D replaces d_T in the argument to the functions b and c. This is because the theorem takes T to be a BFS tree of the network graph, which has diameter at most D.

Finally, it is known from [12, 16] that good tree-restricted shortcuts exist in planar graphs and bounded treewidth graphs. We will be using this fact in a later section.

Theorem 4 ([12]). *The family of planar graphs admits tree-restricted shortcuts with block parameter $O(\log d_T)$ and congestion $O(d_T \log d_T)$.*

Theorem 5 ([16]). *The family of graphs of treewidth at most k admits tree-restricted shortcuts with block parameter $O(k)$ and congestion $O(k \log n)$.*

3 SHORTCUTS IN EXCLUDED MINOR GRAPHS

Our main result extends Theorem 4 and Theorem 5 to excluded minor graphs, showing that any family of graphs excluding a fixed minor has good tree-restricted shortcuts. We repeat Theorem 2 with a bit of extra detail.

Theorem 6. *[Main Theorem, Extended Version] The family of graphs excluding a fixed minor H admits tree-restricted shortcuts of quality $q(d) = \tilde{O}(d^2)$. More generally, the family admits block parameter $b(d) = O(d)$ and congestion $c(d) = O(d \log n + \log^2 n)$. The constants in the big-O depend only on the minor H.*

Using the shortcuts framework of Theorem 1, the above theorem translates to the algorithmic result of Corollary 1.

3.1 Two Parts of the Proof

Recall that the Graph Structure Theorem says that any excluded minor graph can be represented as a k-clique-sum of k-almost-embeddable graphs, for some constant k depending on the excluded minor. As with most results utilizing the Graph Structure Theorem, our proof is split into two parts, one handling the k-clique-sums and one for the k-almost-embeddable graphs.

Our proof has two main components, namely, Theorem 7 and Theorem 8 that we state below. It should be clear that they are sufficient to prove the main technical result, Theorem 6.

Clique Sums Part: In the k-clique-sums part, we show that if a family of graphs admits shortcuts with good quality, then so does any k-clique-sum of graphs from this family, for any constant k. In other words, having good tree-restricted shortcuts is a property robust under taking k-clique-sums for a fixed integer k. The theorem below is proved in Section 4.

Theorem 7. *[Shortcuts in Clique Sums] Let $\mathcal{F}$ be a family of graphs that admits tree-restricted shortcuts with block parameter $b_{\mathcal{F}}$ and congestion $c_{\mathcal{F}}$. Let G be a k-clique-sum of graphs in $\mathcal{F}$. Then G admits tree-restricted shortcuts with block parameter $b_G(d) \leq 2k + O(b_{\mathcal{F}}(d_T))$ and congestion $c_G(d) \leq O(k \log^2 n) + c_{\mathcal{F}}(d_T)$.*

On a high level, the proof relies on carefully charging the congestion to bags in the k-clique-sum decomposition. This leads to a bound that relies on the depth of the decomposition, which can be controlled by folding up long bag-paths in the decomposition.

To prove the full result, we use Theorem 7 with $\mathcal{F}$ as the family of k-almost-embeddable graphs, which we show admits tree-restricted shortcuts with block parameter and congestion $\tilde{O}(d)$. Plugging in these parameters, we obtain $b_G(d) = 2k + \tilde{O}(d)$ and $c_G(d) = O(k \log^2 n) + \tilde{O}(d)$ for the final result, which are both $\tilde{O}(d)$ since k

is a constant. Note that Theorem 7 does not assume that $\mathcal{F}$ is any particular family, so it may be of independent interest.

Almost Embeddable Part: The second part of the proof establishes good quality shortcuts for k-almost-embeddable graphs, namely the theorem below, proved in Section 5.

Theorem 8. *[Shortcuts in Almost Embeddable Graphs] An (q, g, k, ℓ)-almost-embeddable graph G admits tree-restricted shortcuts with block parameter $b(d) = O(q + (g+1)k\ell^2 d)$ and congestion $c(d) = O(q + k\ell^2 d(g + \log n))$.*

The proof is fairly technical and uses several novel ideas. The most prominent one is the construction of a structure we call the "combinatorial gate". Intuitively, when given a partition of a genus-bounded graph into balls of low diameter, there exists a small number of special vertices such that any subgraph that intersects many balls has to contain many of these special vertices. We use these combinatorial gates to set up a careful charging scheme that allows high-diameter parts to be given more shortcut edges without a catastrophic increase in the congestion.

Putting Them Together: The main theorem, restated below, follows immediately from Theorem 7 and Theorem 8.

