Distributed Planar Reachability in Nearly Optimal Time

Merav Parter
Weizmann Institute of Science, Rehovot, Israel
merav.parter@weizmann.ac.il

Abstract

We present nearly optimal distributed algorithms for fundamental reachability problems in planar graphs. In the *single-source reachability problem* given is an n-vertex directed graph $G = (V, E)$ and a source node s, it is required to determine the subset of nodes that are reachable from s in G. We present the first distributed reachability algorithm for planar graphs that runs in nearly optimal time of $\widetilde{O}(D)$ rounds, where D is the undirected diameter of the graph. This improves the complexity of $\widetilde{O}(D^2)$ rounds implied by the recent work of [Li and Parter, STOC'19].

We also consider the more general reachability problem of identifying the *strongly connected components* (SCCs) of the graph. We present an $\widetilde{O}(D)$-round algorithm that computes for each node in the graph an identifier of its strongly connected component in G. No non-trivial upper bound for this problem (even in general graphs) has been known before.

Our algorithms are based on characterizing the structural interactions between balanced cycle separators. We show that the reachability relations between separator nodes can be compressed due to a Monge-like property of their directed shortest paths. The algorithmic results are obtained by combining this structural characterization with the recursive graph partitioning machinery of [Li and Parter, STOC'19].

2012 ACM Subject Classification Networks → Network algorithms

Keywords and phrases Distributed Graph Algorithms, Planar Graphs, Reachability

Digital Object Identifier 10.4230/LIPIcs.DISC.2020.38

Funding Partially funded by the ISF, grant no. 713130

1 Introduction

Reachability problems in directed graphs are among the most fundamental graph problems, and as such receive quite a lot of attention in various computational settings. In this paper, we consider two canonical reachability problems for the family of planar graphs. In the *single-source reachability* problem, given a digraph $G = (V, E)$ and a source vertex s, it is required to identify the set of vertices reachable from s. In the *strong connectivity identification* problem, given a digraph $G = (V, E)$, it is required to identify the strongly connected components of G.

Throughout this paper, we consider the standard CONGEST model of distributed computing [34]. In this model, the network is abstracted as an n-vertex graph $G = (V, E)$, with one processor on each vertex. Initially, these processors only know their incident edges in the graph, and the algorithm proceeds in synchronous communication rounds over the graph $G = (V, E)$. In each round, vertices are allowed to exchange $O(\log n)$ bits with their neighbors and perform local computation. We follow the standard assumption that the communication graph is bidirectional[1].

[1] This is the common assumption in the setting of distributed graph algorithms for directed graphs, cf. [9, 5].

The single-source reachability problem along with its cousin the single-source shortest path (SSSP) problem are among the most actively studied problems in the CONGEST model [6, 33, 17, 27, 22, 7, 10, 15, 5, 25, 23]. In their seminal work, Das-Sarma et al. [6] showed a lower bound of $\Omega(D + \sqrt{n}/\log n)$ rounds for the single-source reachability problem for *general* n-vertex graphs with undirected diameter D. Nanongkai [33] gave the first non-trivial upper bound of $O(D + \sqrt{nD})$ rounds for the problem. Shortly after, Ghaffari and Udwani [17] improved the round complexity to $O(D + \sqrt{n}D^{1/4})$ rounds. This remained the state-of-the-art, and many efforts went into providing an SSSP algorithm with a matching round complexity. Very recently, Jambulapati, Liu and Sidford [25] gave an improved bound of $\widetilde{O}(\sqrt{n} + n^{1/3+o(1)} \cdot D^{2/3})$ rounds[2] for the single-source reachability problem.

In a recent work, Forster and Nanongkai [10], building on prior works of Gabow [11] and Klein and Subramanian [26], have established a strong connection between the problems of approximate SSSP in directed graphs and exact SSSP in undirected graphs. Specifically, they gave a quite general transformation that converts an approximate SSSP solution into an exact one, based on the recursive scaling approach [11, 26]. For the reduction to hold, the approximate SSSP algorithm is required to work for directed graphs, even if the initial input graph is undirected. This further strengthens the motivation for understanding reachability problems in the distributed setting.

In this work, we provide the first reachability algorithms for the family planar graphs which has been studied thoroughly in the distributed setting. The primary motivation for studying this family is two-fold. First, planar graphs are ubiquitous in real life communication. Secondly, this graph family escapes the well-known lower bound of $\Omega(D + \sqrt{n})$-rounds by Das Sarma et al. [35], giving the hope to solve many of the classical global graph problems in optimal time of $O(D)$ rounds. The area of *distributed planar algorithms* for global graph problems was introduced in a sequence of two works of Ghaffari and Haeupler [13, 14]. In their work, they introduced the useful notion of *low-congestion shortcuts* which provide a unified framework for solving global problems in the CONGEST model. Using this machinery, [14] presented improved algorithms for the minimum spanning tree and minimum-cut problems. Low-congestion shortcuts and their algorithmic applications have been further studied in [19, 20, 21, 28, 18]. An additional key algorithmic tool for planar graphs is given by the notion of *balanced separators* [30]. Ghaffari and Parter [16] presented a distributed construction of balanced cycle separators in the nearly optimal time of $\widetilde{O}(D)$ rounds.

The most relevant work to our paper is by Li and Parter [29] that presented poly(D) algorithms for distance computation in planar graphs. In their work, they introduced a very useful tool that allows one to efficiently apply a divide and conquer based approach using the balanced cycle separator of [16]. Specifically, as already observed in [16], the key limitation of applying the cycle separator algorithm in a recursive manner is rooted in the fact that the diameter of the graphs, obtained throughout the recursive process might be very large (e.g., once the cycle separator is removed). This ultimately leads to a large cycle separator, which increases the round complexity of the algorithm. To mitigate this technicality, [29] presented the notion of *bounded diameter decomposition* (BDD). Roughly speaking, the BDD is recursive decomposition of the graph in a balanced manner, in a way that guarantees that (i) the diameter of each subgraph is $\widetilde{O}(D)$ and (ii) all subgraphs at the same recursion layer are nearly edge disjoint. Using the BDD, [29] showed a poly$(D, 1/\epsilon)$-round algorithm for computing a $(1+\epsilon)$ approximation of the diameter in weighted graphs; and $\widetilde{O}(D^2)$ algorithms for computing the *exact* distance labels and SSSP trees. Very recently, Abboud, Cohen-Addad

[2] The $\widetilde{O}$ notation hides poly logarithmic factors in the number of vertices n.

and Klien [1] showed that the exact computation of the weighted diameter requires $\Omega(n)$ rounds, even if the underlying unweighted diameter is constant. This demonstrates a gap between the problem of exact SSSP and diameter computation in weighted planar graphs.

Finally, another line of research related to our work is given by the metric compression schemes for planar graphs. Abboud et al. [2] gave an efficient (centralized) compression scheme to encode the $S \times S$ distances in a unweighted undirected graphs. Using the unit-Monge property, they showed that the $S \times S$ distances can be encoded in $O(\min\{|S|^2, \sqrt{|S|n}\})$ bits. In this paper we consider *directed* graphs, and aimed at compressing reachability information in the *distributed* setting. Nevertheless, the heart of our compression technique is inspired by that of [2] and shares several technical similarities. That is, the directed paths in our work have a Monge-like property in a similar manner to the unweighted undirected shortest paths of [2]. Whereas the Monge property has been heavily utilized in many centralized algorithms for planar graphs [8, 32, 31, 24, 3, 4], our paper shows that it is also useful in the distributed setting.

1.1 Our Results

We present distributed algorithms for several fundamental graph problems in *directed* planar graphs. Starting with the single-source reachability problem, we provide an improved algorithm for computing reachability labels of size $\widetilde{O}(D)$ bits within $\widetilde{O}(D)$ rounds. Given the labels of any vertex pair u and v, one determine if there is a directed path from u to v in G and vice-versa. Given these labels, the single source reachability problem can be solved by broadcasting the $\widetilde{O}(D)$-bit label of the source to the entire network.

▶ **Theorem 1.** *[Reachability Labels and Single Source Reachability] There exists a randomized distributed algorithm that for every D-diameter planar digraph computes $\widetilde{O}(D)$-bit reachability labels within $\widetilde{O}(D)$ rounds, w.h.p.*[3] *This yields a single-source reachability algorithm with $\widetilde{O}(D)$ rounds.*

We then turn to consider the generalization of the problem to multiple sources S. A direct application of Thm. 1 gives an $\widetilde{O}(|S|D)$-round solution. We then show that when the S sources lie (not necessarily consecutively) on the boundary of a single face, the round complexity can be improved to $\widetilde{O}(D + |S|)$ rounds. This variant becomes useful for the identification of the strongly connected components of G, as will be explained next. Throughout the paper we use the notion of *virtual edges*. These are edges (u, v) that are not in G but their addition to G preservers its planarity. In the distributed setting, the endpoints of the virtual edges know their incident virtual edges and their (combinatorial) embedding in the planar graph.

