$\tilde{\text{O}}$ptimal Fault-Tolerant Labeling for Reachability and Approximate Distances in Directed Planar Graphs[*]

Itai Boneh[1,2], Shiri Chechik[3], Shay Golan[1,2], Shay Mozes[1], and Oren Weimann[2]

[1]Reichman University, Israel
[2]University of Haifa, Israel
[2]Tel-Aviv University, Israel

Abstract

We present a labeling scheme that assigns labels of size $\tilde{O}(1)$ to the vertices of a directed weighted planar graph G, such that for any fixed $\varepsilon > 0$ from the labels of any three vertices s, t and f one can determine in $\tilde{O}(1)$ time a $(1+\varepsilon)$-approximation of the s-to-t distance in the graph $G \setminus \{f\}$. For approximate distance queries, prior to our work, no efficient solution existed, not even in the centralized oracle setting. Even for the easier case of reachability, $\tilde{O}(1)$ queries were known only with a centralized oracle of size $\tilde{O}(n)$ [SODA 21].

[*]This research was partially funded by Israel Science Foundation grant 810/21.
The second author was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant Agreement No. 803118, UncertainENV).

Contents

1 Introduction

In network optimization, computing distances is essential for applications such as routing in transportation, communication, and logistics. Real-world networks often face temporary inaccessibility of nodes or edges due to maintenance, damage, or congestion, necessitating algorithms that can efficiently report distances in dynamic conditions. A *distance oracle* is a data structure that can report the distance between any two vertices of a given graph, with the objective of achieving an efficient tradeoff between time and space, ideally approaching constant query time and linear space. Expanding on this, *distance labeling schemes* assign compact labels to vertices, allowing queries based solely on these labels, which is especially useful in distributed systems where local information alone determines distances or reachability. Research on labeling schemes spans various graph properties, including adjacency [KNR92, ADK17, PRSWN16, AKTZ19, AN17, BGL07], distances [GPPR04, BCG+22, Tho04, ACG12, GKK+01], connectivity [KKKP04, HL09, Kor10], and Steiner trees [Pel05]. See [Rot16] for a survey. Similarly, *reachability oracles* and *reachability labeling schemes* aim to answer reachability queries rather than distances, also benefiting distributed applications. The resilience of real-world networks to failures has led to research on robust data structures capable of accommodating disruptions.

In this paper, we focus on labeling schemes for approximate distances and reachability in directed edge-weighted planar graphs in the presence of a single vertex failure, also referred to as a *fault-tolerant* approximate distance (reachability) labeling scheme. The objective is to assign a compact label to each vertex, such that given the labels of any three vertices s, t, f, one can efficiently approximate the distance between s and t in the graph $G \setminus \{f\}$, or determine if t is reachable from s in the graph $G \setminus \{f\}$. Fault-tolerant labeling schemes (also called forbidden-set labeling schemes) have been extensively studied for various problems, such as connectivity, distances, and routing, and across multiple graph families (see, e.g., [CGK09, CT10, FKMS07, Twi06, ACG12, BCG+22, ACGP16, Rot16]). Let us first review the most relevant related work. We focus on the directed case, as it is the focus of this paper and is generally more challenging than the undirected case. In many instances, techniques developed for undirected graphs do not extend to directed graphs.

Exact distance oracles and labeling for directed planar graphs. The problem of exact distance oracles for directed planar graphs has been extensively studied over the past few decades [ACC+96, Dji96, CX00, FR06, Kle05, WN10, Nus11, Cab12, MS12, CDW17, GMWWN18, CGMW19, CGL+23]. Notably, recent advances have led to very strong solutions [CGL+23] that achieve almost optimal $n^{1+o(1)}$ space and near-optimal $\tilde{O}(1)$ query time. This result is particularly interesting because it reveals a significant gap between oracles and labeling schemes. Specifically, it was shown in [GPPR04] that exact distance labeling for planar graphs requires polynomial-sized labels of size $\Omega(\sqrt{n})$, regardless of the query time (and there is a known tight upper bound of $O(\sqrt{n})$ [GPPR04, GU23]).

Exact distance oracles and labeling for directed planar graphs with failures. Exact distance oracles for directed planar graphs in the presence of failures have bounds that are not as favorable compared to those without failures. [BLM12] introduced a single-source fault-tolerant distance oracle with near-optimal $\tilde{O}(n)$ space and $\tilde{O}(1)$ query time. They further extended their construction to handle the all-pairs variant of the problem, resulting in increased space of $\tilde{O}(n^{1.5})$ and query time of $\tilde{O}(\sqrt{n})$. Subsequently, [CMT19] presented an improved fault-tolerant distance oracle for the all-pairs version in planar graphs. Their oracle accommodates multiple failures; however, it features a polynomial tradeoff between the oracle's size and query time that may be less advantageous compared to previous constructions. Recently, an exact fault-tolerant distance labeling scheme for planar graphs with label size $\tilde{O}(n^{2/3})$ (accommodating a single failure) was presented in [BCG+22].

Importantly, in the context of the all-pairs version, the fault-tolerant oracles mentioned above have considerably worse bounds compared to the best-known distance oracles without faults for planar graphs.

Approximate distance labeling for directed planar graphs. Since exact distance labels require polynomial-sized labels [GPPR04], researchers have pursued more compact labels that yield *approximate* distances. [GKK+01] studied such approximate labels across general graphs and various graph families. Specifically, for planar graphs, they introduced $O(n^{1/3} \log n)$-bit labels that provide a 3-approximation of distances. In the same year, [GKR01] developed even smaller 3-approximate labels requiring only $O(\log^2 n)$ bits, while Thorup presented $(1+\varepsilon)$-approximate labels of size $O(\log n/\varepsilon)$ for any fixed $\varepsilon > 0$ [Tho04].

Reachability oracles and labeling for directed planar graphs. The reachability question was also very well studied in both general and planar graphs. [HLNW17] provided conditional lower bounds for combinatorial constructions of reachability oracles, showing that no non-trivial combinatorial reachability oracle constructions exist for general directed graphs.[1] Specifically, they proved that it is impossible to design a reachability oracle that simultaneously achieves $O(n^{3-\varepsilon})$ preprocessing time and $O(n^{2-\varepsilon})$ query time, for any $\varepsilon > 0$.

Since non-trivial reachability oracles are not attainable for general graphs, efforts have been directed towards developing improved reachability oracles for specific graph families. Notably, graphs possessing separators of size $s(n)$ admit a straightforward reachability oracle of size $\tilde{O}(n \cdot s(n))$ and query time $\tilde{O}(s(n))$. Consequently, planar graphs (and more extensive graph classes such as H-minor free graphs) admit oracles of size $\tilde{O}(n^{1.5})$ and query time $\tilde{O}(\sqrt{n})$. In a groundbreaking result, Thorup [Tho04] introduced a near-optimal reachability oracle for directed planar graphs with $\tilde{O}(n)$ space and $\tilde{O}(1)$ query time. This result can also be adapted to a labeling scheme with label size $\tilde{O}(1)$. Subsequently, [HRT15] further improved this construction to a truly optimal oracle with $O(n)$ space and $O(1)$ query time.

Fault-tolerant reachability oracles for directed planar graphs. Fault-tolerant reachability oracles have been studied extensively in general graphs, see e.g. [vdBS19, GIP17, Cho16, BCR18, KS99, GGI+17]. In planar graphs, one can leverage the more powerful fault-tolerant *distance* oracles mentioned above [BLM12, CMT19, BCG+22]. However, in the all-pairs version, these oracles have considerably worse bounds when compared to the best known distance oracles without faults for planar graphs. In a groundbreaking development, [IKP21] in SODA 2021 introduced a nearly optimal fault-tolerant reachability oracle for directed planar graphs. Their innovative approach finally achieved near-optimal $\tilde{O}(n)$ size, $\tilde{O}(n)$ construction time, and $\tilde{O}(1)$ query time. It is not known how to turn the oracle of [IKP21] into a fault-tolerant reachability labeling scheme or into a fault-tolerant approximate distance oracle.

Fault-tolerant approximate distance labeling for undirected planar graphs. For undirected planar graphs [ACG12] presented labels of size $\tilde{O}(1)$ that for any fixed $\varepsilon > 0$, from the labels of vertices s, t, and the labels of a set F of failed vertices, can report in $\tilde{O}(|F|^2)$ time a $(1+\varepsilon)$-approximation of the shortest s-to-t path in the graph $G \setminus F$. One would hope to generalize this result to the directed case, even just settling for the seemingly easier task of reachability, and even for a single fault. Unfortunately, it seems this result crucially relies on the graph being undirected.

Remaining research questions. Previously, there was no efficient labeling scheme even just for reachability in directed planar graphs, only an oracle. For reachability in planar graphs, the best

[1]The term combinatorial is often referred to algorithms that do not utilize fast matrix multiplications.

previously known fault-tolerant labeling scheme was the one for fault-tolerant exact distances by [BCG+22], in which the label size is $\tilde{O}(n^{2/3})$. Is this the best possible? Ideally, the goal would be to devise a labeling scheme in which the sum of the label sizes is roughly equal to the size of the state-of-the-art oracle. However, achieving this goal is not always possible. For instance, as mentioned above, in [CGL+23], an almost optimal exact distance oracle is given for directed planar graphs (without faults) of size $O(n^{1+o(1)})$ and query time $\tilde{O}(1)$. On the other hand, it was shown in [GPPR04] that exact distance labels (even without faults) for planar graphs necessitate polynomial-sized labels regardless of the query time. Given this discrepancy between oracles and labeling schemes for distances in planar graphs, a natural question arises: does the same discrepancy exist for fault-tolerant reachability or for approximate distances? In other words, is it possible to design a labeling scheme for directed planar graphs with label size $\tilde{O}(1)$ and query time $\tilde{O}(1)$? Or does a similar gap exist between fault-tolerant reachability oracles and labeling schemes, as in the case of exact distances?

Furthermore, in the case of approximate distances, there is not even an oracle with near-optimal bounds capable of handling a single failure in planar directed graphs. If one wants an approximate distance oracle for directed planar graphs, the best option up to our work is to use an exact fault-tolerant oracle, which is far from the optimal bounds we aim for both in terms of space and query time.

A natural question is whether it is possible to devise an efficient approximate fault-tolerant distance oracle for directed planar graphs with near-optimal bounds of $\tilde{O}(n)$ size and $\tilde{O}(1)$ query time? If the answer to this question is positive, a further question would be whether it is also possible to obtain an approximate fault-tolerant distance labeling scheme with near-optimal $\tilde{O}(1)$ label size. A positive answer to this would also resolve the open question for the simpler case of reachability.

Our results. We answer the above two questions in the affirmative by providing a near optimal fault tolerant approximate distance labeling scheme and reachability in directed planar graphs with $\tilde{O}(1)$ label size and $\tilde{O}(1)$ query time, see Theorem 5.2.

2 Technical Overview

In this section, we first discuss the main challenges in extending the non-faulty reachability labels of Thorup [Tho04] to handle faults. Then, we introduce a high level overview of these labels. Finally, we discuss the challenges in extending our single-fault reachability labels to approximate distance labels, and how we overcome them.

2.1 Challenges in extending [Tho04]

Thorup's non-faulty reachability labeling [Tho04], stores for each vertex s and each relevant path separator P, the first vertex on P that is reachable from s in G and the last vertex of P that can reach s in G, denoted as $\mathsf{first}_\mathsf{G}(s, P)$ and $\mathsf{last}_\mathsf{G}(s, P)$, respectively. To determine if vertex s can reach vertex t by a path that intersects the path separator P, one can simply check if $\mathsf{first}_\mathsf{G}(s, P)$ precedes $\mathsf{last}_\mathsf{G}(t, P)$ on P (denoted as $\mathsf{first}_\mathsf{G}(s, P) \leq_P \mathsf{last}_\mathsf{G}(t, P)$). We call the general idea of reducing s-to-t reachability to finding the first/last reachable vertices on some path P separating s and t as the *'Find the First'* approach.

One of the main challenges we face with applying the 'Find the First' approach is the occurrence of failures anywhere along the relevant path separator P. A faulty vertex f on P requires us to store additional information, including the closest vertex to f that appears after f on P and is reachable from the starting vertex s via a path internally disjoint from P. Considering that failures can happen at any vertex P, this means we would need to store all vertices that are reachable from s on P via a path internally disjoint from P. This requirement renders the approach impractical.[2]

To address this issue, we need to adopt a different approach and develop new techniques specifically tailored for accommodating failures in the directed case.