Theorem 6. *[Main Theorem, Extended Version] The family of graphs excluding a fixed minor H admits tree-restricted shortcuts of quality $q(d) = \tilde{O}(d^2)$. More generally, the family admits block parameter $b(d) = O(d)$ and congestion $c(d) = O(d \log n + \log^2 n)$. The constants in the big-O depend only on the minor H.*

Proof. By Theorem 3, there is a constant k such that the family of H-free graphs is contained in $\mathcal{L}_k$, so it suffices to prove the claim for $\mathcal{L}_k$. Let $\mathcal{F}$ be the family of k-almost-embeddable graphs. By Theorem 8, $\mathcal{F}$ admits tree-restricted shortcuts with block parameter $b_{\mathcal{F}}(d) = O(d)$ and congestion $c_{\mathcal{F}}(d) = O(d \log n)$. Plugging in $\mathcal{F}$, $b_{\mathcal{F}}$, and $c_{\mathcal{F}}$ into Theorem 7, we conclude that $\mathcal{L}_k$ admits tree-restricted shortcuts with block parameter $O(d)$ and congestion $O(d \log n + \log^2 n)$, as desired. □

4 SHORTCUTS IN CLIQUE SUM GRAPHS

In this section, we prove Theorem 7, restated below.

Theorem 7. *[Shortcuts in Clique Sums] Let $\mathcal{F}$ be a family of graphs that admits tree-restricted shortcuts with block parameter $b_{\mathcal{F}}$ and congestion $c_{\mathcal{F}}$. Let G be a k-clique-sum of graphs in $\mathcal{F}$. Then G admits tree-restricted shortcuts with block parameter $b_G(d) \leq 2k + O(b_{\mathcal{F}}(d_T))$ and congestion $c_G(d) \leq O(k \log^2 n) + c_{\mathcal{F}}(d_T)$.*

Local and Global Shortcuts: The intuition behind our construction is as follows. Let G be a k-clique-sum of graphs in $\mathcal{F}$, and consider a k-clique-sum decomposition tree $\mathbb{DT}$ of G. Its existence is guaranteed by Fact 1. Consider a part $P \subseteq V(G)$, which could either span much of a single bag in $\mathbb{DT}$, or traverse through multiple bags, or both. As a result, we construct two types of shortcuts—**local** shortcuts and **global** shortcuts—to handle each case separately. At a high level, local shortcuts, which are constrained within a single bag, are meant to deal with parts that behave wildly within a bag, while global shortcuts, which can span multiple bags, treat parts that stretch across many different bags. In particular, for each part P, we specify one bag on which we construct local shortcuts for P, and let global shortcuts handle the rest. The shortcut for P is simply the union of the local and global shortcuts.

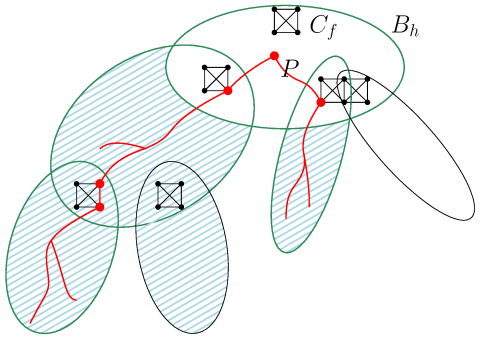

Figure 1: Global shortcut construction. The part P is shown in red. The global T-restricted shortcut is the intersection of T (not shown) with the shaded region. C_f denotes the partial clique leading to the parent of h, which is not used in the global shortcut.

Root $\mathbb{DT}$ at an arbitrary bag, and define $d_{\mathbb{DT}}$ to be the depth of the rooted tree $\mathbb{DT}$. We first prove a weaker result whose global shortcut depends on the value of $d_{\mathbb{DT}}$ in its congestion, then later show how to "compress" $\mathbb{DT}$ to a low depth independent of $d_{\mathbb{DT}}$, thereby removing the dependence of $d_{\mathbb{DT}}$.

Lemma 1. *Let $\mathcal{F}$ be a family of graphs that admits tree-restricted shortcuts with block parameter $b_{\mathcal{F}}$ and congestion $c_{\mathcal{F}}$. Let G be a k-clique-sum of graphs in $\mathcal{F}$ with decomposition tree $\mathbb{DT}$. Then G admits tree-restricted shortcuts with block parameter $b_G(d_T) \leq k + b_{\mathcal{F}}(d_T)$ and congestion $c_G(d_T) \leq k d_{\mathbb{DT}} + c_{\mathcal{F}}(d_T)$. (Note the dependence on $d_{\mathbb{DT}}$, the depth of the decomposition tree $\mathbb{DT}$, which is unrelated to d_T, the diameter of the spanning tree T.)*

Proof. Let T be an arbitrarily rooted spanning tree of G of diameter d_T. Take a k-clique-sum decomposition tree $\mathbb{DT}$, root it at an arbitrary bag, and suppose that the rooted tree has depth $d_{\mathbb{DT}}$. In the rooted setting, define the set $\text{desc}(i) \subseteq V(\mathbb{DT})$ for $i \in V(\mathbb{DT})$ to be i along with all of its descendants in $\mathbb{DT}$.