▶ **Lemma 2.** *[Multi-Source Reachability] Let S be a collection of source vertices lying on a boundary of a single face of a planar graph G (possibly with one virtual edge). Then, there exists a randomized distributed algorithm that computes the $S \times V$ reachability in $\widetilde{O}(D + |S|)$ rounds w.h.p.*

Finally, we have almost all the necessary ingredients to identify the strongly connected components (SCCs) of the graph. To the best of our knowledge, there are no non-trivial algorithms for this problem, even for general graphs. One of the challenges in handling

[3] As usual, w.h.p. refers to success guarantee of $1 - 1/n^c$ for any given constant c, where n is the number of vertices.

this task is the following. When applying a recursive graph partitioning to the input graph, the SCCs of G are not necessarily extensions of the SCCs of the descendent subgraphs in this recursion. That is, the directed path P_1, P_2 from u to v and from v to u can become intact in different levels of the recursive partitioning. For that purpose, we designed a more delicate recursive scheme based on the multi-source reachability algorithm of Lemma 2. Our algorithm has a nearly optimal round complexity, which matches also the complexity of connectivity identification in undirected graphs by [14].

▶ **Lemma 3.** *[Strong Connectivity Decomposition] There exists a randomized algorithm that computes the strongly connected components (SCCs) of a planar graph G within $\widetilde{O}(D)$ rounds, w.h.p.*

Weighted Digraphs. We also consider reachability problems in *weighted* digraphs such as computing the directed SSSP tree. While our compression scheme cannot be extended to incorporate distances, we observe that the SSSP algorithm of [29] designated for undirected graphs, can in fact be easily adapted to the digraphs as well. We also show an algorithm for computing the divide minimum-weight directed cycle within $\widetilde{O}(D^2)$ rounds. We have:

▶ **Lemma 4** (Directed MSSP and Minimum Directed Girth)**.** *Given a weighted planar digraph $G = (V, E, w)$, there are randomized algorithms that w.h.p. compute (1) an exact MSSP w.r.t a subset of sources $S \subseteq V$ within $\widetilde{O}(D^2 + D \cdot |S|)$ rounds, and (2) a minimum weighted directed cycle within $\widetilde{O}(D^2)$ rounds.*

The most intriguing open question left by our work concerns the computation of SSSP in $o(D^2)$ rounds. We hope that our nearly optimal solution for the single-source reachability problem will serve the first step in that direction.

1.2 Technical Overview

Our algorithms are based on several existing tools by [16, 29] as well as several new tools, on which we elaborate more.

1.2.1 Balanced Separators and Bounded Tree Decompositions [16, 29]

A balanced separator of a planar graph G is a subset of vertices $S \subseteq V(G)$ whose removal breaks G into components of size at most $2/3n$. The celebrated result of Lipton and Tarjan [30] demonstrates the existence of a balanced cycle separator of size $O(D)$ for every planar graph with undirected diameter D. More generally, their proof shows that given a spanning tree T in G, there exist two tree paths in G plus one additional edge[4] (possibly not in G) that form a balanced *cycle* separator C for G. Ghaffari and Parter [16] gave an $\widetilde{O}(D)$-round randomized algorithm for computing these separators for biconnected planar graphs. This was later generalized for any planar graphs by [29]. In what follows, we denote the cycle separator of G by $\text{sep}(G)$.

Li and Parter [29] introduced the notion of *bounded-diameter decomposition* (BDD) of a D-diameter graph G. Viewed on the high-level, the BDD algorithm recursively applies the separator algorithm of [16] to decompose the graph into smaller and "almost" edge-disjoint subgraphs with bounded diameters. This recursive decomposition is represented by a tree $\mathcal{T}$, where each vertex of this tree, denoted as a *bag*, corresponds to subgraphs G' of G. We refer

[4] Adding that extra edge to G preserves its planarity. We refer to that edge as a *virtual edge*.

to the bags by their corresponding subgraphs. The root bag corresponds to G and the leaf bags correspond to subgraphs of G of size $O(D \log n)$. The child bags of each bag $G' \subseteq G$ are defined based on the cycle separator $\text{sep}(G')$ of G'. In an ideal scenario, the separator decomposes G' into two child bags G'_{in} and G'_{out} that are embedded in the interior and the exterior of the cycle $\text{sep}(G')$, respectively. The actual algorithm is more involved as it needs to satisfy several constraints to guarantee the efficiency of the distributed computation inside these bags. Due to these complications, the BDD algorithm might define several child bags $G'_1, \ldots, G'_\ell$ rather than just two.

The bounded diameter decomposition has several useful properties. The tree $\mathcal{T}$ has a logarithmic depth, as the child bags are defined based on *balanced* separators. The leaf bags contain $O(D \log n)$ vertices. In addition, in every level i of the tree $\mathcal{T}$, all the bags of that level are nearly edge-disjoint which allows one to work on all subgraphs of the same level in *parallel*. An important property of the decomposition is its *diameter preserving*: the diameter of each bag G' is bounded by $O(D \log n)$. Consequently, the separator size is bounded by $|\text{sep}(G')| = O(D \log n)$ for every bag G'. This property is crucial for the efficiency of divide-and-conquer based computations in G. Another useful property of the BDD algorithm is the following. For every bag G', each separator vertex $u \in \text{sep}(G')$ appears on at most three child bags of G'. Every other vertex belongs to a *unique* child bag. Keeping these properties in mind, we are now ready to highlight the key algorithmic ideas in our constructions.

1.2.2 New Approach for Reachability Labels via Reachability Preservers

At the heart of the single-source reachability algorithm lies an algorithm that computes short reachability labels for all the vertices in the graph. In the centralized setting, Thorup [36] gave an ingenious (centralized) technique to compute reachability labels of $\widetilde{O}(1)$ bits. These labels are designed based on applying various reachability computations on a modified graph $\widehat{G}$ obtained from G. Since in our setting we use the labels for the purpose of computing reachability, we will be using instead the labeling scheme by Gavoille et al. [12]. Although the latter labels have $\widetilde{O}(D)$ bits, their distributed computation is more natural, and as we will see can be done in $\widetilde{O}(D)$ rounds.

High-level description the distributed label computation. Our algorithm applies two recursive phases. The first phase of the algorithm computes in a bottom-up manner a succinct *reachability preserver* for the separator vertices. The second phase computes the reachability labels by applying a top-down recursion. Interestingly, the second phase requires no communication, and can be applied locally at each vertex based on their local knowledge on the BDD, and the reachability preservers constructed in the first phase.

For our recursive framework to hold, we introduce the notion of *extended separator set* that contains the separator vertices of G', as well as the separator vertices of all the *ancestor bags* of G' in the BDD tree. That is, letting $anc(G')$ be these ancestor bags of G', then the extended separator set, denoted by, $\text{sep}^+(G')$ is the union of $\text{sep}(G'')$ for every $G'' \in anc(G')$ and the vertices of $\text{sep}(G')$. The important observation is that since the depth of the BDD tree is logarithmic, the extended separator set of each bag G' is bounded by $\widetilde{O}(D)$. The main technical part of the algorithm is in the computation of the reachability preservers with respect to the extended separator sets. For a given sub-graph $G' \subseteq G$ and a subset of vertices S', a reachability preserver $H(S', G')$ is a graph whose vertex set is S' and for every $s, s' \in S'$, the edge (s, s') is in $H(S', G')$ iff there exists a directed path from s to s' in G'. Our goal is to compute for every bag G', a reachability preserver $H(G') = H(\text{sep}^+(G'), G')$

that captures the reachability relations between the vertices of $\text{sep}^+(G')$ in the graph G'. In the output format, it is required for every vertex u to learn the preserver $H(\text{sep}^+(G'), G')$ for every bag G' that contains u.