It is worth mentioning that in both our reachability labeling scheme and the reachability oracle of [IKP21], the most challenging scenario arises already when all three vertices s, t, and f are situated on the same path separator P. In [IKP21], this situation was addressed by employing a complex data structure that extends dominator trees and previous data structures designed for handling strong-connectivity in general (non-planar) graphs under failures [GIP17]. However, these data structures do not seem to be distributable into a labeling scheme (for example, they rely on an orthogonal range data structure and on a binary search step which do not seem suitable for a labeling scheme). We tackle this case without relying on dominator trees or similar sophisticated techniques. This conceptually simpler solution is amenable to extension into an approximate distance labeling. We believe that it should be possible to extend our labeling scheme to multiple failures and to other graph families.

2.2 High level overview of our reachability labels

In this section we provide a high-level overview of the techniques and ideas used in order to obtain the reachability. Specifically, we show how to break down the reachability task to a series of 'Find the First' style sub-tasks. Then, the same conceptual partition into sub-tasks can be applied to approximate distances labeling, with some modification to each sub-task that takes path lengths into account. We apply a fully recursive decomposition of the graph G using shortest path separators [Tho04], which induces a hierarchical decomposition of the graph (with $O(\log n)$ levels), where every subgraph is partitioned in the next level into two subgraphs, separated by $O(1)$ shortest paths. For simplicity, we will assume here that each separator is composed of a single shortest path.

Consider first the task of reachability labeling *without* faults [Tho04]. In this case, there exists a separator P that separates s and t. Let $a = \mathsf{first}_\mathsf{G}(s, P)$ be the first vertex on P, reachable from s in G and let $b = \mathsf{last}_\mathsf{G}(t, P)$ be the last vertex on P that can reach t in G. It is straightforward that s can reach t in G if and only if a appears on P earlier than b (which we denote by $a \leq_P b$). See Figure 1. Thus, a labeling scheme for this simple problem is that every vertex v stores $\mathsf{first}_\mathsf{G}(v, P)$ and $\mathsf{last}_\mathsf{G}(v, P)$ for every separator P above it in the recursive decomposition.

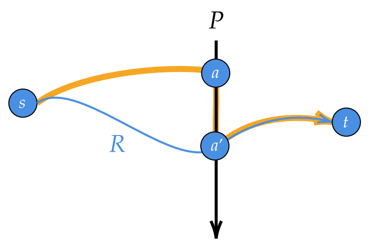


Figure 1: If R is a (blue) path from s to t that crosses P (at a'), then there exists an (orange) path from s to t that goes through $a = \mathsf{first}(s, P)$, then from a to a' along P and finally from a' to t along R).

[2]The same difficulty arises when trying to adapt the fault-tolerant labeling scheme for undirected planar graphs of [ACG12].

The 'Find the First' framework. To handle faults, we repeatedly employ the high-level approach of the non-faulty labels. If a path R in some graph H (in particular $H = G \setminus \{f\}$) from s to t visits a separator P, then there is a path from s to t that visits $a = \mathsf{first}_{\mathsf{H}}(s, P)$. Therefore, if we know that a path from s to t visits P, we can reduce s-to-t reachability to a-to-t reachability, and to finding a.

We consider two cases regarding the faulty vertex f: it is either on P or it is not on P.

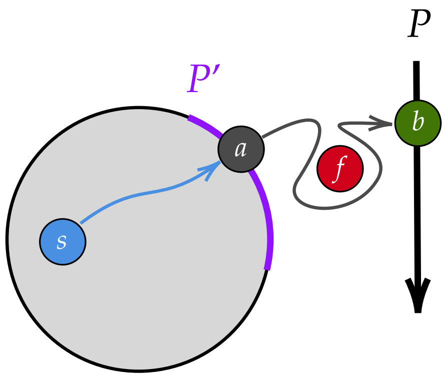


Figure 2: An lustration of the case $f \notin P$, a path from s to $b = \mathsf{first}_{\mathsf{G\setminus\{f\}}}(s, P)$. The blue subpath is from s to $a \in P'$, and the gray subpath is from a to $b \in P$.

If $f \notin P$ (see Figure 2), then the task is similar to the non-faulty case, except that now we are interested in $\mathsf{first}_{\mathsf{G\setminus\{f\}}}(s, P)$ instead of $\mathsf{first}_{\mathsf{G}}(s, P)$ which depends on f. We therefore need to introduce a labeling scheme that, given the labels of s and f can compute $b = \mathsf{first}_{\mathsf{G\setminus\{f\}}}(s, P)$.

The 'Find the First' logic allows us to break the problem of finding b itself into other subproblems of the form *find the first* reachable vertex on other separators; In addition to the separator P that separates s from t, we will use the separator P' that separates s from f. Let R be a path from s to b in $G \setminus \{f\}$. It is easy to handle the case where R does not cross P'. Otherwise, R crosses P'. Notice that we can assume without loss generality that R visits $a = \mathsf{first}_{\mathsf{G_{P'}}}(s, P')$, where $G_{P'}$ is the side of the separator P' that contains s and not f. It is clear that $\mathsf{first}_{\mathsf{G\setminus\{f\}}}(s, P) = \mathsf{first}_{\mathsf{G\setminus\{f\}}}(a, P)$. The label of s stores a, which allows us to reduce the problem to a specialized labeling scheme with the following settings: We are given two paths P' and P and two vertices $a \in P'$ and $f \notin P' \cup P$ and we wish to find $\mathsf{first}_{\mathsf{G\setminus\{f\}}}(a, P)$. We denote this labeling scheme by $\mathcal{L}^{\mathsf{P'} \xrightarrow{\mathsf{f}} \mathsf{P}}$. Designing $\mathcal{L}^{\mathsf{P'} \xrightarrow{\mathsf{f}} \mathsf{P}}$ turns out to be the main technical challenge in the case $f \notin P$, which we solve as follows.

For a vertex $v \in P'$ let u be the last vertex in P' such that $\mathsf{first}_{\mathsf{G}}(v, P) = \mathsf{first}_{\mathsf{G}}(u, P)$. Let C_u be a path in G from u to $\mathsf{first}_{\mathsf{G}}(u, P)$, we call such a path *canonical*. Moreover, we set $C_v = C_u$. Notice that when considering a path from $a \in P'$ to $\mathsf{first}_{\mathsf{G}}(a, P)$, we can always consider a path that goes from a to the beginning of C_a on P' and then continues on C_a. A helpful property of the paths C_v for $v \in P'$ is that they are disjoint, i.e. if $C_x \neq C_y$ then $C_x \cap C_y = \emptyset$. Therefore, each vertex f can store the (at most one) u such that $f \in C_u$ it lies on. If $f \notin C_a$, then $\mathsf{first}_{\mathsf{G\setminus\{f\}}}(a, P) = \mathsf{first}_{\mathsf{G}}(a, P)$. Otherwise $f \in C_a$, let R be a shortest path from a to $\mathsf{first}_{\mathsf{G\setminus\{f\}}}(a, P)$ in $G \setminus \{f\}$, let C_1 and C_2 be the prefix and suffix of C_a up to f. We distinguish between three cases: If $R \cap C_a = \emptyset$, the label of a stores $\mathsf{first}_{\mathsf{G\setminus C_a}}(a, P)$. If R intersects C_1, then $\mathsf{first}_{\mathsf{G\setminus\{f\}}}(a, P) = \mathsf{first}_{\mathsf{G\setminus\{f\}}}(u, P)$ (where $u \in P'$ is the first vertex of C_a), and f stores $\mathsf{first}_{\mathsf{G\setminus\{f\}}}(u, P)$. Otherwise, R intersects C_2 and $\mathsf{first}_{\mathsf{G\setminus\{f\}}}(a, P) = \mathsf{first}_{\mathsf{G}}(a, P)$. We recognize this case by storing in the label of a the last f on C_a such that $\mathsf{first}_{\mathsf{G\setminus\{f\}}}(a, P) = \mathsf{first}_{\mathsf{G}}(a, P)$.

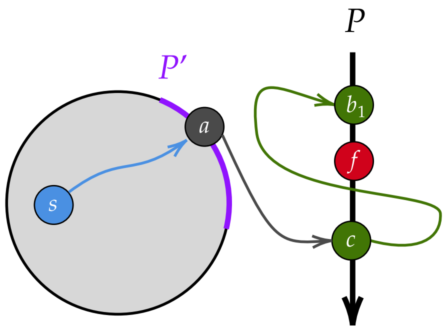


Figure 3: An lustration of the case $f \in P$.

If $f \in P$ (see Figure 3), it is no longer true that s can reach t in $G \setminus \{f\}$ if and only if $\mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s,P) \leq_P \mathsf{last}_{\mathsf{G}\setminus\{\mathsf{f}\}}(t,P)$. Instead, let P_1 and P_2 be the prefix and suffix of P before and after f (excluding f) respectively. Now it holds that s can reach t in $G \setminus \{f\}$ if either $\mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s,P_1) \leq_P \mathsf{last}_{\mathsf{G}\setminus\{\mathsf{f}\}}(t,P_1)$ or $\mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s,P_2) \leq_P \mathsf{last}_{\mathsf{G}\setminus\{\mathsf{f}\}}(t,P_2)$. Therefore, our goal is to retrieve $\mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s,P_1)$ and $\mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s,P_2)$ from the labels of s and f. We focus here on labels for finding $b_1 = \mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s,P_1)$, the labels for $\mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s,P_2)$ are similar.

Let R be a path in $G \setminus \{f\}$ from s to $\mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s,P_1)$. For the sake of simplicity we assume that R visits some vertex a' of P', and that the first vertex on P that is on R after a' is some vertex c' in P_2. We denote as G_P the side of the separator P that contains s. We break the task of computing $\mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s,P_1)$ into three sub-tasks.

1. Finding $a = \mathsf{first}_{\mathsf{G}_{\mathsf{P}'}}(s,P')$ (where again, P' separates s and f, and $G_{P'}$ is the side of P' that contains s and not f).
2. Finding $c = \mathsf{first}_{\mathsf{G}_\mathsf{P}\setminus\{\mathsf{f}\}}(a,P_2)$.[3]
3. Finding $\mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(c,P_1)$.

We show that $\mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s,P_1) = \mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(c,P_1)$, which immediately implies the usefulness of these three tasks. Let a' be the first vertex on R that is on P', and let c' be the first vertex on R after a' that is on P (specifically on P_2, due to our assumption). Notice that $R[s,a']$ is a path in $G_{P'}$ and $R[a',c']$ is a path in $G_P \setminus \{f\}$. Since a' is reachable from s in $G_{P'}$, the vertex a precedes a' on P'. We can therefore assume without loss of generality that R visits a, since s can reach a and a can reach a' via P', which implies that a can reach $\mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s,P_1)$. By the same reasoning, we get that $\mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(c,P_1) = \mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(a,P_1) = \mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s,P_1)$.

The most challenging out of the above three sub-problems is the third one. In order to tackle it, we further partition it into three sub-problems.

1. Given two vertices $b \in P_2$ and f, find $\mathsf{first}_{\mathsf{G}\setminus\mathsf{P}_1}(b,P_2)$
2. Given two vertices $b \in P_2$ and f, find the first vertex reachable on P_1 from b via a path that is internally disjoint from P_1.
3. Given two vertices $b \in P_1$ and f, find $\mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(b,P_1)$.

Again, it follows from 'Find the First' arguments that the above three problems, when put together, allow us to find $\mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(c,P_1)$ for a vertex $c \in P_2$.

[3]This is an inaccurate simplification for the sake of reducing clutter. In reality, the sub-problem is to find a first vertex using paths that are *internally disjoint* from P.

2.3 Extending the reachability labels to approximate-distance labels

We proceed to describe the labels for $(1+\varepsilon)$-distance approximation. Our $(1+\varepsilon)$-approximate distance labeling builds upon the same high-level approach of the reachability labeling. In order to adapt our reachability labeling for $(1+\varepsilon)$-distance approximation, we first need to modify some of the notations and ideas used in the reachability settings.

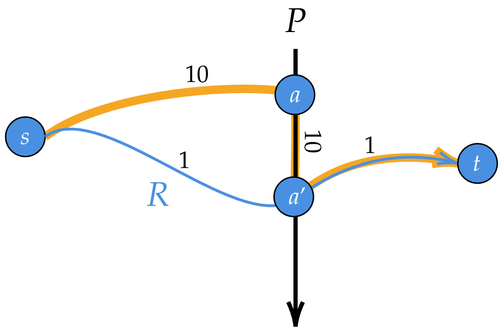


Figure 4: A naive attempt to use the reachability strategy for approximate distances may find a path which is much longer than the shortest path, hence fails to approximate the distance.