Consider a part $P \subseteq V(G)$. Since P is connected, we know, by properties (4) and (5) of Definition 8, that the set of bags $S_P := \{j \in V(\mathbb{DT}) \mid V(B_j) \cap P \neq \emptyset\}$ is connected in $\mathbb{DT}$. Therefore, the lowest common ancestor, denoted by h_P, of S_P is also inside S_P. Similarly, for an edge $e \in E(G)$ we can define the set of bags that contain that edge $S_e := \{j \in V(\mathbb{DT}) \mid e \in E(B_j)\}$ and its lowest common ancestor $h_e \in S_e$.

Global Shortcuts: See Figure 1. The construction of the global shortcut is simple. For each edge f' to a child i of h_P such that $P \cap V(C_{f'}) \neq \emptyset$, allow part P to use all edges in $\left(\bigcup_{j \in \text{desc}(i)} E(B_j) \cap T\right) \setminus E(B_{h_P})$. Informally, the global shortcut "takes care" of all vertices in P except for those in B_{h_P}, which leaves constructing the local shortcut for P in B_{h_P}. More precisely, remember that T is rooted and consider the roots of the block components of P when using only the global shortcut: they are restricted to B_{h_P}.

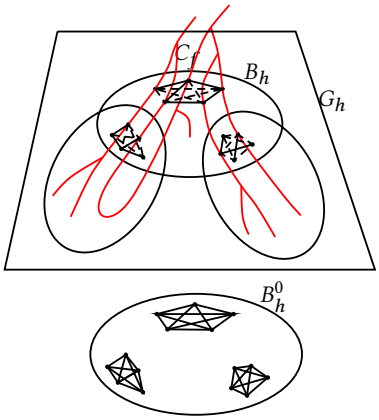

Figure 2: Local shortcut construction. On the left, T is solid red. Dotted black edges are edges absent from the partial k-cliques. On the right is B_h^0 for the B_h on the left.

We now argue about the congestion. Consider an edge $e \in E(G)$, and let $\mathcal{B}$ be the set of bags on the $\mathbb{DT}$-root-path to h_e, including h_e. Clearly, $|\mathcal{B}| \leq d_{\mathbb{DT}}$. Edge e can only be assigned to parts that contain a vertex in the partial-clique on a parent edge of a bag in $\mathcal{B}$. Hence its congestion is at most $k|\mathcal{B}| \leq kd_{\mathbb{DT}}$.

Local Shortcuts: See Figure 2. Let h be an arbitrary bag, we apply the following argument to all of them. We now focus on the local shortcut within B_h. Let $T_h^1 := T \cap B_h$ be the forest when we look at B_h in isolation (note that the tree T can become disconnected). We will repair T_h^1 in the next paragraph.

Let $B_h^0 \in \mathcal{F}$ be the original bag of B_h, which is B_h with all partial k-cliques involved in the clique-sum completed to full k-cliques (see Figure 2). In particular, $V(B_h) = V(B_h^0)$. We emphasize that $B_h^0 \in \mathcal{F}$ by the definition of partial-cliques.

In order to find a tree-restricted shortcut on B_h, we have to define the tree. The forest $T_h^1 := T \cap B_h^0$ might be disconnected, so we have to repair it. First, we define a **path contraction** operation between two vertices $s, t \in V(B_h^0)$. Consider the unique path between s and t in T, represented as a sequence of vertices $s = u_0, u_1, \ldots, u_* = t$. Delete any vertex $u_i \notin V(B_h)$ and one is left with (a sequence of vertices representing a) valid path in B_h^0 between s and t. Note that the contracted path is a graph minor of T.

We form the repaired tree T_h^2 in the following way: for every two $s, t \in V(B_h^0)$, take the path contraction between them and union it into T_h^2. It is clear that (1) T_h^2 is a subgraph of B_h^0, in fact, it is a spanning tree of B_h^0, (2) T_h^1 is a subgraph of T_h^2, and (3) T_h^2 is a contraction of T. The last property implies that T_h^2 is connected and that its diameter is at most d_T. Also, note that the same argument shows that for any part P, its restriction $B_h^0[P]$ is also connected since we can contract any path inside P and the resulting path is still in B_h^0 and contains only vertices in P—the only unimportant difference being that this path might be on T.