The preservers of the bags G' are built in a bottom-up manner, and their efficient computation is based on being able to locally compress the reachability information of the $\text{sep}^+(G')$ vertices. The leaf bags of the BDD contain just $O(D \log n)$ vertices and thus, every vertex can collect its leaf bag information and locally compute the preserver $H(G')$. Then, in every independent step of the recursion we are given a bag G' with child bags $G'_1, \ldots, G'_\ell$. By the induction, the vertices of G'_j have already computed the preservers $H(G'_j)$ for every j. It is then required to compute $H(G')$ within $\widetilde{O}(D)$ rounds. For every child bag G'_j, let $\widehat{H}(G'_j)$ be the preserver $H(G'_j)$ induced on the vertices of $\text{sep}^+(G') \cap V(G'_j)$. We then show that the preserver $H(G')$ can be obtained by computing the $\text{sep}^+(G') \times \text{sep}^+(G')$ reachability in the union of the graphs $\bigcup_j \widehat{H}(G'_j)$. Thus, the computation of the preserver $H(G')$ boils down into the following task for every child G'_j:

> *Every vertex in* $\text{sep}^+(G') \cap V(G'_j)$ *is required to send its incoming and outgoing neighbors*[5] *in the graph* $\widehat{H}(G'_j)$ *to all vertices in* G'.

Since each vertex in $\text{sep}^+(G') \cap V(G'_j)$ might have $\Omega(D)$ neighbors in $\widehat{H}(G'_j)$, sending these neighbors, explicitly, leads to a total of $|\text{sep}^+(G') \cap V(G'_j)| \cdot \Omega(D) = \Omega(D^2)$ bits of information. Our key observation is that the information on the incoming and outgoing neighbors in the graph $\widehat{H}(G'_j)$ can be compressed into just $\widetilde{O}(1)$ bits! Using this compression scheme, the entire information on the edges of $\widehat{H}(G'_j)$, for every child bag G'_j, can be encoded in $\widetilde{O}(D)$ bits, in total. We also show that the encoding has a nice structure that allows all vertices in G' to locally decode it, reconstruct the graphs $\widehat{H}(G'_j)$, and consequently compute the desired preserver $H(G')$.

Once the labels are computed, the single-source reachability is computed within extra $\widetilde{O}(D)$ rounds, by sending the label of s to all the vertices.

Strongly Connected Components (SCCs) Identification. The most challenging reachability problem studied in this paper concerns the identification of the strongly connected components in G. The main obstacle for identifying these components in a divide-and-conquer based approach is rooted in the fact that the SCC of G are not necessarily extensions of the SCC of its child components G'. That is, it might be the case that $u, v \in G'$ do not belong to the same SCC in G', although they belong to the same SCC in G. Our approach for solving this problem in the nearly optimal time of $\widetilde{O}(D)$ rounds is based on the following steps. First, for every bag G' the vertices compute the reachability preservers $H(G')$ that describe the reachability in G' between all pairs of vertices in $\text{sep}^+(G')$. Next, we use the multi-source reachability algorithm to compute the $\{u\} \times \text{sep}(G')$ reachability for every vertex u and every bag G' that computes u. The separator vertices $\text{sep}(G')$ play the role of the multiple sources S. We will use here the fact that the $\text{sep}(G')$ vertices lie on a face (in fact, an almost-face) in both[6] G'_{in} and G'_{out}. This allows us to safely apply the multi-source algorithm w.r.t the vertices of $\text{sep}(G')$ in both G'_{in} and G'_{out}. The output of these two computations can be combined to obtain the reachability from u to the $\text{sep}^+(G')$ vertices in G'. The algorithm

[5] For a directed graph H and a vertex u, the incoming (outgoing) neighbors of u are all vertices v such that $(v, u) \in H$ (resp., $(u, v) \in H$).

[6] Recall that G'_{in} and G'_{out} are the subgraphs that embedded in the interior (reps., exterior) of the cycle separator $\text{sep}(G')$.

then applies a top-down recursive procedure whose goal is to let each vertex $u \in G'$ to learn the reachability relations in G, rather than in G'. That is, at the end of this procedure, each vertex $u \in G'$ learns (i) a reachability preserver for the $\text{sep}^+(G') \times \text{sep}^+(G')$ reachability in G; and (ii) its reachability in G to the $\text{sep}(G')$ vertices.

1.3 Preliminaries and Tools

Graph Notation. For a graph $G = (V, E)$ and a subset of vertices $X \subseteq V$, let $G[X]$ be the induced subgraph on X. Throughout, we assume that the communication graph $G = (V, E)$ is connected (in the undirected sense) and has (undirected) diameter D. Otherwise, our algorithms can be applied on each (undirected) connected component of G. For a directed graph $G = (V, E)$, a subgraph $G' \subseteq G$ and $u, v \in V$, we say that $u \preceq_{G'} v$ if there is a directed path from u to v in G'. We denote by $u \rightsquigarrow v$ a directed path from u to v. The set of vertices that have a directed path to a vertex u will be denoted by the incoming vertices to u. The outgoing vertices of u are defined in an analogous manner. Let $N_{in}(u, G), N_{out}(u, G)$ be the set of incoming (resp., outgoing) neighbors of u in G. When G is clear form the context, we may simply write $N_{in}(u)$ and $N_{out}(u)$. Throughout, we use the notion of combinatorial embedding where each vertex knows the clockwise orientation of its neighbors. This embedding can be computed in $\widetilde{O}(D)$ rounds [13].

▶ **Definition 5** (Reachability Preservers). *Given a directed graph $G = (V, E)$, a subgraph G' and $S \subseteq V$, the* reachability preserver *for S in the subgraph G', denoted by $H(S, G')$, is a graph with a vertex set S that captures the $S \times S$ reachability in G'. That is, $V(H(S, G')) = S$ and for every $u, v \in S \cap G'$, $(u, v) \in H(S, G')$ iff $u \preceq_{G'} v$.*

▶ **Definition 6** (Almost-Faces). *For a planar graph $G = (V, E)$, an* almost-face *is a subset of edges E' in G such that there exists a planar graph $G' = (V, E \cup E'')$ where $E' \cup E''$ is a face in G'. That is, a set of edges E' is almost-face if one can add edges to the planar graph between the endpoints of E' to make it a face.*

We will use the fact that an almost-face has either an empty interior or exterior in the planar embedding. Our reachability algorithms make an extensive use of several distributed tools designed for *undirected* planar graphs. Interestingly, despite the fact that our goal is to compute reachability information, we will still be using useful procedures for undirected graphs such as the computation of the cycle separator, and the bounded diameter decomposition of the (underlying undirected) input graph.

Distributed Cycle Separators. For a graph $G = (V, E)$, a subset of vertices $S \subseteq V$ is a *balanced separator* if the removal of S breaks G into connected components that are constant factor smaller than the number of vertices in G. For a graph G and a spanning tree $T \subseteq G$, a *balanced cycle separator* S is a cycle formed by two tree-paths $\pi(x, y)$ and $\pi(y, z)$ (where possibly $z = y$) plus an additional edge (x, z) which is not necessarily in G (which we call a virtual edge). This cycle defines two regions in G, the region inside the cycle and the region outside, where the number of vertices in both these regions is bounded by $2n/3$. For biconnected graphs, [16] gave a randomized algorithm for computing a balanced cycle separator in $\widetilde{O}(D)$ rounds. Recently, [29] extended it to general planar graphs (i.e., 1-connected). We therefore have:

▶ **Lemma 7.** *[16, 29] Given a D-diameter planar graph G and a spanning tree $T \subseteq G$, there exists a randomized algorithm that is $\widetilde{O}(D)$ rounds w.h.p. computes a balanced cycle separator that consists of two tree paths of T plus one additional edge (which might be not in G).*

Bounded Diameter Decomposition. For an undirected graph G of diameter D, *a bounded diameter decomposition* is a recursive balanced decomposition of the vertices into subgraphs of bounded diameter. The decomposition is represented by a tree $\mathcal{T}$, whose vertices, denoted as *bags*[7], correspond to subgraphs in G. This decomposition should satisfy two crucial properties. First, the diameter of each bag is bounded by $O(D \log n)$ which enables the computation of the $O(D \log n)$-size cycle separator in a recursive manner. The second property is that each edge e belongs to at most two subgraphs in each recursion level. This allows one to work on all subgraphs of the same level simultaneously. Each bag G' in the decomposition has a topological closure $\overline{\mathcal{O}(G')}$ that might contain only vertices that appear on the separator of the ancestor bags of G' in the BDD tree $\mathcal{T}$. An additional useful property of the BDD that will be heavily exploited by our algorithms is that each vertex of G' appears on at most three child bags of G'.