In order to motivate our modified definitions, let us consider a naive application of the 'Find the First' approach for approximate distances: i.e., given s, t, f, and a separator P that separates s from t, find $a = \mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s, P)$ and reduce the task of approximating $\mathsf{dist}_{G\setminus\{f\}}(s,t)$ to the task of approximating $\mathsf{dist}_{G\setminus\{f\}}(s,a)$ and $\mathsf{dist}_{G\setminus\{f\}}(a,t)$ (see Figure 4). This approach fails due to two reasons: First, while s can reach t via a, it may be the case that the subpath from s to a is very expensive, while reaching a later vertex on P is significantly cheaper. Second, even if reaching a is cheap, the subpath $P[a,a']$ might be significantly more expensive than the distance from s to t (a' is the actual vertex on P used by a shortest s-to-t path). Therefore, we have no guarantee on the distance from a to t. Compactly, the two problems can be put down as follows:

$$(1)\quad \mathsf{dist}_{G\setminus\{f\}}(s,a) \not\lesssim (1+\varepsilon)\mathsf{dist}_{G\setminus\{f\}}(s,a') \qquad (2)\quad \mathsf{dist}_{G\setminus\{f\}}(a,t) \not\lesssim (1+\varepsilon)\mathsf{dist}_{G\setminus\{f\}}(a',t)$$

We adjust our approach to treat these issues as follows. Using the framework of Thorup [Tho04] for (non-faulty) approximate distance labeling, we can assume that the separators of the recursive decomposition have the following structure: For some number r (think of r as an estimation of the distance from s to t in $G\setminus\{f\}$), the length of each separator in G is $O(r)$. In particular, we can partition each separator path into $O(\frac{1}{\varepsilon})$ subpaths, each of length $O(\varepsilon r)$. Let R be a shortest s to t path in $G\setminus\{f\}$. Now, when applying 'Find the First' strategy, we would like to find the first vertex on the specific subpath P that is visited, rather than on the entire separator. This resolves the second problem above. Since the cost of P is now bounded by $O(\varepsilon r)$, we have that $\mathsf{dist}(\mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s,P),t) \le \mathsf{dist}(a',t) + O(\varepsilon r)$. Since $\mathsf{dist}_{G\setminus\{f\}}(s,t) \approx r$, an additive factor of $O(\varepsilon r)$ to the answer is acceptable.

To handle the first problem, we refine the definition of $\mathsf{first}_{\mathsf{G}}(s,P)$ to take the length of the path into account. Specifically, we define $\mathsf{first}^{\alpha}_{\mathsf{G}}(s,P)$ to be the first vertex reachable on P from s *with a path of length at most* α. By guessing α such that $\mathsf{dist}_{G\setminus\{f\}}(s,a') \le \alpha \le \mathsf{dist}_{G\setminus\{f\}}(s,a') + O(\varepsilon r)$, we have that $\hat{a} = \mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s,P)$ is still earlier than a' on P, and $\mathsf{dist}_{G\setminus\{f\}}(s,\hat{a}) \le \mathsf{dist}_{G\setminus\{f\}}(s,a') + O(\varepsilon r)$.

It follows from the above discussion that $\mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s,P)$ with an appropriate value of α could be used to approximate the s to t distance similar to the way $\mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s,P)$ is used in reachability.

Notice that each application of the 'Find the First' approach in the approximate distance settings may introduce an additional additive factor of $O(\varepsilon r)$ to the final answer. When finding an approximate shortest path, we only apply this approach a constant number of times, so the additive $O(\varepsilon r)$ factors do not aggregate too much and sum up to $O(\varepsilon r)$ over all 'Find the First' applications.

Sadly, we do not know how to design a labeling scheme for finding $\mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s,P)$. We therefore introduce a further relaxation by defining the δ-$\mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s,P)$ property. We say that a vertex a^* is a δ-$\mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s,P)$ if $a^* \leq_P \mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s,P) \leq_P a'$ and $\mathsf{dist}(s,a^*) \leq \alpha+\delta$. Notice that for $\delta = O(\varepsilon r)$, the vertex a^* functions as a 'legitimate' middle point between s and t. That is, we have $\mathsf{dist}_{G\setminus\{f\}}(s,a^*) \leq \alpha + O(\varepsilon r) \leq \mathsf{dist}_{G\setminus\{f\}}(s,a') + O(\varepsilon r)$ and $\mathsf{dist}_{G\setminus\{f\}}(a^*,t) \leq \mathsf{len}(P[a^*,a']) + \mathsf{dist}_{G\setminus\{f\}}(a',t) \leq \mathsf{dist}_{G\setminus\{f\}}(a',t) + O(\varepsilon r)$. We can therefore define our sub-problems as the tasks of finding δ-$\mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s,P)$.

Adjusting reachability sub-routines to approximate distance sub-routines. The high level approach of 'Find the First' allows us to break the approximate-distance task into sub-problems in the same way as in reachability. However, adapting the labeling schemes for these problems from reachability to approximate distances poses significant technical difficulties, and requires new involved machinery. To demonstrate these difficulties, we focus on one of the sub-problems defined for the reachability labeling.

Consider the following sub-problem, which we call the *easy problem*. We are given a path P in G such that all the edges touching P emanate or enter P from or to the left, and we are interested in assigning labels to the vertices of P such that given the labels of two vertices $f <_P b$, one can find $\mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(b,P_1)$ such that P_1 is the prefix of P preceding f. Due to the context in which this sub-problem is used, we can also assume that P_1 has no outgoing edges to $G\setminus P_1$, and P_2 (the suffix of P following f) does not have incoming edges from $G\setminus P_2$. Essentially, this means that we are only interested in paths from P_2 to P_1 that are internally disjoint from P, apart from a prefix that is a subpath of P (see Figure 5).

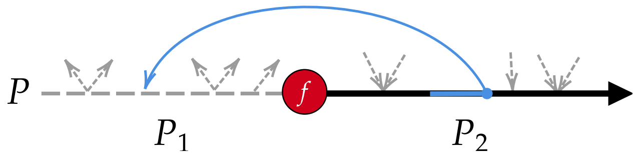


Figure 5: An illustration of G_1. We are interested in P_2 to P_1 paths (like the blue path) that may start with some edges of P_2 and then continue with a subpath which is internally disjoint from P. Formally, this is achieved by removing all in-going edges to P_2 and all out-going edges from P_1 (the removed edges are displayed in gray in the figure).

Indeed, it is easy to solve the easy problem; Given some faulty vertex f, we define for every $v \in P_2$ the vertex $v_f = \mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(v,P_1)$. Notice that v_f may be undefined: it is possible that v cannot reach any vertex on P_1. Let D_v be a path from v to v_f in $G\setminus\{f\}$. We show that for every two vertices x,y on P_2 that can reach P_1, it holds that $x_f = y_f$. This follows from the following two arguments: First, if $x <_P y$ we must also have $x_f \leq_P y_f$ since x can reach y. Then, if x_f is strictly before y_f, the paths D_x and D_y intersect (see Figure 12). In particular, an intersection means that y can reach $x_f <_P y_f$, a contradiction. We use this observation to define the label of f. Specifically, the label of f will store the last vertex ℓ on P_2 that can reach P_1, and ℓ_f. It follows from our observation that given the index of b on P_2, and the label of f, we can set $\mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(b,P_2)$ to be ℓ_f if $b \leq_P \ell$ or null otherwise.

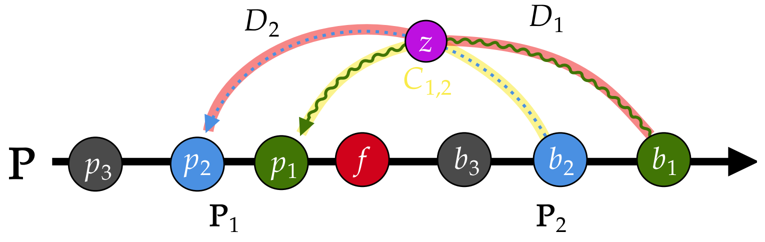
In the approximate-distance version of the easy problem, the settings are exactly the same but we are also given a number α and an approximation parameter ε. We are interested in assigning labels to the vertices of P such that given the labels of two vertices $f <_P b$, we can output a vertex that is an $\varepsilon\alpha\text{-}\mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(b, P_1)$. In this case, the length of P is zero.

Let us try to apply a similar logic to the approximate-distances variant. For a faulty vertex f, and for every $v \in P_2$ we now define $v_f = \mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(v, P_1)$. Because $\mathsf{len}(P) = 0$, it is still true that for every two vertices on P_2 such that $x <_P y$, we have $x_f \leq_P y_f$. However, it is no longer true that $x_f = y_f$. Now, if $x_f <_P y_f$, the intersection of D_x and D_y implies a path of length $\mathbf{2}\alpha$ from y to x_f. In particular, we have that the first vertex $p_f \in P_1$ that can be reached from some vertex $p \in P_2$ within budget α, is an $\alpha\text{-}\mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(v, P_1)$ for every $v \in P_2$ that can reach P_1 within budget α.

Therefore, for $\varepsilon = 1$ the approximation variant of the problem can be solved using an almost identical labeling scheme to the reachability variant. The label of vertex f will store the last vertex ℓ on P_2 that can reach a vertex on P_1 via a path of length at most α, and also stores p_f.

The 'Good Cross – Bad Cross' Framework. At first glance, it may seem as if the above logic completely fails when $\varepsilon < 1$. The length of the path implied by the intersection D_x and D_y may actually be 2α, and if so x_f may be not $\varepsilon\alpha\text{-}\mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(y, P_1)$. We overcome this issue by introducing the framework of *good cross and bad cross* which we use to solve multiple problems, in each of these problems this idea is used in a different way. This is one of the main technical contributions of the paper, and we strongly believe it would find future applications.

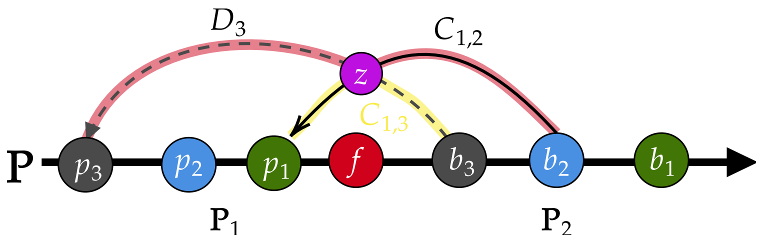
Let x and y be the first and last vertices on P_2 that can reach P_1 with budget α. Consider the 'lucky' situation in which y can reach x_f with budget $(1+\varepsilon)\alpha$. In this case, the construction we describe above would work. Namely, for every $v \in P[x, y]$, we have x_f is $\varepsilon\alpha\text{-}\mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(v, P_1)$. We call this kind of situation a *good cross*. A *bad cross* is the complementary case in which $\mathsf{dist}_{G\setminus\{f\}}(y, x_f) > (1+\varepsilon)\alpha$. In this case, in particular the path $D_{x,y}$ from y to x_f that uses a prefix of D_y and a suffix of D_x is too costly. We exploit the fact that the remainders of D_x and D_y that do not participate in $D_{x,y}$ form a cheap path in $G \setminus \{f\}$ from x to y_f.



Let us show how the concept is used to obtain labels of size $O(\frac{1}{\varepsilon})$. Let b_1 be the last vertex in P_2 that can reach P_1 with budget α, and let $p_1 = \mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(b_1, P_1)$. Now, let b_2 be the last vertex on $P_2(f, b_1]$ that has a 'bad cross' with b_1, i.e. the last vertex earlier than b_1 such that $\mathsf{dist}_{G\setminus\{f\}}(b_1, p_2) > (1+\varepsilon)\alpha$ for $p_2 = \mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(b_2, P_1)$. Denote the vertex following b_2 on P as b'_1 and $\mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(b'_1, P_1) = v_1$. By the definition of b_2, the vertices b_1 and b'_1 good cross each other, which means that $\mathsf{dist}_{G\setminus\{f\}}(b_1, v_1) \leq (1+\varepsilon)\alpha$. Due to $\mathsf{len}(P) = 0$ we also have that v_1 is an $\varepsilon\alpha\text{-}\mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(b, P_1)$ for every $b \in P_2(b_2, b_1]$.

We now discuss the implication of the bad cross between b_1 and b_2 . Consider shortest paths D_1 and D_2 from b_1 to p_1 and from b_2 to p_2, respectively. Due to planarity, the paths must intersect. Let $z \in D_1 \cap D_2$, and denote the path $D_{1,2} = D_1[b_1, z] \cdot D_2[z, p_2]$. Due to the bad cross, $\mathsf{len}(D_{1,2}) > (1+\varepsilon)\alpha$. Together, D_1 and D_2 cost at most 2α. Hence, the path $C_{1,2} = D_2[b_2, z] \cdot D_1[z, p_1]$ must have length of at most $2\alpha - (1+\varepsilon)\alpha = (1-\varepsilon)\alpha$.

As it is, $C_{1,2}$ does not seem like a 'useful' path: it allows b_2 to reach p_1 which can only be worse (i.e. later on P_1) than p_2. It is helpful to think of the length of $C_{1,2}$ as a 'budget' for bad crosses. We will describe an iterative procedure in which every time a bad cross occurs, we get a path that resembles $C_{1,2}$ whose length is smaller by $\varepsilon\alpha$.