Next, construct a T_h^2-restricted shortcut, discard all edges in $T_h^2 \setminus T = T_h^2 \setminus T_h^1$, and discard all edges contained in C_f, where f is the parent $\mathbb{DT}$-edge of h. The resulting assignment is the local shortcut of B_h.

The congestion of the local shortcut is $c_{\mathcal{F}}(d_T)$. Fix an edge $e \in E(G)$, and note that it is only locally assigned in the bag h_e (due to discarding edge of C_f). But the local congestion of h_e is $c_{\mathcal{F}}(d_T)$, as claimed. The total congestion is at most the sum of the local and global one, hence it is at most $kd_{\mathbb{DT}} + c_{\mathcal{F}}(d_T)$.

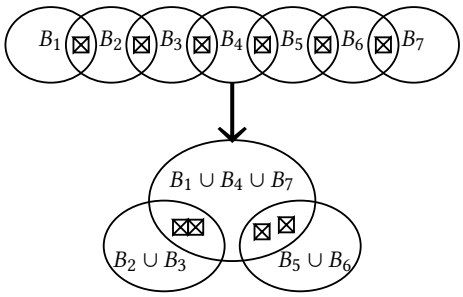

Figure 3: Compressing a k-clique-sum decomposition tree with high depth.

Bounding the Block Parameter: With all shortcut edges established, we now upper bound the block parameter for each part $P \subseteq V(G)$. Remember that T is, arbitrarily, rooted. We will bound the number of nodes $v \in V(G)$ that are roots of block components. Note that $v \in B_{h_P}$ since otherwise the global shortcut assigns the T-parent edge of v to P. But in the lowest common ancestor B_{h_P}, v can be a block root only if either (a) it is a vertex in C_f, where f is the parent $\mathbb{DT}$-edge of h_P, or (b) it is a block root of a local shortcut inside B_{h_P}. Summing up the contributions of these two cases, the total number of block roots, and therefore block components, can be at most $k + b_{\mathcal{F}}(d_T)$.

□

To improve the $d_{\mathbb{DT}}$ factor in the congestion and prove the main result of this section, we compress the decomposition tree $\mathbb{DT}$ to reduce its depth to $O(\log^2 n)$, in a similar way to the compression scheme in [4] for treewidth decompositions.

Theorem 7. *[Shortcuts in Clique Sums] Let $\mathcal{F}$ be a family of graphs that admits tree-restricted shortcuts with block parameter $b_{\mathcal{F}}$ and congestion $c_{\mathcal{F}}$. Let G be a k-clique-sum of graphs in $\mathcal{F}$. Then G admits tree-restricted shortcuts with block parameter $b_G(d) \leq 2k + O(b_{\mathcal{F}}(d_T))$ and congestion $c_G(d) \leq O(k \log^2 n) + c_{\mathcal{F}}(d_T)$.*

Proof. Let $\mathbb{DT}$ be a k-clique-sum decomposition tree of G. To motivate the main proof, we first consider the case when $\mathbb{DT}$ is a single path from root to leaf. This case will directly help in the general case, in which we apply *heavy-light decomposition* to the tree, breaking it up into chains, and then treat each chain as a single path; we will present this general case next.

Case When $\mathbb{DT}$ Is a Path: Assume that $\mathbb{DT}$ is a rooted path with bags $B_1, \ldots, B_{d_{\mathbb{DT}}}$, in that order. We recursively construct a balanced binary decomposition tree $\mathbb{DT}'$ as follows.

(1) Group the bags $B_1, B_{\lceil d_{\mathbb{DT}}/2 \rceil}, B_{d_{\mathbb{DT}}}$ into a single bag B_r.
(2) Recursively solve the paths $B_2, \ldots, B_{\lceil d_{\mathbb{DT}}/2 \rceil - 1}$ and $B_{\lceil d_{\mathbb{DT}}/2 \rceil + 1}, \ldots, B_{d_{\mathbb{DT}}}$.

(3) Attach the two resulting trees as subtrees of B_r (see Figure 3).

We call this operation **folding** a path.

Call the new decomposition tree $\mathbb{DT}'$; it is almost a k-clique-sum decomposition tree, with one exception: an edge may no longer a partial k-clique, but a union of two partial k-cliques. We call such edges **double edges**. Note that, while we can add edges within each of the two partial k-cliques and keep the graph in the family $\mathcal{F}$, we cannot add edges between a vertex in one partial k-clique and a vertex in the other. Hence, we cannot simply treat the union of two partial k-cliques as a single partial $2k$-clique. However, a bag B_i can have at most two children connected by double edges.