We will use the following notation for a BDD tree $\mathcal{T}$. Let $anc(G')$ be the set of ancestor bags of G' on $\mathcal{T}$, and let $\text{sep}(G')$ be the cycle separator of G' obtained by the algorithm of Lemma 7. We slightly override notation by sometimes treating $\text{sep}(G')$ as the the set of cycle edges, and sometime as the set of separator vertices. Note that one of the edges of the cycle separator might not be in G. We refer to edges not in G by *virtual edges*. These virtual edges can be safely added to the graph without breaking planarity, and are used mainly for analysis purposes. The cycle separator $\text{sep}(G')$ defines two regions interior and exterior to the cycle. In the BDD tree, G' has exactly one *internal* child bag G'_{in} which is embedded in the interior of $\text{sep}(G')$, however G' might have several *external* child bags (i.e., the lie in the exterior of $\text{sep}(G')$). Handling these many *external* child bags leads to several challenges in our arguments. In the full paper we provide a formal definition of the BDD.

▶ **Theorem 8** (Bounded diameter decomposition for planar graphs, [29]). *Let $G = (V, E)$ be an unweighted planar graph with diameter D. There is a randomized distributed algorithm that w.h.p. computes the recursive partitioning of G represented by a tree $\mathcal{T}$ of height $O(\log n)$ within $\widetilde{O}(D)$ rounds. In particular, every bag $X \in V_{\mathcal{T}}$ has a unique ID and every vertex knows the IDs of all the bags that contain it.*

2 Nearly Optimal Reachability in $\widetilde{O}(D)$ Rounds

[29] presented an $\widetilde{O}(D^2)$ round algorithm for computing (exact) distance labels in undirected graphs. Just like our reachability labels, their labels are also based on the labeling scheme of Gavoille et al. [12]. In the algorithm of [29] the labels are computed in a recursive manner, where in each step of the recursion the distance labels of the separator vertices $\text{sep}(G')$ are broadcast over G' for every bag G' in the BDD. Since every label has $\widetilde{O}(D)$ bits and $|\text{sep}(G')| = \widetilde{O}(D)$, this led to a round complexity of $\widetilde{O}(D^2)$ rounds. The key contribution in this section is in providing an algorithm for computing the reachability labels without explicitly sending the labels of the separator vertices. In our algorithm, the separator vertices compress their reachability information into $\widetilde{O}(1)$ bits which will allow the vertices to reconstruct the label information.

For a given bag G' in the BDD tree $\mathcal{T}$, let $anc(G')$ be the ancestor bags of G' in $\mathcal{T}$. Define the *extended separator* vertices of G' by

$$\text{sep}^+(G') = \text{sep}(G') \cup \bigcup_{G'' \in anc(G')} \text{sep}(G'') \ .$$

[7] We note that the BDD construction has *no* relation to the tree decomposition of bounded treewidth graphs. The term bags is used in [29] to denote the vertices of the BDD.

Observe that $|\text{sep}^+(G')| = \widetilde{O}(D)$ since the height of the BDD is logarithmic and the separator of each bag is of size $O(D \log n)$. The key computational step is the computation of the reachability preservers $H(\text{sep}^+(G'), G')$ for every bag G' in the BDD tree (see Def. 5). Using this information, the vertices will be able to compute the reachability labels with no further communication. The structure of this section is as follows. In Sec. 2.1, we provide the basic characterization for compressing the reachability information between separator vertices. Then in Sec. 2.2 we describe the algorithm for computing the labels. The single-source reachability algorithm follows immediately given these labels.

Remark. Note that in the $\text{sep}^+(G')$ definition, we do not restrict the separator vertices $\text{sep}(G'')$ for $G'' \in anc(G')$ to be included in $V(G')$. However, by the definition of $H(\text{sep}^+(G'), G')$, only vertices of $\text{sep}^+(G') \cap V(G')$ are included. We still choose to define it in this way for the purpose of the future sections, in which it will be important to take into account the $\text{sep}^+(G')$ vertices that are not in $V(G')$.

2.1 Distributed Compression of Reachability Information

The efficient computation of the reachability preservers are based on several structural claims that allow the vertices in the extended separator sets to compress their reachability information. The lemmas provided in this section are built upon several properties of the BDD algorithm, and in particular to the way in which the child bags of a given bag G' are defined. While the proof arguments contain the necessary details, it will be recommended for the reader to go through the short description of the BDD algorithm to be provided in the full paper and in [29]. The arguments in this subsections have two parts. In the first part we show that the extended separator set $\text{sep}^+(G')$ lie on a collection of $O(\log n)$ almost-faces in G'. In the second part we show that the reachability information of vertices lying on two faces can be compressed efficiently. Putting these two parts together imply that the compression of the reachability information of the $\text{sep}^+(G') \times \text{sep}^+(G')$ pairs. The next lemma shows that the separator vertices of all the ancestor bags of a bag G' lie on $O(\log n)$ almost-faces in G'. The main challenge arises in the case where the BDD algorithm defines several external child bags throughout the decomposition (i.e., bags that are embedded in the exterior of the cycle separator). Recall that by virtual edge we refer to edges not in G but whose addition to G preserve its planarity. In our algorithms, the vertices know their incident virtual edges and their combinatorial embedding.

▶ **Lemma 9.** *The separator vertices of $anc(G')$ lie on a collection $O(\log n)$ almost-faces in G'. On each such face, the separator vertices lie* consecutively*. In addition each such almost-face has $O(\log n)$ virtual edges (i.e., that are not in G).*

Proof. For every level-j bag G', we will prove by induction on $i \in \{1, \ldots, j\}$ that the separator vertices $S_{[j-1,j-i]}$ of all its ancestor bags of G' in levels $j-1, \ldots, j-i$ lie consecutively on at most i almost-faces of G'.

Base case: For every bag G_1 with a parent bag G_2, we show that the vertices of $\text{sep}(G_2) \cap V(G_1)$ lie consecutively on a single almost-face of G_1 that has at most one virtual edge. Recall that the cycle separator $\text{sep}(G_2)$ is defined by two shortest paths and one additional edge e', possibly not in G. For the sake of showing that the separator vertices lie on an almost-face, we assume w.l.o.g. that $\text{sep}(G_2)$ forms a cycle in G. We now distinguish between two cases. The first case is where $\text{sep}(G_2)$ intersects the topological closure $\overline{\mathcal{O}(G_2)}$ in at most a single segment. In this case, G_2 has exactly two child bags $G_{2,in}$ and $G_{2,out}$ that lie in the interior (resp., exterior) region defined by $\text{sep}(G_2)$. We then have that the vertices of $\text{sep}(G_2)$ define the topological closure of $G_{2,in}$ and an internal face in $G_{2,out}$, so the claim holds.

Next, consider the case where $\text{sep}(G_2)$ intersects with the topological closure $\overline{\mathcal{O}(G_2)}$ in at least two non-consecutive segment. In this case, the BDD algorithm defines several bags in the exterior of the cycle $\text{sep}(G_2)$, one bag per connected region of $\mathcal{O}(G_2) \setminus \overline{\mathcal{O}(G_{2,in})}$. The vertices of $\text{sep}(G_2)$ will appear on the topological closure (and thus on the outer face) of these bags in a consecutive manner. To see this, consider a clockwise walk along the cycle $\text{sep}(G_2)$ and observe that any pair of non-continuous intersection points with the topological closure $\overline{\mathcal{O}(G_2)}$ defines a bag that is connected in $(G_2 \cup \text{sep}(G_2)) \setminus \mathcal{O}(G_{2,in})$. For these bags, it is easy to see that the vertices of $\text{sep}(G_2)$ lie consecutively on their topological closure. Finally, since the cycle $\text{sep}(G_2)$ contains at most one virtual edge, there exists at most one bag that might not be connected in $G_2 \setminus \mathcal{O}(G_{2,in})$ (although connected in $(G_2 \cup \text{sep}(G_2)) \setminus \mathcal{O}(G_{2,in})$ due to the virtual edge on $\text{sep}(G_2)$). The BDD algorithm splits this bag into *two* connected components. This breaks $\text{sep}(G_2)$ into two segments each will be on the topological closure of their bags. The induction base holds.

Inductive Step: Assume that the claim holds for each level-j bag G' w.r.t all the separator vertices of its ancestor bags in levels $j-1, \ldots, j-i$. We will prove the claim for all ancestor bags in levels $j-1, \ldots, j-i-1$. Consider a directed path $G_{j-i-1}, \ldots, G_j$ of length i in the BDD tree, that starts at a level-$(j-i-1)$ bag G_{j-i-1} and ends at a level j bag G_j.