Let us proceed by choosing b_3 to be the last vertex on $P(f, b_2]$ that has a bad cross with b_2. By the same reasoning, the vertex $v_2 = \mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(b'_2, P_1)$, with b'_2 being the successor of b_3 on P, is an $\varepsilon\alpha$-$\mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(b, P_1)$ for every $b \in P(b_3, b_2]$. Consequently, a shortest path D_3 from b_3 to $p_3 = \mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(b_3, P_1)$ must intersects $C_{1,2}$ at some vertex z. This yields the path $D_{2,3} = C_{1,2}[b_2, z] \cdot D_3[z, p_3]$ with $\mathsf{len}(D_{2,3}) > (1+\varepsilon)\alpha$ (due to the bad cross between b_2 and b_3).

Now, the sum of the lengths of D_3 and $C_{1,2}$ is bounded by $(2-\varepsilon)\alpha$. Therefore, we have that $C_{1,3} = D_3[b_3, z] \cdot C_{1,2}[z, p_1]$ has length at most $(2-\varepsilon)\alpha - (1+\varepsilon)\alpha = (1-2\varepsilon)\alpha$.

Clearly, this process must terminate after $O(\frac{1}{\varepsilon})$ steps. The sequences of b_i's and v_i's found in the process are stored in the label of f, which is sufficient to classify every b to the subpath of P such that $b \in P(b_i, b_{i-1}]$ and report v_i as an answer for b. Thus, a labeling scheme with labels of size $O(\frac{1}{\varepsilon})$ is achieved for the approximate version of the easy problem.

The labeling scheme we described above is the simplest application of the 'good cross-bad cross' technique. As it turns out, this idea proves useful in generalizing many of the reachability subroutines to the approximate distance settings. Intuitively, the reachability variants of each subroutine depend on arguments of the form: *"There is a path from b_1 to p_1 and a path from b_2 to p_2. These paths intersect, so there is a path from b_1 to p_2"*. One would like to have the following similar argument for approximate distances: *"There is a path of length α from b_1 to p_1 and a path of length α from b_2 to p_2. These paths intersect, so there is a path of length $(1+\varepsilon)\alpha$ from b_1 to p_2"*. Unfortunately, the latter statement is not true in general. The 'good cross-bad cross' technique exploits the fact that if the latter statement is not true (which we can consider as a 'bad event'), then *some* path is cheap. The cheap path on its own is not necessarily useful, e.g. the path from b_2 to p_1 is not an approximate shortest path. However, the length of the cheap path can be used as a decreasing potential that cannot become negative, and therefore bounds the number of 'bad' events.

The solution for the easy problem can be described as a selection procedure for 'useful' vertices that terminates quickly due to 'good cross-bad cross' arguments. The challenge in applying the 'good cross- bad cross' approach to other sub-problems is in defining the selection mechanism. Specifically, the paths in the easy problem have a convenient structure in the sense that every two paths must intersect. This is not the case in other sub-problems. For example, in other sub-problems, the cheap paths created as a result of a bad cross in each step of the algorithm seem entirely unrelated to the sub-problem at hand. One needs to creatively find an intricate way this relation can be made.

3 Preliminaries

The decomposition tree. In [Tho04, Lemma 2.2], Thorup proved that we can assume the graph G has an undirected spanning tree T (i.e., T is an unrooted spanning tree in the undirected graph obtained from G by ignoring the directions of edge) such that each path in T is the concatenation of $O(1)$ directed paths in G.

This way, we can describe the process of decomposing G into pieces in the undirected version of G. After describing the decomposition, we will replace each undirected path of T defined in the process by its $O(1)$ corresponding directed paths in G. We therefore proceed to describe the decomposition treating G as an undirected graph with a rooted spanning tree T.

A balanced fundamental cycle separator [KM, Lemma 5.3.2] (cf. [LT79] and [Tho04, Lemma 2.3]) is a simple cycle C in G whose vertices are those of a single path of the (unrooted) spanning tree T, such that the removal of the vertices of C and their incident edges separates G into two roughly equal sized subgraphs. The balance of the separator can be defined with respect to a weight function on the vertices of G rather than just the number of vertices.

The recursive decomposition tree $\mathcal{T}$ of G is defined as follows. Each node of $\mathcal{T}$ corresponds to a subgraph of G (called a *piece*). The root piece of $\mathcal{T}$ is the entire graph G. The boundary ∂G of G is defined to be the empty set. We define the children of a piece H in $\mathcal{T}$ recursively. Let ∂H be the boundary of H, and let C be a fundamental cycle separator which balances the number of non-boundary vertices of H. Let Q be the set of maximal subpaths of C that are internally disjoint from ∂H. To reduce clutter we sometimes refer to a vertex $v \in Q$, by which we mean that v belongs to some path in Q. The vertices of H that are enclosed by C (including the vertices of C) belong to one child H_1 of H. The vertices of subpaths in Q and the vertices of H not enclosed by C belong to the other child H_2. Note that the vertices of Q are the only vertices of H that belong to both children. The endpoints of Q that belong to ∂H are called the *apices* of H. The importance of apices arises from the fact that apices are the only vertices that belong to more than two pieces at the same depth of $\mathcal{T}$.[4] We call the paths in Q without the apices of H the *separator* of H. We do not include the apices in the separator paths to guarantee that the separator is vertex disjoint from ∂H. The boundary ∂H_i of a child H_i of H consists of the separator of H and of the subpaths of ∂H induced by the vertices of H_i.

Since H_2 might not contain all the vertices of C, the subgraph induced on the spanning tree T by H_2 may become disconnected. To overcome this slight technical issue we embed in H_2, if necessary, an artificial root connected by artificial edges to the rootmost apex in each of the resulting components of the tree. The embedding remains planar since all these apices were on the fundamental cycle separator of the parent piece H. We treat the artificial root and edges as part of the spanning tree of H_2. To guarantee that the addition of the artificial root and edges does not affect the reachability of non-artificial vertices of H_2, we direct the artificial edges into the artificial root.

The leaves of $\mathcal{T}$ (called *atomic* pieces) correspond to pieces with $O(1)$ non-boundary vertices. The depth of $\mathcal{T}$ is $O(\log n)$. For convenience, we consider all $O(1)$ vertices of an atomic (leaf) piece that are not already boundary vertices as the separator of the piece. It follows that the boundary ∂H of any piece H consists of $\tilde{O}(1)$ vertex disjoint paths (the subpaths induced by the vertices of H on the separators of the ancestor pieces of H), and each of the paths in ∂H lies on a single face of H. Also, since in each node of the decomposition tree only $\tilde{O}(1)$ apices are created, there are $\tilde{O}(1)$ apices along any root-to-leaf path in $\mathcal{T}$.

Having defined the decomposition tree $\mathcal{T}$ we can go back to treating G as a directed graph. As

[4]The term apex was previously used in exactly the same context in [ACG12].

we explained above, each path we had discussed in the undirected version of G is the union of $O(1)$ directed paths in G. From now on when we refer to the separator paths of a piece H (resp., paths of ∂H), we mean the set of directed paths comprising the undirected separator paths of H (resp., the set of directed paths comprising the paths of ∂H).

To be able to control the size of the labels in our construction we need to be aware of the number of pieces of $\mathcal{T}$ to which a vertex belongs. The only vertices of a piece H that belong to both its children are the vertices of the separator path of H and the $\tilde{O}(1)$ apices of H. The fact that separator paths are disjoint from the boundary imply that every vertex belongs to the separator of at most one piece in $\mathcal{T}$. Hence, if a vertex is not an apex, it appears in $O(\log n)$ pieces of $\mathcal{T}$. Apices, on the other hand require special attention because they may belong to many (e.g., polynomially many) pieces of $\mathcal{T}$; High degree vertices may be apices in many pieces, and we will need a special mechanism for dealing with such vertices. Dealing with apices (like dealing with holes in other works on planar graphs) introduces technical complications that are not pertinent to understanding the main ideas of our work.[5]

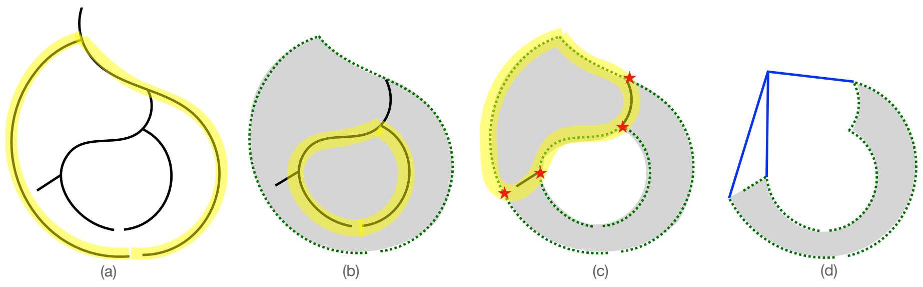


Figure 6: Illustration of the recursive decomposition process. (a) a portion of the spanning tree T of G is shown in black. A balanced fundamental cycle separator of G is highlighted in yellow. (b) The piece H containing the vertices of G enclosed by the fundamental cycle of G is indicated in shaded grey. The boundary ∂H (dashed) consists of the separator of G. Portions of the spanning tree T induced by the vertices of H are shown, along with a fundamental cycle separator of H, highlighted in yellow. (c) The piece H_2 containing the vertices of H not strictly enclosed by the fundamental cycle separator of H is indicated in shaded grey. The boundary ∂H_2 (dashed) consists of ∂H and of the separator of H. Portions of the spanning tree T induced by the vertices of H_2 are shown, along with a fundamental cycle separator of H_2, highlighted in yellow. There are two maximal subpaths Q of the fundamental cycle separator of H_2 that do not belong to ∂H_2 (the two parts of the highlighted yellow cycle that are not dashed). The endpoints of Q that belong to ∂H_2 are the apices of H_2 (the four red stars). (d) The piece H_{22} containing the vertices of H_2 not strictly enclosed by the fundamental cycle of H_2 is indicated in shaded grey. The boundary ∂H_{22} (dashed) consists of the paths induced by H_{22} on ∂H_2, and of the separator of H_2. Since the subgraph induced on T by the vertices of H_{22} is disconnected, an artificial root and edges (in blue) are added to H_{22}.

We associate with every vertex $v \in G$ the (at most 2) rootmost pieces H in $\mathcal{T}$ in which v is an apex (or the atomic pieces containing v if v is never an apex). We denote these pieces by H_v. Note that every piece that contains a vertex v is either an ancestor of a piece in H_v or a descendant of a piece in H_v. For a vertex $v \in G$ we define the *ancestor pieces* of v to be the set of (weak)

[5]A reader who is not interested in those details can safely skip the parts dealing with apices and just act under the assumption that each vertex appears in 2 atomic pieces (leaves) of $\mathcal{T}$, and that the following definition of ancestor pieces of a vertex v just degenerates to the set of $O(\log n)$ ancestors of the 2 atomic pieces containing v.

ancestors in $\mathcal{T}$ of the pieces H_v. By definition of H_v, every vertex, apex or not, has $O(\log n)$ ancestors pieces. We similarly define the ancestor separators/paths/apices of a vertex $v \in G$ as the separators/separator-paths/apices of any ancestor piece of v. See Figure 7.

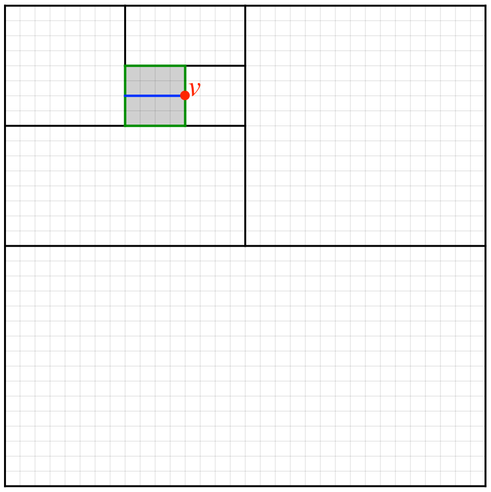


Figure 7: Pieces in a recursive decomposition. The separators in this example alternate between horizontal and vertical lines. All rectangular pieces that contain the gray piece H are its ancestors. The boundary ∂H of H is shown in green. The maximal subpath of the cycle separator of H that is internally disjoint from ∂H is shown in blue. The vertex v is an apex of H because it is an endpoint of this maximal subpath. The separator of H is the blue path (without its endpoints).

We say that the separator Q of a piece H *separates* two vertices u and v (in H) if any u-to-v path in H must touch the separator Q of H or an apex of H. I.e., if at least one of the following holds: (1) $u \in Q$ or u is an apex of H, or (2) $v \in Q$ or v is an apex of H, or (3) each of u and v is in one distinct child of H. Note that if Q separates u and v in H then every u-to-v path in G either touches Q or touches the boundary of H.