Using the terminology of the above proof, let B_h^0 still be the bag B_h with all partial cliques filled in with edges (the union of two cliques in a double edge will not have edges between them). The only difference this incurs in the proof is the following: in the global shortcut, partial cliques on the edge of $\mathbb{DT}$ can now contain $2k$ vertices instead of k, doubling the congestion; and, in the local shortcut, a part restricted to to a bag $B_h^0[P]$ might not be connected anymore. However, we claim that it consists of at most $O(1)$ connected components: for each connected component we find a "representative vertex" in that component as follows. If (1) the component touches a partial clique in a double edge to a child, then the representative is the lowest numbered vertex in such a partial clique, and otherwise (2) we pick any vertex in the component. Now there will be at most $O(1)$ different representatives, thereby finishing the claim since no two different components can have the same representative. One can see this by arguing if (1) a part touches a partial clique in a double edge to a child, the it has at most 4 possibilities; otherwise (2) the part is already connected via the previous proof.

We construct local shortcuts considering connected components of the parts as separate (sub)parts and union the assignment in the end. This only decreases the congestion, and increases the block by a multiplicative $O(1)$ to a total of $2k + O(b_{\mathcal{F}}(d_T))$.

We now discuss the general case, when $\mathbb{DT}$ is an arbitrary tree. The main steps of the proof are as follows. First, we compute a *heavy-light decomposition* [17] of $\mathbb{DT}$. Then, we fold every chain in the heavy-light decomposition the same way we fold a single path, so that the resulting tree decomposition has depth $O(\log^2 n)$.

Heavy-Light Decomposition: The heavy-light decomposition is a decomposition of any rooted tree into vertex-disjoint paths, called **heavy chains**, such that any path from the root to a leaf changes at most $O(\log n)$ heavy chains, where n is the number of vertices in the tree. The decomposition is simple: for each non-leaf vertex of the tree, connect it to the child vertex with the largest number of vertices in its own subtree. On any path from root to leaf, if traveling from vertex u to vertex v changes heavy chains, then vertex u has at least twice as many vertices in its subtree than does v; such an event can only occur $\log_2 n$ times along the path.

Folding a Chain: Once we compute the heavy-light decomposition, we partition the vertices of $\mathbb{DT}$ into heavy chains, and then fold each chain independently. Then, we connect the resulting binary trees in the following natural way: if the root of chain C_1 is a child of some vertex v, then we connect the root of the binary tree of C_1 to v. Note that this is not a double edge. We get a rooted tree $\mathbb{DT}'$ of depth $O(\log^2 n)$ with the following key property: while every vertex in the new decomposition tree can have many children, it has at most two children connected via double edges. Therefore, the same argument for double edges in the single path case also applies here. With the depth of $\mathbb{DT}'$ reduced to $O(\log^2 n)$, the result follows. □

5 SHORTCUTS IN ALMOST EMBEDDABLE GRAPHS

In this section, we prove Theorem 8. In particular, we prove that k-almost-embeddable graphs admit good shortcuts. Recall that these graphs have bounded genus with an additional constant number of apices and vortices of constant depth added.

5.1 Apex Graphs

In this section, we add apices to $(0, g, k, l)$-almost-embeddable ("Genus + Vortex") graphs. At first glance, the addition of an apex to a graph might seem trivial, since the graph only changes by one vertex, and using that vertex can only make the shortcuts better. However, notice that the diameter of the graph can shrink arbitrarily with the addition of an apex, and our shortcuts on the apex graph must be competitive with the new diameter. Hence, we need ideas beyond our shortcut constructions for the graph without the apex. For a simple example, in a cycle graph, shortcuts with quality $\Theta(n)$ are considered good. However, by adding a single central vertex, we can transform the graph into the wheel graph where "good" shortcuts should have quality $\Theta(1)$. While good shortcuts actually do exist in the wheel graph, there are examples of graphs with good shortcuts where adding a apex makes good shortcuts impossible.

To streamline our arguments for (q, g, k, l)-almost-embeddable graphs ("Apex+Genus+Vortex") graphs, we will define a couple of intermediate properties which do not depend on the graph topology. More precisely, we will define the notions of β-**cell-assignment** and s-**combinatorial gates**. On a very high level, We will show that:

(1) A Genus+Vortex graph has an s-combinatorial gate, for an appropriately chosen s. (Section 5.3 and the Appendix)
(2) Graphs with s-combinatorial gate are β-cell-assignable, for appropriately chosen β and some technical stipulations. (Section 5.2)
(3) Graphs that are β-cell-assignable and each cell locally admits good tree-restricted shortcuts also globally admit good tree-restricted shortcuts, barring various technicalities. (Section 5.4)

In each part, we separately prove the statements with Genus+Vortex graphs replaced by planar graphs. It is recommended that the reader, in their first reading, focus only on the lemmas regarding planar graphs with a single apex, namely Lemmas 2, 3, 5, and 7.