By the induction assumption, we assume that the vertices of $\bigcup_{k=j-i-1}^{j-2} \text{sep}(G_k)$ lie consecutively on at most $i-1$ *almost-faces* $f_1, \ldots, f_{i-1}$ in G_{j-1}. The cycle separator $\text{sep}(G_{j-1})$ might have several types of interaction with each face $f_{j'}$ (e.g., containment, with or without intersection, etc.). See Fig. 1 for an illustration of the below mentioned cases. The simpler case is where $f_{j'}$ is embedded in the interior of the cycle $\text{sep}(G_{j-1})$.

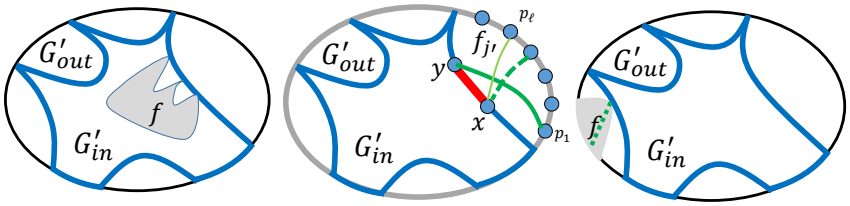

Figure 1 (1) The face f is strictly embedded in the interior of the cycle $\text{sep}(G')$. In case where vertices of f appear also in the external bags of G', it must be that these vertices are contained in $\text{sep}(G')$. (2) $f = \overline{\mathcal{O}(G')}$, that is, all vertices of f are lying on the topological closure. Note that as f is a topological face, it holds that if all vertices of f are on the topological closure then necessarily $f = \overline{\mathcal{O}(G')}$. In the case where one of the regions is not connected in G', the segment of f is split into two consecutive segments, each appearing on the topological closure one child bag. This property follows by a simple path-crossing argument. Let (x, y) be the virtual edge on $\text{sep}(G')$, in this case we will have one bag containing x and the other bag containing y. For example, if p_1 is connected in G' to y, it must also hold that all other vertices $p_{i \geq 1}$ are connected to y. This is because any path from p_i to x would intersect the path connecting p_1 and y. (3) In the last case, f is fully contained in one of the regions enclosed by the topological closure and the cycle separator. The case where the vertices embedded in this region are not connected in G' is further illustrated in Fig. 2.

The induction step then holds for $G_{j-1,in}$ w.r.t the separator vertices on $f_{j'}$. Letting $G_{j-1,out} = G_{j-1} \setminus \mathcal{O}(G_{j-1,in})$, then since $f_{j'}$ is embedded in the interior of the cycle defined by $\text{sep}(G_{j-1})$, we get that $f_{j'} \cap V(G_{j-1,out}) \subset \text{sep}(G_{j-1})$. By the induction assumption, the vertices of $\text{sep}(G_{j-1})$ lie consecutively on the external bags of G_{j-1}. We therefore have that the vertices of $f_{j'} \cup \text{sep}(G_{j-1})$ lie consecutively in the external bags of G_{j-1}. From now on, assume that that $f_{j'}$ is embedded in the exterior of the cycle $\text{sep}(G_{j-1})$.

Case 1: $\mathsf{sep}(G_{j-1})$ and $\overline{\mathcal{O}(G_{j-1})}$ have at most one mutual segment. In this case G_{j-1} has exactly two child bags in the BDD, namely, $G_{j-1,in}$ and $G_{j-1,out}$. Since $\mathrm{sep}(G_{j-1})$ and $f_{j'}$ are cycles, it holds that $f_{j'}$ is either fully contained in $G_{j-1,in}$ or $G_{j-1,out}$. In the case where $f_{j'} \subset G_{j-1,out}$, we have that $f_{j'} \cap G_{j-1,in} \subset \mathrm{sep}(G_{j-1})$ and since $\mathrm{sep}(G_{j-1})$ forms an almost-face in $G_{j-1,in}$ it holds that $(f_{j'} \cup \mathrm{sep}(G_{j-1})) \cap G_{j-1,in}$ forms an almost-face in both $G_{j-1,in}$. The argument works in a symmetric manner in case where $f_{j'} \subset G_{j-1,in}$.

Case 2: $\mathsf{sep}(G_{j-1})$ and $\overline{\mathcal{O}(G_{j-1})}$ have more than two mutual segments. This is the most involved case in the BDD algorithm where an external child bag is defined for each region enclosed by $\mathrm{sep}(G_{j-1})$ and the topological closure $\overline{\mathcal{O}(G_{j-1})}$. There are two sub-cases.

Case 2.1: all the vertices of $f_{j'}$ are embedded on $\overline{\mathcal{O}(G_{j-1})}$. Note that since $f_{j'}$ is an almost-face, it holds that in this case $f_{j'} = \overline{\mathcal{O}(G_{j-1})}$. For example, this case always occur for the interior child in the BDD algorithm whose topological closure is defined by the separator of its parent bag. In this case, the boundary might be split in a consecutive manner to several child bags. See middle figure of Fig. 1. There might be at most one child bag that would be disconnected into two components. We claim that the segment of $f_{j'}$ will be decomposed into two consecutive segments, one per component. To see this, let (x, y) be the virtual edge on $\mathrm{sep}(G_{j-1})$, and let $p_1, \ldots, p_k$ be the vertices on $f_{j'}$ lying on the topological closure from left to right, also in this ordering x appears to the left of y. Observe that if p_1 is in the component of y then all other vertices p_i for $i \geq 2$ must be in the component of y, as otherwise their path to x must intersect with the path connecting y and p_1. See Fig. 1(2). Let ℓ be the maximum index such that p_ℓ is in the component of x. By the same argument (of crossing paths) it holds that all vertices p_i for $i \in \{1, \ldots, \ell\}$ must be in the component of x as well. Thus the segment of $f_{j'}$ is cut into two consecutive segments appearing on the topological boundary of two child bags.

Case 2.2: $f_{j'}$ has at least one vertex that is not on $\overline{\mathcal{O}(G_{j-1})}$. In this case, as $f_{j'}$ is an almost-face, $f_{j'}$ is fully embedded in one of the regions enclosed by the topological boundary $\overline{\mathcal{O}(G_{j-1})}$ and the cycle separator $\mathrm{sep}(G_{j-1})$. See Fig. 1(1,3) and Fig. 2 for an illustration. Since $\mathrm{sep}(G_{j-1})$ contains at most one virtual edge, there exists at most one region, such that the set of vertices embedded in this region are not connected in G_{j-1} (although connected in $G_{j-1} \cup \mathrm{sep}(G_{j-1})$ due to the virtual edge). The BDD algorithm splits this region into two regions such that all vertices embedded in those regions are now connected in G. Specifically, letting (x, y) be the virtual edge on $\mathrm{sep}(G_{j-1})$, then one region is connected in G to x and other is connected to y. We next claim that by the Monge-like property, the separator vertices that by induction assumption lie consecutively the face $f_{j'}$ are divided into at most two consecutive segments. Note that if $f_{j'}$ contains at most one virtual edge, then all vertices in $f_{j'}$ are connected in G_{j-1}, and thus fully contained in one of the external child bags. So it remains to consider the case where $f_{j'}$ contains at least two virtual edges. Let $\sigma_1, \ldots, \sigma_\ell$ be the segments of $f_{j'}$ ordered in a clockwise manner obtained by removing the virtual edges on that face. By the Monge property it then holds that for any pair of non-neighboring segments σ_j and σ_{j+k} that are connected to x, it must hold that all the vertices on the intermediate segments $\sigma_{j+1}, \ldots, \sigma_{j+k-1}$ are connected to x as well. We then have that the vertices of f'_j can be divided into two segments $[x_1, x_2]$ and $[y_1, y_2]$, where all vertices on the segment $[x_1, x_2]$ (resp., $[y_1, y_2]$) are connected to x (resp., y) and $(x_1, y_1), (x_2, y_2)$ are virtual edge on f'_j. We can then add a virtual edge (x_1, x_2) and (y_1, y_2) and omit the virtual edges $(x_1, y_1), (x_2, y_2)$. This defines two new

almost-faces $f_{j'_1}$ and $f_{j'_2}$ that cover all the vertices on $f_{j'}$, each is fully contained in a unique external child bag of G_{j-1}. (Note that one of the virtual edges $(x_1, y_1), (x_2, y_2)$ might form a multiple edge in case where $[y_1, y_2]$ is simply an edge.) See Fig. 2 for an illustration.