For a subgraph H, a path P and a vertex v, let $\mathsf{first}_{\mathsf{H}}(v, P)$ denote the first vertex of P that is reachable from v in H, and let $\mathsf{last}_{\mathsf{H}}(v, P)$ denote the last vertex of P that can reach v in H. If vertex u appears before vertex v on a path P then we denote this by $u <_P v$ (or simply $u < v$ if P is clear from the context). Throughout the paper, we gradually describe the information stored in the labels along with the explanations of why this particular information is stored (and why it is polylogarithmic). To assist the reader, we highlight in gray the parts that describe the information stored. For starters, we let every vertex $v \in G$ store in its label, for every ancestor path P of v, the identity of P and, if $v \in P$, the location of v in P (so that given two vertices u, v of P, we can tell if $u < v$). We denote $P[u, v]$ the subpath of P between vertices u and v. To avoid unnecessary repetitions in the text, we assume that this information is included in every labeling scheme described in this paper, and do not include it explicitly in their descriptions.

Thorup's non-faulty labeling. Using the above definitions and notations, it is now very simple to describe Thorup's non-faulty reachability labeling [Tho04]. Consider any vertex v. Let H be the rootmost piece in $\mathcal{T}$ in which v belongs to the separator. The crucial observation is that v is separated from every other vertex in G either by the separator of H or by the separator of some ancestor piece of H. Hence, every vertex $v \in G$ stores in its label $\mathsf{first}_{\mathsf{G}}(v, P)$ and $\mathsf{last}_{\mathsf{G}}(v, P)$ for every path P of the separator of every ancestor of the rootmost piece in which v belongs to the separator.

Then, given a query pair u, v, there exists a u-to-v path in G if and only if $\mathsf{first}_{\mathsf{G}}(u, P) < \mathsf{last}_{\mathsf{G}}(v, P)$ for one of the $O(1)$ paths P of the separator of an ancestor piece of the rootmost piece whose separator separates u and v. Both u and v store the relevant information for these paths in their labels. We note that in Thorup's scheme we do not need to worry about apices since each vertex v only stores information in pieces above the first time v appears on a separator.

4 Reachability Labeling

In this section, we describe our reachability labeling scheme. I.e., what to store in the labels so that given the labels of any three vertices s, f, t we can infer whether t is reachable from s in $G \setminus \{f\}$. We call the s-to-t path R in $G \setminus \{f\}$ the *replacement path.*

Theorem 4.1. *There exists a labeling scheme for reachability for G that, given vertices s, t, f returns whether t is reachable from s in $G \setminus \{f\}$. The size of each label is $\tilde{O}(1)$.*

4.1 Reduction into simpler problems

Instead of describing our labeling scheme that proves Theorem 4.1 directly, we describe labeling for several specialized tasks, and show how to combine these specialized labeling schemes into the desired labeling scheme for reachability. While this division may seem somewhat excessive for the description of the reachability labels, it serves to clearly present and explain the high level structure of the scheme, in preparation for the more complicated and elaborate scheme for approximate distances in Section 5. The first reduction is to two labeling schemes from a vertex to a path in the presence of a failed vertex, one for the case that the failing vertex is not on the path, and the other for the case that it is.

Lemma 4.2. *There exists a labeling scheme $\mathcal{L}^{\mathtt{s}\xrightarrow{\mathtt{f}}\mathtt{P}} = \mathcal{L}_{G,P}^{\mathtt{s}\xrightarrow{\mathtt{f}}\mathtt{P}}$ where G is a planar graph equipped with a decomposition tree $\mathcal{T}$, and P is a path in $\mathcal{T}$. Let s and $f \notin P$ be two vertices of G that are not an ancestor apex of one another, and such that P is an ancestor of both s and f. Given the labels of s and f, one can compute the index on P of the vertex b that is $\mathsf{first}_{\mathsf{G}\setminus\{\mathtt{f}\}}(s, P)$. In this labeling scheme, the only vertices that store a label are those that have P as an ancestor. The size of each label stored by such a vertex is $\tilde{O}(1)$.*

Lemma 4.3. *There exists a labeling scheme $\mathcal{L}^{\mathtt{s}\rightarrow\mathtt{P}\setminus\{\mathtt{f}\}} = \mathcal{L}_{H,P}^{\mathtt{s}\rightarrow\mathtt{P}\setminus\{\mathtt{f}\}}$ where H is a planar graph equipped with a decomposition tree $\mathcal{T}$, and P is a path in $\mathcal{T}$ such that both endpoints of P lie on the same face of H. Let s and $f \in P$ be two vertices of H that are not an ancestor apex of one another, and such that P is an ancestor of both s and f. Given the labels of s and f, one can compute the indices on P of the vertices $b_1 = \mathsf{first}_{\mathsf{H}\setminus\mathsf{f}}(s, P_1)$ and $b_2 = \mathsf{first}_{\mathsf{H}\setminus\mathsf{f}}(s, P_2)$, where P_1 (resp. P_2) is the prefix (resp. suffix) of P that precedes (resp. follows) f, excluding f. In this labeling scheme, the only vertices that store a label are those that have P as an ancestor. The size of each label stored by such a vertex is $\tilde{O}(1)$.*

Proof of Theorem 4.1. Let G_{rev} be the graph obtained from G by reversing all edges (the reverse graph G_{rev} has exactly the same decomposition tree as G, but the direction of each path is reversed).

The label of a vertex v consists of the following:

1. For each piece H in the recursive decomposition $\mathcal{T}$ of G such that $v \in H \setminus \partial H$, v stores its label in the standard (non-faulty) labeling of Thorup for $H \setminus \partial H$.

2. For every ancestor piece H in G, v stores $\mathsf{last}_{\mathsf{H}}(v,P)$ for each of the $O(\log n)$ paths P of ∂H.

3. Using of Lemma 4.2, for every ancestor path P of v in the recursive decomposition $\mathcal{T}$, v stores $\mathcal{L}^{\mathtt{s}\xrightarrow{\mathtt{f}}\mathtt{P}}_{G,P}(v)$.

4. Using Lemma 4.3, for every ancestor piece H of v in $\mathcal{T}$, for every path P of the cycle separator C of H, v stores $\mathcal{L}^{\mathtt{s}\longrightarrow\mathtt{P}\setminus\{\mathtt{f}\}}_{H^{\times P},P}(v)$, where $H^{\times P}$ is the graph obtained from $H \setminus \partial H$ by making an incision along all the edges of C[6] other than those of P. Note that, because of the incision, the endpoints of P lie on a single face of $H^{\times P}$, so Lemma 4.3 indeed applies.

5. For every ancestor apex a of v, for every path P that is an ancestor of v, v stores in its label $\mathsf{first}_{\mathsf{G}\setminus\mathsf{a}}(v,P)$, and $\mathsf{first}_{\mathsf{G}\setminus\mathsf{v}}(a,P)$. If a lies on P then let P_2 be the suffix of P after a (not including f). v also stores $\mathsf{first}_{\mathsf{G}\setminus\mathsf{a}}(v,P_2)$. Similarly, if v lies on P then let P_2 be the suffix of P after v (not including v). v also stores $\mathsf{first}_{\mathsf{G}\setminus\mathsf{v}}(a,P_2)$.

6. v also stores all the above items computed in the graph G_{rev} instead of the graph G.

Size. Since each vertex has only $\tilde{O}(1)$ ancestor pieces, paths and apices, and by Lemmas 4.2 and 4.3, all items above sum up to a label of size $\tilde{O}(1)$.

Decoding and Correctness. Let $\hat{H}$ be the rootmost piece in $\mathcal{T}$ whose separator Q separates t and f. Let H be a child piece of $\hat{H}$ that contains t (if both children of $\hat{H}$ contain t then, if one of the children does not contain f we choose H to be that child). Note that by choice of H, $f \notin H \setminus \partial H$.

We assume without loss of generality that $s \in \hat{H}$. We handle the other case analogously to the description below, by swapping the roles of s and t and working in G_{rev} instead of in G.

Observe that by definition of H and of separation, $f \in \partial H$ iff $f \in Q$. Consider first the case when a replacement path R does not touch ∂H. i.e., s,t and R are all contained in $H \setminus \partial H$, and f is not contained in $H \setminus \partial H$. In this case querying Thorup's non-faulty labels for $H \setminus \partial H$ (stored in item (1)) will correctly identify the existence of a replacement path.

To treat the case where the replacement path R touches ∂H, we separately handle the cases where $f \notin Q$ and $f \in Q$.

When $f \notin Q$ (and so, $f \notin \partial H$). In this case, R must have a suffix contained in H, and this suffix is unaffected by the fault f. More precisely, R exists iff $\mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s,P) < \mathsf{last}_{\mathsf{H}}(t,P)$ for one of the paths P forming ∂H. The vertex $\mathsf{last}_{\mathsf{H}}(t,P)$ can be retrieved from item (2) of the label of t. Notice that by the rootmost choice of $\hat{H}$, H is an ancestor piece of t, so t indeed stores $\mathsf{last}_{\mathsf{H}}(t,P)$. It thus remains only to describe how to find $\mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s,P)$ from the labels of s and f. If either s or f store $\mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s,P)$ in item (5), we are done. Otherwise neither s nor f is an ancestor apex of the other, and since both s and f are in $\hat{H}$, P is indeed an ancestor of both s and f, so, by Lemma 4.2, $\mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s,P)$ can be obtained from $\mathcal{L}^{\mathtt{s}\xrightarrow{\mathtt{f}}\mathtt{P}}_{G,P}(s)$ (stored in item (3) of the label of s) and $\mathcal{L}^{\mathtt{s}\xrightarrow{\mathtt{f}}\mathtt{P}}_{G,P}(f)$ (stored in item (3) of the label of f).

[6]We refer here to the cycle separator C of H and not to the separator Q of H because the separator Q only includes the subpaths of C that do not belong to ∂H. Here we want to make the incision along the entire cycle separator.

When $f \in Q$. Let P be the path of Q that contains f. Consider first the case where the replacement path R touches some path $\hat{P} \neq P$ of Q or of the boundary of some ancestor of H. Since boundary paths are vertex disjoint, $f \in P$ implies $f \notin \hat{P}$. Hence, we can obtain $\mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s, \hat{P})$ in a similar manner to the case $f \notin Q$ above, with $\hat{P}$ taking the role of P. In an analogous manner, we can obtain $\mathsf{last}_{\mathsf{G}\setminus\{\mathsf{f}\}}(t, \hat{P})$ as we just found $\mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s, \hat{P})$, but with $\hat{P}$ taking the role of P, t taking the role of s, and G_{rev} taking the role of G (we cannot use part (2) of the label of t in this case because now f does belong to ∂H). Then, s can reach t in $G \setminus \{f\}$ iff $\mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s, \hat{P}) <_{\hat{P}} \mathsf{last}_{\mathsf{G}\setminus\{\mathsf{f}\}}(t, \hat{P})$. See Figure 9 (right).

Now consider the case where other than P, R does not touch any path of Q or any path of the boundary of an ancestor of H. In this case, we might as well use labels in $\hat{H}^{\times P}$ instead of in G because R does touch $\partial\hat{H}$, and only crosses Q at P. Let P_1 and P_2 be the prefix and suffix obtained from P by deleting f. If either s or f is an ancestor apex of one another then $\mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s, P_2)$ is stored in item (5) of either s or t, and, if $\mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s, P_1)$ exists, then it is equal to $\mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s, P)$, which is also stored in item (5) of either s or t. (If $\mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s, P)$ is not a vertex of P_1 then $\mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s, P_1)$ does not exist. If neither s not t is an ancestor apex of the other, then P is an ancestor of both s and f, and $\mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s, P_1)$ and $\mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s, P_2)$ can be obtained, by Lemma 4.3, from $\mathcal{L}^{\mathsf{s}\to\mathsf{P}\setminus\{\mathsf{f}\}}_{\hat{H}^{\times P},P}(s)$ (stored in item (4) of the label of s) and $\mathcal{L}^{\mathsf{s}\to\mathsf{P}\setminus\{\mathsf{f}\}}_{\hat{H}^{\times P},P}(f)$ (stored in item (4) of the label of f). Then, s can reach t in $G \setminus \{f\}$ iff either $\mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s, P_1) <_{P_1} \mathsf{last}_{\mathsf{G}\setminus\{\mathsf{f}\}}(t, P_1)$ or $\mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s, P_2) <_{P_2} \mathsf{last}_{\mathsf{G}\setminus\{\mathsf{f}\}}(t, P_2)$. □

4.2 The $\mathcal{L}^{\mathsf{s}\xrightarrow{\mathsf{f}}\mathsf{P}}$ labeling (Lemma 4.2)

To show the labeling schemes $\mathcal{L}^{\mathsf{s}\xrightarrow{\mathsf{f}}\mathsf{P}}$ (i.e., prove Lemma 4.2) we will compose the following specialized labeling schemes, which we prove in the sequel.