5.2 Cell Partitions, β-Cell-Assignment and s-Combinatorial Gate

In this section, we first introduce the notions of "cell partitions", "β-cell-assignment" and "s-combinatorial gates". Second, we prove that the second property implies the first.

Definition 14. *A **cell partition** of G is simply a partition of V_G into disjoint, connected components with a small diameter, called the **cells**.*

Note that the diameter condition is the only thing differentiating it from the definition of parts. It is helpful to think of cells as low-diameter components, whereas parts may be long and skinny. A canonical example for a cell partition is the following. Given an apex graph of diameter D, remove the apex and start a concurrent BFS from each node adjacent to the removed apex. Each node in the graph (except the apex) gets assigned to exactly one BFS component. We call such BFS components cells. For most of this section, we will ignore any extra property that a cell partition might have and assuming nothing besides them being disjoint, connected and having a controlled diameter.

A graph is cell-assignable if we can relate its cells and parts in a way that no cell is assigned to too many parts and parts are assigned to **almost all** intersecting cells.

Definition 15. *A graph $G = (V_G, E_G)$ is β-cell-assignable if the following holds. For every valid family of parts $\mathcal{P}$ (as in Definition 9) and every valid cell partition C of diameter d there exists a relation $\mathcal{R} \subseteq C \times \mathcal{P}$ with the following properties:*

*(i) each part is in relation with **all** cells it intersects, **except** for at most 2 of them*
(ii) each cell is in relation with at most β parts

Note: β is a function of the cell diameter d.

We will not prove directly that Genus+Vortex graphs are β-cell-assignable. Instead, we focus on a combinatorial property that we show implies cell-assignment. This property is called a "combinatorial gate" and it intuitively asserts that every two touching cells have a "gate" that covers all the edges between them. Furthermore, the boundary of such a gate is called a "fence" and its size should be controlled. The reader is encouraged to review **??** for a mental picture of combinatorial gates on a planar graph.

Definition 16. *For a subset of vertices $S \subseteq V$, define the ∂S to be the set of vertices in S on the boundary of S, i.e., the vertices in S whose neighborhoods intersect $V \setminus S$.*

Definition 17. *Let $G = (V, E)$ be a graph from a family $\mathcal{F}$, and let C be a partition of G into cells. We define a s-**combinatorial gate** to be a collection $\mathcal{S} = \{(F_i, S_i)\}_i$ where $F \subseteq V$ are called **fences**, $S \subseteq V$ are **gates**, and the following properties hold:*

(1) Fences are a subset of their corresponding gates. I.e., $F \subseteq S$ for all $(F, S) \in \mathcal{S}$.
(2) The boundary of a gate are included in its fence. I.e., $\partial(S) \subseteq F$ for all $(F, S) \in \mathcal{S}$.
(3) Each edge $\{a, b\} \in E$ whose endpoints are in different cells must be covered by some gate. I.e., $a \in S \wedge b \in S$ for some gate S.
(4) Each gate S intersects at most two cells in C.
(5) The non-fence vertices of the gates are disjoint. I.e., for every $v \in V$ there is at most one $(F_i, S_i) \in \mathcal{S}$ s.t. $v \in S_i \setminus F_i$.
(6) The average size of fences compared to the number of cells is at most s. I.e., $\sum_{(F,S)\in\mathcal{S}} |F| \leq s|C|$.

Since this condition is entirely combinatorial, the proofs that imply β-cell-assignment are also combinatorial. Therefore, these results are self-contained and disregard any possible structure in the graph, for example, planarity. Next, we state that the s-combinatorial boundary implies β-cell-assignment via the following two lemmas. The proofs are omitted due to space constraints.

Lemma 2. *Suppose a graph G with cell partition C has an s-combinatorial gate $\mathcal{S}$. Then, for any collection of parts $\mathcal{P}$, either there exists a part intersecting at most two cells, or there exists a cell intersecting at most $2s$ parts.*

Lemma 3. *Let $\mathcal{F}$ be a family of graphs that is closed under taking minors. Suppose that there is a function $s(d) : \mathbb{N} \to \mathbb{N}$ such that every graph $G \in \mathcal{F}$ satisfies the following property:*

- *If G has a cell partition of diameter d, then there exists an $s(d)$-combinatorial gate $\mathcal{S}$ of subsets of $V(G)$.*

Then, every graph $G \in \mathcal{F}$ with a cell partition of diameter d is $2s(d)$-cell-assignable.