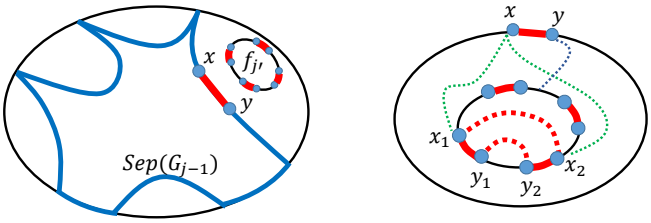

Figure 2 Left: The subgraph G_{j-1}, the topological closure of G_{j-1}, namely, $\overline{\mathcal{O}(G_{j-1})}$ is shown in black, and the cycle separator $\text{sep}(G_{j-1})$ is in blue. The face $f_{j'}$ is fully contained in a region whose induced subgraph (i.e., on the vertices embedded in this region) is not connected in G_{j-1}. Right: Zoom into this region. The red edges indicate virtual edges. The virtual edge (x, y) lies on the boundary of a connected region in $\mathcal{O}(G_{j-1}) \setminus \overline{\mathcal{O}(G_{j-1,in})}$. The removal of (x, y) partitions the vertices embedded in this region into two components, one rooted at x and the other rooted at y. By the Monge property the set of vertices on the face $f_{j'}$ that are connected to x lies on a consecutive segment of the face. This allows us to re-define two almost-faces each will be fully contained in a unique child bag of G_{j-1}.

Finally, it is easy to see by induction on i that each level-i bag contains at most i virtual edges, since each separator $\text{sep}(G'_{j'})$ adds at most one virtual edge to its descendent bags in the BDD tree. Also, since the face split (as described above) occurs only when the almost-face has at least two virtual edges, the number of virtual edges is kept. ◀

Monge-Like Properties of Reachability between Faces. Our compression scheme of the reachability information is based on the well known Monge properties. We begin with encoding the reachability relations between disjoint sets of vertices on a single face, then encoding the reachability between all-pairs on a face, and finally encoding the reachability between vertices on two faces. The next auxiliary lemmas are modifications of Lemmas 2.1, 2.2 and 2.4 in [2]. In [2] the goal was to encode distances in unweighted graphs, whereas here our goal is to encode reachability information. This leads to small modifications both in the statements of the lemmas as well as in the arguments.

▶ **Lemma 10.** *[Analogue of Lemma 2.1, [2]] Let $C = (v_1, v_2, \ldots, v_{|C|})$ be the cyclic walk of a face partitioned into two parts $C_1 = (v_1, v_2, \ldots, v_\ell)$ and $C_2 = (v_{\ell+1}, v_{\ell+2}, \ldots, v_{|C|})$. Then, for any subsets $C'_1 \subseteq C_1$ and $C'_2 \subseteq C_2$ the reachability relations between $C'_1 \times C'_2$ can be encoded using $\widetilde{O}(|C'_1| + |C'_2|)$ bits.*

▶ **Lemma 11.** *[Analogue of Lemmas 2.2, [2]] Let $C = (v_1, \ldots, v_{|C|}\}$ be the cycle walk of a face of a planar graph. Then, all reachability relations for a subset $C' = (v_{q_1}, \ldots, v_{q_s})$ can be encoded in $\widetilde{O}(|C'|)$ bits.*

▶ **Lemma 12.** *[Analogue of Lemmas 2.4 in [2]] Let $C_{in} = (v_1, \ldots, v_q)$ and $C_{ext} = (u_1, \ldots, u_\ell)$ be the cyclic walk of two faces of a planar graph. Then, the reachability relations between any subsets $C'_{in} \subseteq C_{in}$ and $C'_{ext} \subseteq C_{ext}$ can be encoded with $\widetilde{O}(|C'_{in}| + |C'_{ext}|)$ bits*[8].

[8] In Lemma 2.3 of [2], it is required that C'_{ext} is a prefix of C_{ext}. For our purposes, as we care for

The algorithmic implication: We are next ready to state the algorithmic implication of the structural properties provided above, which will be extensively used in our algorithms. We need the following notation. Let u be a vertex and f be a face in G'. Let $V(f)$ be the set of vertices on the boundary of f. A vertex $u \in G'$ is *incoming-active* w.r.t f if there exists some vertex $s \in V(f) \cap \text{sep}^+(G')$ satisfying that $s \preceq_{G'} u$. Otherwise, the vertex is *incoming-inactive*. The notions of outgoing-active and outgoing-inactive are defined accordingly. For two faces f_1 and f_2 in a subgraph G', let $A_{in}(f_1, f_2, G')$ be the set of *incoming-active vertices* in $\text{sep}^+(G') \cap V(f_1)$ w.r.t. f_2 formally defined as follow:

$$A_{in}(f_1, f_2, G') = \{u \in \text{sep}^+(G') \cap V(f_1) \mid \exists v \in \text{sep}^+(G') \cap V(f_2) \text{ satisfying } v \preceq_{G'} u\} . \quad (1)$$

The set of outgoing-active vertices $A_{out}(f_1, f_2, G')$ is defined analogously. These definitions are analougoes when the faces are almost-faces. Throughout, because of the symmetry between computing the incoming vertices and the outgoing vertex, w.l.o.g., when saying active vertices we mean outgoing-active vertices.

▶ **Lemma 13.** *Let G' be bag with a child bag G'_j in the BDD tree, and let $f'_{1,j}, \ldots, f'_{k,j}$ be the almost-faces of G'_j on which the vertices of $\text{sep}^+(G') \cap V(G'_j)$ are lying. Assume that all vertices of G' know: (1) the ordering of the $\text{sep}^+(G') \cap V(G'_j)$ vertices on the $f'_{i,j}$ faces, (2) the set of incoming-active $A_{in}(f'_{i_1,j}, f'_{i_2,j}, G'_j)$ and outgoing-active $A_{out}(f'_{i_1,j}, f'_{i_2,j}, G'_j)$ vertices for every pair of almost-faces $f'_{i_1,j}, f'_{i_2,j}$, $i_1, i_2 \in \{1, \ldots, k\}$. Then, every vertex $u \in f'_{i_2,j}$ can encode the set of all its outgoing vertices from $\text{sep}^+(G') \cap V(f'_{i_1,j})$ using $\widetilde{O}(1)$ bits.*

2.2 The Single Source Reachability Algorithm

The reachability algorithm has two phases. The first phase computes, in a top-bottom manner, a reachability preserver $H(G') = H(\text{sep}^+(G'), G')$ for every bag G' where $(u, v) \in H(G')$ iff $u \preceq_{G'} v$, for every $u, v \in \text{sep}^+(G')$. At the end of this phase, each vertex u knows the preserver $H(G')$ for every bag G' containing u. The second phase is performed locally at each vertex, and does not require any communication. Each vertex u will compute in a top-down manner a reachability label $L_G(u)$ of $\widetilde{O}(D)$ bits, such that given a label $L_G(v)$ it can determine the u-v reachability in G. Thus, by letting the source vertex s send its reachability label, all vertices can determine their reachability to s.

Step 1: Computing the Reachability Preservers

The reachability preservers are computed in a **bottom-up manner** on the BDD tree of depth $d = O(\log n)$. Starting from the leaf level, in every independent level $i \in \{1, \ldots, d\}$ of the recursion, the goal is to compute the reachability preserver $H(G') = H(\text{sep}^+(G'), G')$ for a level-$(d - i + 1)$ bag G' given the reachability preservers of its child bags. Note that the reachability preserver $H(G')$ of each bag is computed based on the reachability in G' rather than G. Our observations for every bag G' and its child bags $G'_1, \ldots, G'_k$, are as follows:

▶ **Observation 14.** *(1) $\text{sep}^+(G') \subseteq \bigcup_{j=1}^{\ell} \text{sep}^+(G'_j)$. (2) For every $s_1, s_2 \in \text{sep}^+(G')$ such that $s_1 \preceq_{G'} s_2$ it holds that $s_1 \preceq_{\bigcup_j \widehat{H}(G'_j)} s_2$ where $\widehat{H}(G'_j) = \{(u, v) \in H(G'_j) \mid u, v \in \text{sep}^+(G') \cap V(G'_j)\}$, i.e., $\widehat{H}(G'_j) = H(G'_j)[\text{sep}^+(G')]$.*

reachability rather than distances, C'_{ext} can be an arbitrary subset of C_{ext}.

Proof. (1) follows by the definition of the set $\text{sep}^+(\cdot)$ sets. To see (2), it is sufficient to prove that for every $s_1, s_2 \in \text{sep}^+(G')$ such that $s_1 \preceq_{G'} s_2$ but $(s_1, s_2) \notin \bigcup_j \widehat{H}(G'_j)$, it holds that $s_1 \preceq_{\bigcup_j \widehat{H}(G'_j)} s_2$.