Lemma 4.4. *There exists a labeling scheme $\mathcal{L}^{\mathsf{P'}\xrightarrow{\mathsf{f}}\mathsf{P}} = \mathcal{L}^{\mathsf{P'}\xrightarrow{\mathsf{f}}\mathsf{P}}_{H,P,P'}$ where H is a planar graph, and P and P' are two paths. Given the labels of a vertex a on P' and a vertex f not in $P' \cup P$, one can retrieve the index on P of the vertex $b = \mathsf{first}_{\mathsf{H}\setminus\mathsf{f}}(a, P)$. The size of each label is $\tilde{O}(1)$.*

Lemma 4.5. *There exists a (trivial) labeling scheme $\mathcal{L}^{\mathsf{s}\to\mathsf{P'}} = \mathcal{L}^{\mathsf{s}\to\mathsf{P'}}_{H,P'}$ where H is a planar graph with a path P'. Given the label of a vertex s, one can retrieve the vertex $\mathsf{first}_{\mathsf{H}}(s, P')$. The size of each label is $\tilde{O}(1)$.*

We note that the proof of Lemma 4.5 is trivial, as each vertex s of H only needs to store the identity of $\mathsf{first}_{\mathsf{H}}(s, P')$. The proof of Lemma 4.4 appears after the following proof of Lemma 4.2.

Proof of Lemma 4.2. The labeling scheme only labels vertices whose ancestor is P. The label of such a vertex v consists of the following:

1. Using Lemma 4.4, for every ancestor path P' of v that has P as an ancestor, the label $\mathcal{L}^{\mathsf{P'}\xrightarrow{\mathsf{f}}\mathsf{P}}_{G,P',P}(v)$.

2. (Composition of labels) For a path P', let G^1 be the graph G labeled with the labels $\mathcal{L}^{\mathsf{P'}\xrightarrow{\mathsf{f}}\mathsf{P}}_{G,P',P}$ of Lemma 4.4. Using Lemma 4.5, for every ancestor path P' of v such that P is an ancestor of P', v stores the label $\mathcal{L}^{\mathsf{s}\to\mathsf{P'}}_{G^1,P'}(v)$ (i.e., the label $\mathcal{L}^{\mathsf{s}\to\mathsf{P'}}_{G^1,P'}(v)$ contains not only the identity of the desired vertex $a = \mathsf{first}_{\mathsf{H}}(v, P')$, but also the label $\mathcal{L}^{\mathsf{P'}\xrightarrow{\mathsf{f}}\mathsf{P}}_{G,P',P}(a)$ that a is labeled with in G^1).

Size. Since each vertex has $\tilde{O}(1)$ ancestor paths and by Lemmas 4.3 and 4.4, the size of the label of each vertex is $\tilde{O}(1)$.

Decoding and Correctness. Assume, per the statement of the lemma that P is an ancestor of both s and f and that s and f are not an ancestor apex of one another. Consider the set of leafmost pieces in $\mathcal{T}$ that contain both s and f. Let H'' be such a piece. It must be that H'' is an ancestor piece of both s and f or else one of s and f is an ancestor apex of the other. Hence, there are only $O(1)$ leafmost pieces that contain both s and f. To avoid unnecessary clutter we shall assume there is a unique piece H''. In reality we would have to apply the same argument for all $O(1)$ such pieces. Since H'' is an ancestor piece of both s and f, we can find the piece H'' by traversing the list of ancestors of s (stored in s) and of f (stored in f) until finding the lowest common ancestor. Let H' be the child piece of H'' that contains only s (if H'' is an atomic piece then define $H' = H''$). Since P is an ancestor of both s and f, P is a boundary path of a (possibly weak) ancestor $\hat{H}$ of H''.

Consider a path in $G \setminus \{f\}$ from s to $\mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s, P)$. Such a path begins with a prefix that is contained in H' from s to $a = \mathsf{first}_{\mathsf{H}'}(s, P')$ where P' is some path of $\partial H'$. Since $f \notin H'$, the candidate a can be retrieved from the label $\mathcal{L}^{\mathsf{s}\to\mathsf{P}'}$ of s, stored in item (3), along with the label of a in $\mathcal{L}_{G,P',P}^{\mathsf{P}'\xrightarrow{\mathsf{f}}\mathsf{P}}$. From this label, and from the label $\mathcal{L}_{G,P',P}^{\mathsf{P}'\xrightarrow{\mathsf{f}}\mathsf{P}}$ of f stored in item (2) we can obtain $\mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(a, P)$ (which is equal to $\mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s, P)$). Note that the conditions of Lemma 4.4 apply since $f \notin H'$ (and hence $f \notin P'$), and $f \notin P$. □

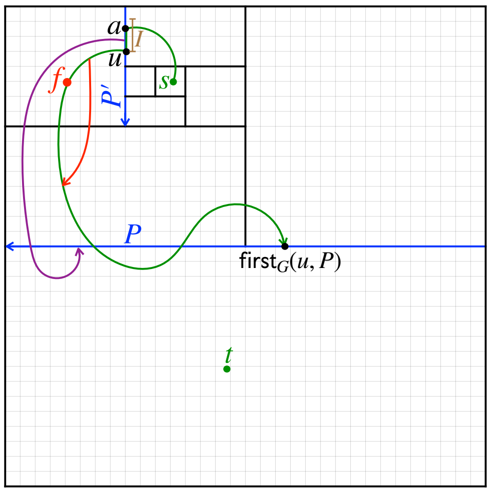


Figure 8: When $f \notin Q$: The (green) s-to-t path in G first touches the (blue) separator path $P' \in \partial H'$ at vertex a (that is found using Lemma 4.5), then continues along P' to u, then goes through f to $\mathsf{first}_{\mathsf{G}}(u, P)$ (that is found using Lemma 4.4) on the (blue) separator path $P \in \partial H$. The (brown) interval I contains all vertices p that can reach $\mathsf{first}_{\mathsf{G}}(p, P)$ through u. When f fails, the replacement s-to-t path in $G \setminus \{f\}$ either takes a (red) detour around f (and then f stores this information) or is (purple) completely disjoint from the original (green) u-to-$\mathsf{first}_{\mathsf{G}}(u, P)$ part (and then s stores this information).

Proof of Lemma 4.4. Consider the Pair of paths P' and P in the statement of the lemma. We shall define a set $C = C_{P',P}$ of canonical paths from P' to P in the graph H. Notice that for two vertices $p \leq_{P'} q$ on P', $\mathsf{first}_{\mathsf{G}}(p, P) \leq_P \mathsf{first}_{\mathsf{G}}(q, P)$ (this is because any vertex of P' earlier than q can reach

$\mathsf{first}_{\mathsf{G}}(q, P)$ through q). It follows that P' can be partitioned into maximal contiguous intervals of vertices that have the same $\mathsf{first}_{\mathsf{G}}(q, P)$. For every such interval I, we can think of all $p \in I$ as first going along P' until the last vertex u of I, and then reaching $\mathsf{first}_{\mathsf{G}}(u, P)$ in G using the same path S_I. We call S_I the canonical path of interval I, and let C be the set of all canonical paths for all the intervals in the partition of P'. For a vertex $a \in I$, we call S_I the canonical interval (of P', P) used by a.

Observe that the paths in C are mutually disjoint, since otherwise the corresponding intervals would be the same interval. It follows that f can be on at most one of those canonical paths. The canonical paths and intervals are useful because their definition only depends on P' and P, but not on the identity of s and f.

The label of a vertex v consists of the following:

1. $\mathsf{first}_{\mathsf{H}}(v, P)$.
2. if v lies on P', v stores:
   (a) its index on P'.
   (b) $\mathsf{first}_{\mathsf{H} \setminus \mathsf{S_I}}(v, P)$, where S_I is the canonical path used by v.
3. if v lies on some canonical path S_I of $C_{P',P}$, we let v store: v also stores:
   (a) the indices in P' of the endpoints of I.
   (b) the index of the last vertex of the prefix of I that can reach $\mathsf{first}_{\mathsf{H}}(f, P)$ in $H \setminus v$.
   (c) $\mathsf{first}_{\mathsf{H} \setminus \mathsf{v}}(u, P)$, where u is the last vertex of I.

Size. Clearly, each vertex stores $\tilde{O}(1)$ bits.

Decoding and Correctness. Given the labels of a and f, we can check if the index of a on P' (stored in item (2a) is in the interval I stored in item (3a) of the label of f. If a is not in the interval I, then the path in H from a to $\mathsf{first}_{\mathsf{H}}(a, P)$ does not go through f and therefore $\mathsf{first}_{\mathsf{H} \setminus \mathsf{f}}(a, P) = \mathsf{first}_{\mathsf{H}}(a, P)$, which is available in item (1) of the label of a. Otherwise, a is in the interval I stored by f, and the path in H from a to $\mathsf{first}_{\mathsf{H}}(a, P)$ does go through f. In particular $\mathsf{first}_{\mathsf{H}}(a, P) = \mathsf{first}_{\mathsf{H}}(f, P)$. Let S_1 be the prefix of S_I ending just before f, and let S_2 be the suffix of S_I starting immediately after f. Then, there are two options regarding the path in $H \setminus f$ from a to $\mathsf{first}_{\mathsf{H} \setminus \mathsf{f}}(a, P)$:

1. The path intersects S_2. In this case, the path can continue along S_2 until reaching $\mathsf{first}_{\mathsf{H}}(f, P)$, so $\mathsf{first}_{\mathsf{H} \setminus \mathsf{f}}(a, P) = \mathsf{first}_{\mathsf{H}}(f, P)$ and we have it stored in item (1) of the label of f. It only remains to check if this is indeed the case. Consider all vertices $p \in I$ that can reach $\mathsf{first}_{\mathsf{H}}(f, P)$ in $H \setminus f$. They must constitute a (possibly empty) prefix of I (since any such vertex $p \in I$ can reach any later vertex of I by going along I). Item (3b) of the label of f stores the index of the last vertex of the prefix of I that can reach $\mathsf{first}_{\mathsf{H}}(f, P)$ in $H \setminus f$. Therefore, we only need to check if a is earlier on P' than this vertex.

2. The path is disjoint from S_2. To handle this case, we will identify two valid candidates for $\mathsf{first}_{\mathsf{H} \setminus \mathsf{f}}(a, P)$ and take the earlier in P of these two candidates:

(a) The path is disjoint from S. To handle this case, we use item (2b) of the label of a, which stores the first vertex of P that is reachable from a in H using a path that is disjoint from the canonical path S_I) used by a. Since $f \in S_I$ then $f \notin H \setminus S_I$ and so $\mathsf{first}_{\mathsf{H}\setminus\mathsf{S_I}}(a, P)$ is a valid candidate for $\mathsf{first}_{\mathsf{H}\setminus\mathsf{f}}(s, P)$.

(b) The path intersects S_1, so it might as well go through the first vertex u of S_1 (the last vertex of the interval I stored by f). To handle this case, Hence, $\mathsf{first}_{\mathsf{H}\setminus\mathsf{f}}(u, P)$ (stored in item (3c) of the label of f) is also a valid candidate for $\mathsf{first}_{\mathsf{H}\setminus\mathsf{f}}(a, P)$. □

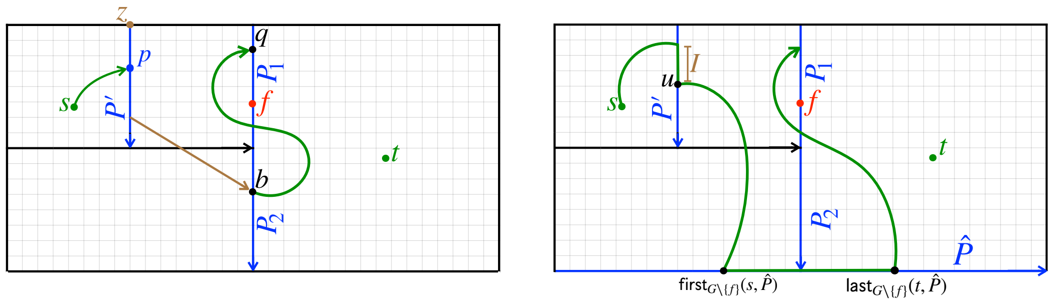


Figure 9: When $f \in Q$: In the left example, the (green) replacement path R touches only $P \in Q$ (the path on which f lies). In this case, b is the first vertex of P_2 that is reachable from s in a path internally disjoint from Q. The path from b to $q = \mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(b, P_1)$ is then found using the mechanism of Section 4.4. In the right example, the (green) replacement path R touches some other path $\hat{P} \neq P$ (on which f does not lie). In this case, using Lemma 4.2 twice we check if $\mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s, \hat{P}) <_{\hat{P}} \mathsf{last}_{\mathsf{G}\setminus\{\mathsf{f}\}}(t, \hat{P})$.