While Lemma 3 works well for planar graphs that are closed under taking minors, Genus+Vortex graphs do not have that property due to the existence of a bounded number of vortices. In particular, if one contracts an edge inside the vortex, the resulting graph is not Genus+Vortex. Therefore, we will deal with cells touching vortices as "special cells" that are not allowed to be contracted.

Lemma 4. *Let $\mathcal{F}$ be a family of graphs, not necessarily closed under taking minors. Suppose that there is a function $s(d) : \mathbb{N} \to \mathbb{N}$ such that every graph $G \in \mathcal{F}$ satisfies the following property:*

- *If G has a cell partition of diameter d, then there exists an $s(d)$-combinatorial gate $\mathcal{S}$ of subsets of $V(G)$.*

Consider a graph $G \in \mathcal{F}$ with a cell partition into two types of cells—normal cells and ℓ special cells—both of diameter d. Let E^* denote the set of edges in special cells. Assume that any graph G' obtained by deleting vertices and contracting edges outside of special cells is still in $\mathcal{F}$. Then, G is $2\ell s(d)$-cell-assignable with respect to a cell partition of only the normal cells.

5.3 Graphs with s-Combinatorial Gate Property

In this section, we show that Genus+Vortex graphs satisfy the s-combinatorial property. We highlight our main ideas by proving the statement for planar graphs before moving on to genus-bounded graphs.

Lemma 5. *Let G be a planar graph with a cell partition of diameter d. Then, there is an $36d$-combinatorial gate $\mathcal{S}$.*

Lemma 6. *Let G be a genus-g graph with (a possibly unbounded number of) vortices of depth k, and consider a cell partition of diameter d such that no vortex is split between more than one cell. Then, there exists an $O((g+1)kd)$-combinatorial gate of G.*

5.4 Wrapping Up: From β-Cell-Assignment to Good Shortcuts

In this section, we finalize our proof for tree-restricted shortcuts in almost embeddable graphs. We do this by showing that if an $(0, g, k, l)$-almost-embeddable ("Genus+Vortex") graph is β-cell-assignable for small enough parameter β, then the same graph with q added apices admits good tree-restricted shortcuts. We first assume that the apex graph has exactly one apex, then establish a simple reduction from the multiple apices case. We begin with the same statement for $(1, 0, 0, 0)$-almost-embeddable ("Apex+Planar") graphs, continue with the full statement apart from the single apex, and finally finish with the most general statement. The proofs are omitted due to space constraints.

Lemma 7. *Let G be a planar graph with a single apex and a diameter d_T spanning tree T of G. For a given set of parts, there exists a T-restricted shortcut with block parameter $O(\log d_T)$ and congestion $O(d_T \log d_T)$.*

Lemma 8. *Let G be a genus-g graph with ℓ vortices of depth k and a single apex and T a spanning tree of G. For a given set of parts, there exists a T-restricted shortcut with block parameter $O((g+1)k\ell^2 d_T)$ and congestion $O(k\ell^2 d_T(g+\log n))$.*

Theorem 8. *[Shortcuts in Almost Embeddable Graphs] An (q, g, k, ℓ)-almost-embeddable graph G admits tree-restricted shortcuts with block parameter $b(d) = O(q + (g+1)k\ell^2 d)$ and congestion $c(d) = O(q + k\ell^2 d(g + \log n))$.*

Proof. Let G be the apex graph and T a the spanning tree of G. If a part contains one of the q apices, we give the entire tree T to the part. This increases the congestion by at most q. For the remaining parts, we do the following. First, add an auxiliary new vertex x that connects to each of the q apices; the diameter can grow by at most 1. Contract these $q+1$ vertices to a single apex to form graph G'; T might now contain cycles, so take a spanning subtree of depth d_T in the contracted T. Apply Lemma 8 to the single apex graph G'. If we extend the shortcuts for each part in the natural way to G, the congestion does not change any further. Furthermore, the block parameter increases by at most $q-1$ because a block component containing x splits into at most q block components. □

6 CONCLUSION AND OPEN PROBLEMS

We have proved all the ingredients we need to prove our main theorem, which we restate for convenience.