Let P be the directed path from s_1 to s_2 in G'. Since $\bigcup_j G'_j = G'$, the path P can be decomposed into sub-paths $P = P_1 \circ P_2 \ldots \circ P_\ell$ such that endpoints of all these paths P_k are in $\text{sep}^+(G')$ (in fact, for $k \in \{2, \ldots, \ell - 1\}$ both endpoints are in $\text{sep}(G')$). In addition, letting $s_{1,k}, s_{2,k}$ be the endpoint of P_k, then there exists a child bag G'_{j_k} such that $P_k \subseteq G'_{j_k}$. Therefore, $(s_{1,k}, s_{2,k}) \in \widehat{H}(G'_{j_k})$ for every $k \in \{1, \ldots, \ell\}$. Concluding that $s_1 \preceq_{\bigcup_j \widehat{H}(G'_j)} s_2$. ◀

For a child bag G'_j of G', let $f_{1,j}, \ldots, f_{\ell,j}$ be the almost-faces of G'_j on which the vertices of $\text{sep}^+(G') \cap V(G'_j)$ are embedded. The vertices of G' are then required to know the following for every bag G'_j for $j \in \{1, \ldots, k\}$:

- (I1) The orientation of the vertices of $\text{sep}^+(G') \cap V(G'_j)$ on the almost-faces $f_{1,j}, \ldots, f_{\ell,j}$.
- (I2) The active incoming and outgoing sets $A_{in}(f_{i_1,j}, f_{i_2,j}, G'_j)$ and $A_{out}(f_{i_1,j}, f_{i_2,j}, G'_j)$ for every $i_1, i_2 \in \{1, \ldots, \ell\}$ (see Eq. (1) to recall the definition of these sets).

By Lemma 13, each vertex $u \in \text{sep}^+(G') \cap V(G'_j)$ can compress its $\{u\} \times \text{sep}^+(G')$ reachability in the G'_j subgraph using $\widetilde{O}(1)$ bits. As each vertex in G' appears on at most three child bags, all the vertices of $\text{sep}^+(G')$ can send the compression of their $\text{sep}^+(G')$ reachability in every child bag G'_j within $\widetilde{O}(D)$ rounds. As a result, all vertices of G' know the graph $\widehat{H}(G'_j) = H(G'_j)[\text{sep}^+(G')]$ for every child bag G'_j. Finally, each vertex u computes the desired preserver $H(G')$ by locally computing $\text{sep}^+(G') \times \text{sep}^+(G')$ reachability in the union of graphs $\bigcup_j \widehat{H}(G'_j)$. The correctness of this step follows by Obs. 14(2). This completes the description of the algorithm. We next show that all vertices in G' can learn the required information (I1) and (I2) within $\widetilde{O}(D)$ rounds.

▷ Claim 15. (I1, I2) can be obtained in $\widetilde{O}(D)$ rounds;

Proof. By Lemma 9, the vertices of $\text{sep}^+(G')$ lie *consecutively* on $O(\log n)$ almost-faces in G'_j. Note by the proof of Lemma 9, the vertices also know their incident virtual edges on these faces. Each vertex $u \in \text{sep}^+(G') \cap V(G'_j)$ simply sends its edges on these faces. Since each vertex belongs to at most three child bags, and since there are $O(\log n)$ faces, each vertex in $\text{sep}^+(G')$ sends $\widetilde{O}(1)$ bits. Since the vertices of $\text{sep}^+(G')$ lie consecutively on those faces, by knowing all their edges on the faces, the vertices can decide a fixed orientation (i.e., increasing clock-wise ordering from the largest vertex ID. To see (I2), since all vertices of G'_j know the subgraph $H(G'_j)$, and as $\text{sep}^+(G'_j) \subseteq \text{sep}^+(G')$, they know their incoming and outgoing vertices from $\text{sep}^+(G')$ in G'_j. In addition, all vertices know (I1), that is, a fixed ordering of the $\text{sep}^+(G')$ vertices on $O(\log n)$ faces. Therefore each vertex $u \in f'_{i_1,j}$ can send a message containing the indicators of all other faces $f'_{i_2,j}$ for which $u \in A_{in}(f_{i_1,j}, f_{i_2,j}, G'_j)$ and in the same manner the list of faces for which $u \in A_{out}(f_{i_1,j}, f_{i_2,j}, G'_j)$. This message has $O(\log n)$ bits, and since each u in G' belongs to at most three child bags, the entire information on all $A_{in}(f_{i_1,j}, f_{i_2,j}, G'_j)$ and $A_{out}(f_{i_1,j}, f_{i_2,j}, G'_j)$ can be delivered in $\widetilde{O}(D)$ rounds. ◁

We therefore have the following.

▶ **Lemma 16.** *Given a BDD tree $\mathcal{T}$ for a planar graph G, there exists a randomized algorithm that computes for each bag $G' \in \mathcal{T}$ the reachability preserver $H(G') = H(\text{sep}^+(G'), G')$, where each vertex u knows the preserver edges $H(G')$ for every bag G' containing u.*

Step 2: (Locally) Computing the Reachability Labels. We next show that given that each vertex u knows the reachability preservers $H(G') = H(\text{sep}^+(G'), G')$ for every bag G' that contains u, it can *locally* compute its reachability label $L_G(u)$ of $\widetilde{O}(D)$ bits, without any additional communication. We follow the same recursive scheme of the labels by [12, 29] with one key difference as will be explained soon. The label $L_G(u)$ of each vertex $u \notin \text{sep}(G)$ consists of three fields: (i) lists of incoming and outgoing vertices to v from $\text{sep}^+(G)$: $In(u, G) = \{v \in \text{sep}^+(G) \mid v \preceq_G u\}$ and $Out(u, G) = \{v \in \text{sep}^+(G) \mid u \preceq_G v\}$, (ii) the identifier of G's child bag that contains u, and (iii) the (recursive) label $L_{G'}(u)$ where G' is the child component of G that contains u in the BDD decomposition. Since $u \notin \text{sep}(G)$, there exists exactly one such bag. Denoting the first two fields in the label by $\widehat{L}_G(u)$, the label u consists of $k = O(\log n)$ sub-labels $L_G(u) = \widehat{L}_{G_0}(u) \circ \widehat{L}_{G_1}(u) \ldots \circ \widehat{L}_{G_k}(u)$. For $u \in \text{sep}(G)$ the label $L_G(u)$ contains only the first field (i). The key difference compared to [12, 29] is that each vertex u keeps the information in each sub-label $\widehat{L}_{G'}(u)$ with respect to the extended-separator set $\text{sep}^+(G')$ rather than the separator set $\text{sep}(G')$. As will see, this adaptation is critical for our scheme to go through.

We focus on a vertex u and explain how it can compute its label in a bottom-up manner on the BDD tree. To avoid cumbersome notation for a leaf bag G', let $\text{sep}(G') = V(G')$. Let ℓ be the minimum value $i \in \{0, \ldots, d\}$ satisfying that $u \in \text{sep}(G')$ for some i-level bag G'. Note that by definition of the BDD, u belongs to exactly one bag in each of the levels $i \in \{0, \ldots, \ell\}$. Let $G = G_0, G_1, \ldots, G_\ell$ be the unique bags that contain u in level $i \in \{1, \ldots, \ell\}$.