4.3 The $\mathcal{L}^{\mathsf{s}\rightarrow\mathsf{P}\setminus\{\mathsf{f}\}}$ labeling (Lemma 4.3)

To show the labeling scheme $\mathcal{L}^{\mathsf{s}\rightarrow\mathsf{P}\setminus\{\mathsf{f}\}}$ (i.e., prove Lemma 4.3) we will compose the following specialized labeling schemes, which we prove in the sequel.

Lemma 4.6. *There exists a labeling scheme $\mathcal{L}^{\mathsf{P}'\rightarrow\mathsf{P}\setminus\mathsf{f}} = \mathcal{L}^{\mathsf{P}'\rightarrow\mathsf{P}\setminus\mathsf{f}}_{H,P,P'}$ where H is a planar graph with paths P' and P. Such that $P \cap P' = \emptyset$, P has no outgoing edges, and P lies on a single face of H. For $f \in P$ let P_1 and P_2 be the prefix and suffix of P before and after f (without f), respectively. Given the labels of two vertices $a \in P'$ and $f \in P$, one can retrieve the two vertices $b_1 = \mathsf{first}_{\mathsf{H}\setminus\{\mathsf{f}\}}(a, P_1)$ and $b_2 = \mathsf{first}_{\mathsf{H}\setminus\{\mathsf{f}\}}(a, P_2)$. The size of each label is $\tilde{O}(1)$.*

Lemma 4.7. *There exists a labeling scheme $\mathcal{L}^{\mathsf{P}\rightarrow\mathsf{P}\setminus\mathsf{f}} = \mathcal{L}^{\mathsf{P}\rightarrow\mathsf{P}\setminus\mathsf{f}}_{G,P}$ where G is a planar graph, and P is a path of G whose two endpoints lie on the same face. Given the labels of two vertices b and f on P let P_1 and P_2 be the prefix and suffix of P before and after f (without f), respectively. One can compute the indices on P of some vertices $b_1 = \mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(b, P_1)$ and $b_2 = \mathsf{first}_{\mathsf{G}\setminus\{\mathsf{f}\}}(b, P_2)$. The size of each label is $\tilde{O}(1)$.*

We first prove Lemma 4.3 assuming Lemmas 4.6 and 4.7, and then prove the two latter lemmas.

Proof of Lemma 4.3. Let H and P be as in the statement of the lemma. The labeling is obtained by composition of labels via Lemmas 4.2, 4.6 and 4.7. To this end we define auxiliary labeled graphs.

Let H_P be the graph H with the vertices of P labeled with their labels in $\mathcal{L}_{H,P}^{\texttt{P}\to\texttt{P}\backslash\texttt{f}}$ of Lemma 4.7. Let H_P^0 be the graph obtained from H_P by deleting all the edges outgoing from the vertices of P. For a path P' such that P is an ancestor path of P, Let $H_{P,P'}$ be the graph H_P^0 with the vertices of P and P' labeled with their labels in $\mathcal{L}_{H_P^0,P',P}^{\texttt{P}'\to\texttt{P}\backslash\texttt{f}}$ of Lemma 4.6.

The label of a vertex v consists of:

1. For every ancestor piece H' of v in $H_{P,P'}$ such that P is an ancestor path of H', for every boundary path P' of H', v stores $\mathcal{L}_{H',P'}^{\texttt{s}\to\texttt{P}'}(v)$ using Lemma 4.5. (i.e., the label $\mathcal{L}_{H',P'}^{\texttt{s}\to\texttt{P}'}(v)$ contains not only the identity of the desired vertex $a = \mathsf{first}_{\mathsf{H}'}(v, P')$, but also the label of a in $H_{P,P'}$).
2. if v is a vertex of P, v stores:
   (a) For every ancestor path P' of v such that P is an ancestor of P', the $\mathcal{L}_{H_P^0,P',P}^{\texttt{P}'\to\texttt{P}\backslash\texttt{f}}(v)$.
   (b) $\mathcal{L}_{H,P}^{\texttt{P}\to\texttt{P}\backslash\texttt{f}}(v)$.

Size. Since each vertex has $\tilde{O}(1)$ ancestor paths and by Lemmas 4.3, 4.6 and 4.7, the size of the label off each vertex is $\tilde{O}(1)$.

Decoding and Correctness As in the setup of the proof of Lemma 4.2, assume, per the statement of the lemma that P is an ancestor of both s and f and that s and f are not an ancestor apex of one another. Consider the set of leafmost pieces in $\mathcal{T}$ that contain both s and f. Let H'' be such a piece. It must be that H'' is an ancestor piece of both s and f or else one of s and f is an ancestor apex of the other. Hence, there are only $O(1)$ leafmost pieces that contain both s and f. To avoid unnecessary clutter we shall assume there is a unique piece H''. In reality we would have to apply the same argument for all $O(1)$ such pieces. Since H'' is an ancestor piece of both s and f, we can find the piece H'' by traversing the list of ancestors of s (stored in s) and of f (stored in f) until finding the lowest common ancestor. Let H' be the child piece of H'' that contains only s (if H'' is an atomic piece then define $H' = H''$). Since P is an ancestor of both s and f, P is a boundary path of a (possibly weak) ancestor $\hat{H}$ of H''.

We give the algorithm for identifying b_1. The argument for b_2 is identical (just replace b_1 by b_2). Consider a path R from s to b_1 in H. Let b be the first vertex of R that belongs to P. Note that the prefix $R[s, b]$ does not contain any edge outgoing from any vertex of P. Hence, identifying this prefix can be done in the graphs H_P^0 and $H_{P,P'}$ in which all such edges were removed. In other words, we may assume without loss of generality that $b = \mathsf{first}_{\mathsf{H}_\mathsf{P}^0}(s, P)$. Consider the maximal prefix of $R[s, b]$ that is entirely contained in H'. Note that since $f \notin H'$, f is not on this maximal prefix. Let P' be the path of $\partial H'$ at which this maximal prefix of R terminates. Let a be the vertex of P' at which this prefix terminates. We may assume without loss of generality that $a = \mathsf{first}_{\mathsf{H}'}(s, P')$. Furthermore, since $R[s, b]$ visits a, $b = \mathsf{first}_{\mathsf{H}_\mathsf{P}^0}(s, P)$ is also $b = \mathsf{first}_{\mathsf{H}_\mathsf{P}^0}(a, P)$.

We now show that the vertices a, b, b_1 can be retrieved from the labels of s and of f. The vertex a can be retrieved, by Lemma 4.5 from the label $\mathcal{L}_{H',P'}^{\texttt{s}\to\texttt{P}'}(s)$ stored in item (1) of the label of s, along with the label $\mathcal{L}_{H_P^0,P',P}^{\texttt{P}'\to\texttt{P}\backslash\texttt{f}}(a)$. By Lemma 4.6, the label of b in H_P^0 can be obtained from $\mathcal{L}_{H_P^0,P',P}^{\texttt{P}'\to\texttt{P}\backslash\texttt{f}}(a)$ and from $\mathcal{L}_{H_P^0,P',P}^{\texttt{P}'\to\texttt{P}\backslash\texttt{f}}(f)$, which is stored in item (2a) of the label of f. Note that, by definition of H_P^0, the label of b in H_P^0 is $\mathcal{L}_{H,P}^{\texttt{P}\to\texttt{P}\backslash\texttt{f}}(b)$. Finally, from $\mathcal{L}_{H,P}^{\texttt{P}\to\texttt{P}\backslash\texttt{f}}(b)$, and $\mathcal{L}_{H,P}^{\texttt{P}\to\texttt{P}\backslash\texttt{f}}(f)$ (stored in item (2b) of the lable of f), we can obtain, by Lemma 4.7, the identity of b_1 in H. □

We next prove Lemma 4.6, restated here for convenince.

Lemma 4.6. *There exists a labeling scheme $\mathcal{L}^{\mathtt{P'\to P\setminus f}} = \mathcal{L}^{\mathtt{P'\to P\setminus f}}_{H,P,P'}$ where H is a planar graph with paths P' and P. Such that $P \cap P' = \emptyset$, P has no outgoing edges, and P lies on a single face of H. For $f \in P$ let P_1 and P_2 be the prefix and suffix of P before and after f (without f), respectively. Given the labels of two vertices $a \in P'$ and $f \in P$, one can retrieve the two vertices $b_1 = \mathsf{first}_{\mathsf{H}\setminus\{\mathsf{f}\}}(a, P_1)$ and $b_2 = \mathsf{first}_{\mathsf{H}\setminus\{\mathsf{f}\}}(a, P_2)$. The size of each label is $\tilde{O}(1)$.*

Proof of Lemma 4.6. Let a be a vertex of P' and f be a vertex of P as in the statement of the lemma. Let z be the first vertex of P'. For a vertex v, Let $P(v)$ denote the set of vertices of P that are reachable from v in H. Since anything reachable from a is also reachable from z, we have that $P(a) \subseteq P(z)$. In fact, planarity dictates the following:

Claim 4.8. *The set $P(a)$ consists of at most two intervals of consecutive vertices of $P(z)$.*

Proof. Consider two vertices $u, v \in P(a)$ that belong to two disjoint intervals of $P(z)$. Let C be the (undirected) cycle formed by the a-to-v path, the a-to-u path, and $P[u, v]$ (see Figure 10). Since P lies on a single face, no vertices of P other than those of $P[u, v]$ are (even weakly) enclosed by C. The vertex z must be enclosed by C, otherwise, the path from z to any vertex of $P[u, v]$ must intersect the a-to-v path or the a-to-u path (and is thus reachable from p as well in contradiction to the two intervals being disjoint). Since z is enclosed by C, any vertex of P that is reachable from z and does not belong to $P[u, v]$ is also reachable from z (since the path to it from z must intersect the a-to-v path or the a-to-u path). □

Equipped with the structure described in Claim 4.8, the label of a vertex v consists of the following:

1. If v is a vertex of P', v stores:
   (a) $\mathsf{first}_{\mathsf{H}}(v, P)$.
   (b) the identities of the endpoints of the two intervals of $P(v)$ in $P(z)$ (where p and z are as defined above).
2. If v is a vertex of P, v stores the identity of the first vertex u of $P(z)$ that is after v on P' (where z is the first vertex of P').

The size of the label is clearly $O(1)$.

To obtain b_1 we simply check whether $\mathsf{first}_{\mathsf{H}}(a, P)$ (stored in item (1a) of the label of a appears before f on P. If so, since P has no outgoing edges, $\mathsf{first}_{\mathsf{H}}(a, P) = \mathsf{first}_{\mathsf{H}\setminus\mathsf{f}}(a, P_1)$. Otherwise, P_1 is not reachable from a in $H \setminus f$.

Next we describe how to find $b_2 = \mathsf{first}_{\hat{\mathsf{H}}^\circ\setminus\mathsf{Q}}(s, P_2)$. If the vertex u stored in item (2) of the label of f falls inside one of the two intervals stored in item (1b) of the label of a then $b_2 = u$. Otherwise, b_2 is the earliest starting endpoint that is later than f among the two endpoints of the two intervals stored in the label of a (if both intervals are before f on P then b_2 does not exist). □

4.4 The $\mathcal{L}^{\mathtt{P\to P\setminus f}}$ labeling (Lemma 4.7)

In this section we present the secondary labeling scheme (with polylogarithmic-size labels), which addresses the following problem: We are given a directed planar graph G and a single path P in G whose endpoints lie on the same face. We need to label the vertices of P such that given the labels of any two vertices b, f of P we can find the first vertex $p < f$ of P that is reachable from b in $G \setminus \{f\}$ and the first vertex $p > f$ of P that is reachable from b in $G \setminus \{f\}$.