Theorem 6. *[Main Theorem, Extended Version] The family of graphs excluding a fixed minor H admits tree-restricted shortcuts of quality $q(d) = \tilde{O}(d^2)$. More generally, the family admits block parameter $b(d) = O(d)$ and congestion $c(d) = O(d \log n + \log^2 n)$. The constants in the big-O depend only on the minor H.*

An obvious open question is whether the block parameter $O(d_T)$ can be improved to $\tilde{O}(1)$, which would result in a near-optimal $\tilde{O}(D)$-round algorithm for MST and $(1+\epsilon)$-approximate mincut on excluded minor network graphs. The bottleneck in the current proof lies in the treewidth argument when arguing about Genus+Vortex graph, which produces the $O(d_T)$ block parameter. This treewidth argument cannot be improved due to lower bounds on treewidth-k graphs, as presented in [16]. Hence, an improvement on Genus+Vortex graphs requires a better understanding of vortices, beyond treating them as simply low-treewidth (or pathwidth) graphs.

REFERENCES

[1] Ittai Abraham and Cyril Gavoille. 2006. Object location using path separators. In *Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*. ACM, 188–197.

[2] Baruch Awerbuch. 1987. Optimal distributed algorithms for minimum weight spanning tree, counting, leader election, and related problems. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*. ACM, 230–240.

[3] Baruch Awerbuch. 1989. Randomized distributed shortest paths algorithms. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*. ACM, 490–500.

[4] Hans L Bodlaender. 1988. NC-algorithms for graphs with small treewidth. In *International Workshop on Graph-Theoretic Concepts in Computer Science*. Springer, 1–10.

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

[6] Michael Elkin. 2006. An unconditional lower bound on the time-approximation trade-off for the distributed minimum spanning tree problem. *SIAM J. Comput.* 36, 2 (2006), 433–456.

[7] Michael Elkin. 2017. Distributed exact shortest paths in sublinear time. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. ACM, 757–770.

[8] Michael Elkin. 2017. A simple deterministic distributed MST algorithm, with near-optimal time and message complexities. *arXiv preprint arXiv:1703.02411* (2017).

[9] Paola Flocchini and Flaminia L Luccio. 2003. Routing in series parallel networks. *Theory of Computing Systems* 36, 2 (2003), 137–157.

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

[11] Juan A Garay, Shay Kutten, and David Peleg. 1998. A sublinear time distributed algorithm for minimum-weight spanning trees. *SIAM J. Comput.* 27, 1 (1998), 302–316.

[12] Mohsen Ghaffari and Bernhard Haeupler. 2016. Distributed algorithms for planar networks II: Low-congestion shortcuts, mst, and min-cut. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 202–219.

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

[14] Mohsen Ghaffari and Fabian Kuhn. 2013. Distributed minimum cut approximation. In *International Symposium on Distributed Computing*. Springer, 1–15.

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

[16] Bernhard Haeupler, Taisuke Izumi, and Goran Zuzic. 2016. Near-Optimal Low-Congestion Shortcuts on Bounded Parameter Graphs. In *International Symposium on Distributed Computing*. Springer, 158–172.

[17] Dov Harel and Robert Endre Tarjan. 1984. Fast algorithms for finding nearest common ancestors. *siam Journal on Computing* 13, 2 (1984), 338–355.

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

[19] Chien-Chung Huang, Danupon Nanongkai, and Thatchaphol Saranurak. 2017. Distributed Exact Weighted All-Pairs Shortest Paths in $\tilde{O}(n^{5/4})$ Rounds. *arXiv preprint arXiv:1708.03903* (2017).

[20] Taisuke Izumi and Roger Wattenhofer. 2014. Time Lower Bounds for Distributed Distance Oracles.. In *OPODIS*. 60–75.

[21] Ken-ichi Kawarabayashi and Paul Wollan. 2011. A simpler algorithm and shorter proof for the graph minor decomposition. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*. ACM, 451–458.

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

[23] Christoph Lenzen and Boaz Patt-Shamir. 2015. Fast Partial Distance Estimation and Applications. 153–162.

[24] Christoph Lenzen and David Peleg. 2013. Efficient Distributed Source Detection with Limited Bandwidth. 375–382.

[25] László Lovász. 2006. Graph minor theory. *Bull. Amer. Math. Soc.* 43, 1 (2006), 75–86.

[26] Danupon Nanongkai. 2014. Distributed approximation algorithms for weighted shortest paths. In *Proceedings of the ACM Symposium on Theory of Computing*. 565–573.

[27] Danupon Nanongkai and Hsin-Hao Su. 2014. Almost-tight distributed minimum cut algorithms. In *International Symposium on Distributed Computing*. Springer, 439–453.

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

[29] Neil Robertson and Paul D Seymour. 1986. Graph minors. V. Excluding a planar graph. *Journal of Combinatorial Theory, Series B* 41, 1 (1986), 92–114.

[30] Neil Robertson and Paul D Seymour. 2003. Graph minors. XVI. Excluding a non-planar graph. *Journal of Combinatorial Theory, Series B* 89, 1 (2003), 43–76.