We start with the base case of computing the label $L_{G_\ell}(u)$. There are two cases. If $\ell = d$, then G_d is a leaf bag, and u knows the entire leaf bag information from the graph $H(G_d)$. Otherwise, if $\ell \leq d - 1$, then $u \in \text{sep}(G_\ell)$. In this case, the label $L_{G_\ell}(u)$ should contain the lists $In(u, G_\ell)$ and $Out(u, G_\ell)$ which can be obtained directly from the preserver $H(G_\ell)$. Assume that u has locally computed the labels $L_{G_\ell}(u), \ldots, L_{G_{i+1}}(u)$. We now explain how it can compute the label $L_{G_i}(u)$. Observe that the label $L_{G_i}(u)$ has the form $L_{G_i}(u) = \widehat{L}_{G_i}(v) \circ L_{G_{i+1}}(v)$. Therefore, it is sufficient to compute the sets $In(u, G_i)$ and $Out(u, G_i)$. By the sub-label $\widehat{L}_{G_{i+1}}(u)$ of the $L_{G_{i+1}}(u)$ label, u knows the sets $In(u, G_{i+1}), Out(u, G_{i+1})$. The sets $In(u, G_i), Out(u, G_i)$ are then obtained by locally computing the reachability of u w.r.t the $\text{sep}(G_i)$ vertices in the graph

$$H(G_i, u) = H(\text{sep}^+(G_i), G_i) \cup \{(u, v) \mid v \in Out(u, G_{i+1})\} \cup \{(v, u) \mid v \in In(u, G_{i+1})\} .$$

We next show that for every $v \in \text{sep}^+(G_i)$ such that $v \preceq_{G_i} u$, it holds that $v \preceq_{H(G_i,u)} u$, and thus $v \in In(u, G_i)$ as desired. The same argument will apply to the $Out(u, G_i)$ set. Let P be a directed $v \rightsquigarrow u$ path in G_i. If $P \subseteq G_{i+1}$, then since $v \in \text{sep}^+(G_{i+1})$, by induction assumption, $v \in In(u, G_{i+1})$ and thus $(v, u) \in H(G_i, u)$. Otherwise, P must contain at least one vertex from $\text{sep}(G_i)$. Let s^* be the last vertex (closest to u) from $\text{sep}(G_i)$ on P. Then, by the selection of s^*, $P[s^*, u] \subset G_{i+1}$. By induction assumption, as $s^* \in \text{sep}^+(G_{i+1})$, it holds that $s^* \in In(u, G_{i+1})$ and thus $(s^*, u) \in H(G_i, u)$. In addition, since $v, s^* \in \text{sep}^+(G_i)$, we have that $(v, s^*) \in H(G_i)$. Overall, there is a directed path $(v, s^*) \circ (s^*, u)$ in $H(G_i, u)$, and thus $v \in In(u, G_i)$. The claim follows. This completes the $\widetilde{O}(D)$-round algorithm for computing the the reachability labels. By broadcasting the reachability label of the source vertex s, we also solve the single-source reachability problem, and establish Thm. 1. The multi-source reachability algorithm and the algorithms for weighted digraphs are deferred to the full version.

References

1 Amir Abboud, Vincent Cohen-Addad, and Philip N Klein. New hardness results for planar graph problems in p and an algorithm for sparsest cut. In *Proc. of the Symp. on Theory of Comp. (STOC)*, 2020. URL: `http://www.wisdom.weizmann.ac.il/~robi/Bertinoro2019_FineGrained/program/program-bertinoro19.html#Cohen-Addad`.

2 Amir Abboud, Pawel Gawrychowski, Shay Mozes, and Oren Weimann. Near-optimal compression for the planar graph metric. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 530–549. SIAM, 2018.

3 Glencora Borradaile, Philip N Klein, Shay Mozes, Yahav Nussbaum, and Christian Wulff-Nilsen. Multiple-source multiple-sink maximum flow in directed planar graphs in near-linear time. *SIAM Journal on Computing*, 46(4):1280–1303, 2017.

4 Sergio Cabello. Subquadratic algorithms for the diameter and the sum of pairwise distances in planar graphs. *ACM Transactions on Algorithms (TALG)*, 15(2):1–38, 2018.

5 Shiri Chechik and Doron Mukhtar. Reachability and shortest paths in the broadcast CONGEST model. In *33rd International Symposium on Distributed Computing, DISC 2019, October 14-18, 2019, Budapest, Hungary*, pages 11:1–11:13, 2019.

6 Atish Das Sarma, Stephan Holzer, Liah Kor, Amos Korman, Danupon Nanongkai, Gopal Pandurangan, David Peleg, and Roger Wattenhofer. Distributed verification and hardness of distributed approximation. In *Proc. of the Symp. on Theory of Comp. (STOC)*, pages 363–372, 2011.

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

8 Jittat Fakcharoenphol and Satish Rao. Planar graphs, negative weight edges, shortest paths, and near linear time. *Journal of Computer and System Sciences*, 72(5):868–889, 2006.

9 Orr Fischer and Rotem Oshman. A distributed algorithm for directed minimum-weight spanning tree. In *33rd International Symposium on Distributed Computing, DISC 2019, October 14-18, 2019, Budapest, Hungary*, pages 16:1–16:16, 2019.

10 Sebastian Forster and Danupon Nanongkai. A faster distributed single-source shortest paths algorithm. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 686–697. IEEE, 2018.

11 Harold N Gabow. Scaling algorithms for network problems. In *24th Annual Symposium on Foundations of Computer Science (sfcs 1983)*, pages 248–258. IEEE, 1983.

12 Cyril Gavoille, David Peleg, Stéphane Pérennes, and Ran Raz. Distance labeling in graphs. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 210–219. Society for Industrial and Applied Mathematics, 2001.

13 Mohsen Ghaffari and Bernhard Haeupler. Distributed algorithms for planar networks i: Planar embedding. In *the Proc. of the Int'l Symp. on Princ. of Dist. Comp. (PODC)*, pages 29–38, 2016.

14 Mohsen Ghaffari and Bernhard Haeupler. Distributed algorithms for planar networks ii: Low-congestion shortcuts, mst, and min-cut. In *Proc. of ACM-SIAM Symp. on Disc. Alg. (SODA)*, pages 202–219, 2016.

15 Mohsen Ghaffari and Jason Li. Improved distributed algorithms for exact shortest paths. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 431–444, 2018.

16 Mohsen Ghaffari and Merav Parter. Near-optimal distributed DFS in planar graphs. In *31st International Symposium on Distributed Computing, DISC 2017, October 16-20, 2017, Vienna, Austria*, pages 21:1–21:16, 2017. URL: `http://www.weizmann.ac.il/math/parter/sites/math.parter/files/uploads/planarDFS_DISC17.pdf`.

17 Mohsen Ghaffari and Rajan Udwani. Brief announcement: Distributed single-source reachability. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, PODC 2015, Donostia-San Sebastián, Spain, July 21 - 23, 2015*, pages 163–165, 2015.

18 Bernhard Haeupler, D. Ellis Hershkowitz, and David Wajc. Round- and message-optimal distributed graph algorithms. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, PODC 2018, Egham, United Kingdom, July 23-27, 2018*, pages 119–128, 2018.

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

20 Bernhard Haeupler, Taisuke Izumi, and Goran Zuzic. Near-optimal low-congestion shortcuts on bounded parameter graphs. In *International Symposium on Distributed Computing*, pages 158–172. Springer, 2016.

21 Bernhard Haeupler, Jason Li, and Goran Zuzic. Minor excluded network families admit fast distributed algorithms. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, PODC 2018, Egham, United Kingdom, July 23-27, 2018*, pages 465–474, 2018.

22 Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. A deterministic almost-tight distributed algorithm for approximating single-source shortest paths. In *Proc. of the Symp. on Theory of Comp. (STOC)*, pages 489–498, 2016.

23 Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. A deterministic almost-tight distributed algorithm for approximating single-source shortest paths. *SIAM Journal on Computing*, STOC16:98–137, 2019. doi:10.1137/16M1097808.

24 Giuseppe F Italiano, Yahav Nussbaum, Piotr Sankowski, and Christian Wulff-Nilsen. Improved algorithms for min cut and max flow in undirected planar graphs. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 313–322, 2011.

25 Arun Jambulapati, Yang P Liu, and Aaron Sidford. Parallel reachability in almost linear work and square root depth. *arXiv preprint arXiv:1905.08841*, 2019.

26 Philip N Klein and Sairam Subramanian. A randomized parallel algorithm for single-source shortest paths. *Journal of Algorithms*, 25(2):205–220, 1997.

27 Christoph Lenzen and Boaz Patt-Shamir. Fast partial distance estimation and applications. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*, pages 153–162, 2015.

28 Jason Li. Distributed treewidth computation. *arXiv preprint arXiv:1805.10708*, 2018.

29 Jason Li and Merav Parter. Planar diameter via metric compression. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019.*, pages 152–163, 2019.

30 Richard J Lipton and Robert Endre Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979.

31 Shay Mozes and Christian Sommer. Exact distance oracles for planar graphs. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete algorithms*, pages 209–222. SIAM, 2012.

32 Shay Mozes and Christian Wulff-Nilsen. Shortest paths in planar graphs with real lengths in o (nlog 2 n/loglogn) time. In *European Symposium on Algorithms*, pages 206–217. Springer, 2010.

33 Danupon Nanongkai. Distributed approximation algorithms for weighted shortest paths. In *Proc. of the Symp. on Theory of Comp. (STOC)*, 2014.

34 David Peleg. *Distributed Computing: A Locality-sensitive Approach*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

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

36 Mikkel Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *Journal of the ACM (JACM)*, 51(6):993–1024, 2004.