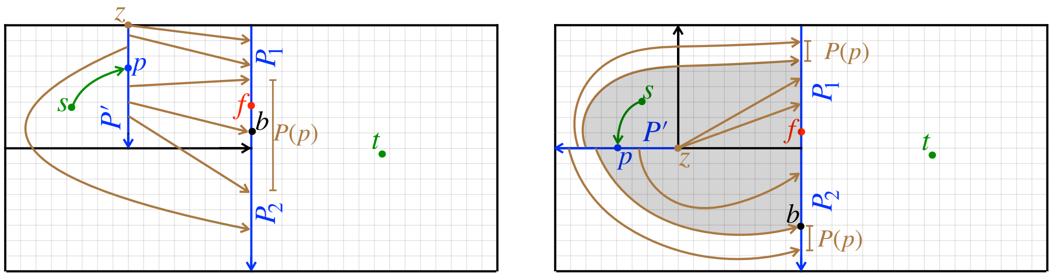


Figure 10: When $f \in Q$: The vertical (blue) separator path P (in this example the separator Q consists of just a single path P) is partitioned by f into P_1 and P_2. The vertex z is the first vertex of p's (blue) path P'. The brown paths are the ways that z can reach P in $\hat{H}^\circ \setminus Q$. The vertex b is the first vertex of P_2 that is reachable from p in $\hat{H}^\circ \setminus Q$. In the left image, f lies inside the interval of vertices $P(p)$ that are reachable from p (b is therefore stored in f). In the right image, f lies outside the two intervals $P(p)$ (and b is therefore stored in s). The shaded area is the cycle C in the proof of Claim 4.8.

An auxiliary procedure

We begin with an auxiliary procedure that will be useful for our labeling. In this procedure, we wish to label the vertices of P, such that given the labels of any two vertices b, f such that f is before b on P (i.e., $f < b$), we can find the first vertex $p > f$ of P that is reachable in G from b using a path that does not touch f or any vertex before f (i.e., does not touch any vertex $v \leq f$ of P).

Let H_P be the graph composed of the path P and the following additional edges: for every pair of vertices $u, v \in P$ where $u > v$, we add an edge (u, v) iff (1) there exists a u-to-v path in G that does not touch P before v, and (2) there is no such w-to-v path for any $w > u$. The following claim shows that instead of working with G, we can work with H_P:

Claim 4.9. *Given any $f < b$, the first vertex $p > f$ that is reachable from b using a path that does not touch f or any vertex of P before f, is the same in G and in H_P.*

Proof. Let S be the corresponding b-to-p path in G for $f < p < b$. The path S visits no vertex of P before f (by definition of S) and no vertex of P between f and p (by definition of p). Hence, in H_P there is an edge (u, p) for some u where $u = b$ or $u > b$, and so S is represented in H_P (by a path composed of a b-to-u prefix along P followed by the single edge (u, p)). In the other direction, let R be the corresponding b-to-p path in H_P. The path R visits no vertex of P before f (by definition of R) and every edge (u, v) of R corresponds to a path in G that visits no vertex of P before v (and therefore visits no vertex of P before f), hence R is appropriately represented in G. □

We call the edges of H_P that are not edges of P *detours*. We will use the fact that detours do not cross (i.e., they form a laminar family):

Claim 4.10. *If there is a detour (u, v) then there is no detour (w, x) with $v < x < u < w$.*

Proof. By concatenating the (w, x) detour, the x-to-u subpath of P, and the (u, v) detour, we get a path in G that does not touch P before v but starts at a vertex $w > u$ contradicting condition (2) of the H_P edges. □

We now explain what needs to be stored to facilitate the auxiliary procedure. We define the *size* $|d|$ of a detour $d = (u, v)$ as the number of vertices of P between v and u. We say that a vertex of P is *contained* in a detour $d = (u, v)$ if it lies on P between v and u. We say that a detour d' is *contained* in detour d if all vertices that are contained in d' are also contained in d. Notice that, from Claim 4.9, the vertex p sought by the auxiliary procedure is an endpoint of the largest detour d that contains b and does not contain f. To find d, every vertex v of P stores a set of $O(\log n)$ nested detours $d^v_1, d^v_2, \ldots$ where d^v_1 is the largest detour containing v, and d^v_{i+1} is the largest detour of size $|d^v_{i+1}| \leq |d^v_i|/2$ containing v. By Claim 4.10, this set of nested detours is well defined. Additionally, for each d^v_i, v stores the largest detour $\hat{d}^v_i$ that is strictly contained in d^v_i but does not contain d^v_{i+1}. See Figure 11.

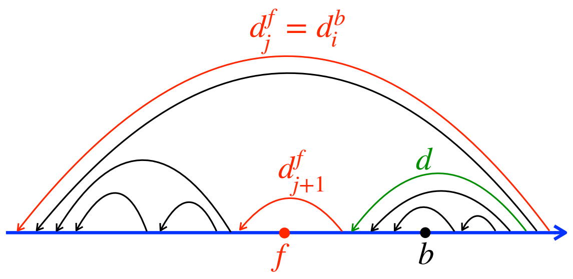


Figure 11: A nested system of $O(\log n)$ detours $d^f_1, d^f_2, \ldots$ of f (in red) used in the auxiliary procedure. The (green) detour d is the largest detour that contains b and does not contain f. Either $d = \hat{d}^f_j$ and is stored in f, or $d = d^b_{i+1}$ and is stored in b.

Consider the smallest detour $d^f_j = d^b_i$ that is saved by both b and f. If such a detour does not exist then there is no detour that contains both b and f and so $d = d^b_1$ is stored in b. If $|d| > |d^f_j|/2$ then d must be the largest detour that is contained in d^f_j but does not contain d^f_{j+1}. In other words, $d = \hat{d}^f_j$ and is stored in f. If on the other hand $|d| \leq |d^f_j|/2$, then $|d| \leq |d^b_i|/2$ and by the choice of d^b_i we have that d^b_{i+1} does not contain f and so $d = d^b_{i+1}$ and is stored in b. This completes the description of the auxiliary procedure.

Description of the $\mathcal{L}^{\texttt{P}\to\texttt{P}\setminus\texttt{f}}$ labeling

Equipped with the above auxiliary procedure, we are now ready to describe the secondary labeling scheme. Recall that there are four cases to consider: Given $b, f \in P$ where b can be before/after f we wish to find the first vertex p before/after f that is reachable from b in $G \setminus \{f\}$. There are four cases to consider:

Given $f < b$, find the first vertex $p < f$ that is reachable from b in $G \setminus \{f\}$. Consider the b-to-p path R in $G \setminus \{f\}$. Let r be the first vertex of R that belongs to P and $r < f$. Let r' be the vertex of P that precedes r on R. Note that $r < f$, that $r' > f$, and that the r'-to-r subpath of R (which we call a *bypass* of f and denote $R[r', r]$) is internally disjoint from P and it either emanates to the left or to the right of P. Moreover, since the endpoints of P lie on the same face, then if R emanates at r' to the left (resp. right) of P it must enter P at r from the left (resp. right) of P. This means that we can assume w.l.o.g that the bypass $R[r', r]$ is the largest such bypass (in terms of the number of vertices of P between r and r'). This is because any smaller bypass that is on the same side of P as $R[r', r]$ is either contained in $R[r', r]$ (in which case we might as well use $R[r', r]$) or intersects $R[r', r]$ implying (in contradiction) that there is a larger bypass. See Figure 12.

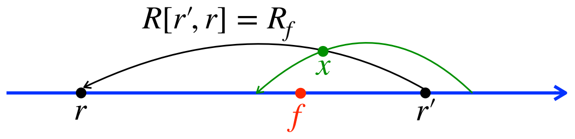


Figure 12: A bypass $R[r', r]$ (in black) of f. If there was an intersecting bypass (in green) that is larger than $R[r', r]$ then the green subpath before x and the black subpath after x would constitute a larger bypass.

We let each vertex $f \in P$ store the endpoints of the largest bypass L_f (resp. R_f) that is internally disjoint from P, emanates left (resp. right) of P at a vertex that is after f, and enters P from the left (resp. right) at a vertex that is before f. The vertex $f \in P$ also stores the first vertex L_f^+ (resp. R_f^+) of P that is before f and is reachable in $G \setminus \{f\}$ from the endpoint of L_f (resp. R_f) that is after f.

In order to find p, we first use the auxiliary procedure of Section 4.4 to find the first vertex $p' > f$ that is reachable in G from b using a path that does not touch any vertex of P before f. Then, we check whether p' is contained in L_f and if so we consider L_f^+ as a candidate for p. Similarly, we check whether p' is contained in R_f and if so we consider R_f^+ as a candidate for p. Finally, we return the earlier of the (at most two) candidates.

Given $f < b$, find the first vertex $p > f$ that is reachable from b in $G \setminus \{f\}$. This case is very similar to the previous case. The only differences are: (1) we let every vertex $f \in P$ store the first vertex L_f^- (resp. R_f^-) of P that is after f and is reachable in $G \setminus \{f\}$ from the endpoints of L_f (resp. R_f), and (2) we add p' itself as a third possible candidate for p.

Given $b < f$, find the first vertex $p > f$ that is reachable from b in $G \setminus \{f\}$. This case and the next one are handled by small (but not symmetric) modifications of the previous two cases. Consider the b-to-p path R in $G \setminus \{f\}$. Let r be the first vertex of R that belongs to P and $r > f$. Let r' be the vertex of P that precedes r on R. The subpath $R[r', r]$ (which we call a *byway* of f) is internally disjoint from P, and if it emanates at r' to the left (resp. right) of P then it must enter r from the left (resp. right) of P. We can assume w.l.o.g that $R[r', r]$ is the smallest such byway (because any larger byway that is on the same side of P as $R[r', r]$ either contains $R[r', r]$ (in which case we might as well use $R[r', r]$) or intersects $R[r', r]$ implying (in contradiction) that there is a smaller byway. See Figure 13.

We let each vertex $f \in P$ store the endpoints of the smallest byway L_f (resp. R_f) containing f that is to the left (resp. right) of P. The vertex $f \in P$ also stores the first vertex L_f^- (resp. R_f^-) of P that is after f and is reachable in $G \setminus \{f\}$ from the endpoint of L_f (resp. R_f) that is after f.

In order to find p, we begin by finding the first vertex $p' < f$ that is reachable in G from b using a path that does not touch any vertex of P after f. This is done by using a variant of the auxiliary procedure of Section 4.4 with the following modification. In H_P we add the detour (u, v) iff (1) there exists a u-to-v path in G that does not touch P before v, and (2) there is no such u-to-w path for any $w < v$. After finding p' using this mechanism, we check whether p' appears on P before the starting point of the byway L_f, and if so we consider L_f^- as a candidate for p. Similarly, we check whether p' appears on P before the starting point of the byway R_f and if so we consider R_f^- as a candidate for p. Finally, we return the earlier of the (at most two) candidates.

Given $b < f$, find the first vertex $p < f$ that is reachable from b in $G \setminus \{f\}$. This case is very similar to the previous case. The only differences are: (1) we let every vertex $f \in P$ store the

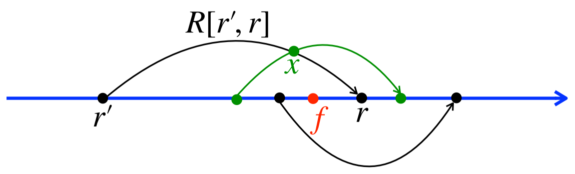


Figure 13: Two byways of f (in black), the top one is $R[r', r]$. If there was an intersecting byway (in green) that is smaller than $R[r', r]$ then the green subpath before x and the black subpath after x would constitute a smaller byway.

first vertex L_f^+ (resp. R_f^+) of P that is before f and is reachable in $G \setminus \{f\}$ from the endpoints of L_f (resp. R_f), and (2) we add p' itself as a third possible candidate for p.

5 Approximate Distance Labeling

In the remainder of this paper, we extend our labeling scheme from reachability to approximate distances. We use standard notation for weighted graphs. For a path P we use $\mathsf{len}(P)$ to denote the total length of P, i.e. the sum of P's edge weights. We also use $\mathsf{dist}_H(x, y)$ to denote the x-to$y$ distance in the graph H.

We begin by describing the scaling approach of Thorup [Tho04]. A $(3, r)$-layered spanning tree T in a digraph H is an unoriented rooted spanning tree (i.e., a rooted spanning tree when ignoring the directions of edges) such that any path in T from the root is the concatenation of at most 3 shortest directed paths in H, each of length at most r. We say that H is a $(3, r)$-layered graph if it has such a spanning tree.

Lemma 5.1. *[Tho04, Lemma 3.2 [7]] Given a directed graph G and a scale r, we can construct in linear time a series of directed graphs $G_1^r, \ldots, G_k^r$, such that:*

1. *The total number of edges and vertices in all G_i^r's is linear in the number of edges and vertices of G.*
2. *Each vertex v of G has an index $i(v)$, such that v does not belong to any G_i^r other than $G_{i(v)}^r$ $G_{i(v)-1}^r$, and $G_{i(v)-2}^r$. Moreover, for any path P of length at most r that starts at v, P is a path in G if and only if P is a path in at least one of the three G_i^r's containing v.*
3. *Each G_i^r is a $(3, r)$-layered graph.*
4. *G_i is a minor of G. That is, G_i is obtained from G by deletion of edges and vertices, and contraction of edges.*

We invoke Lemma 5.1 at exponentially growing scales r (i.e., all powers of 2 up to $2nM$ where M is the largest edge-weight in the graph). To each of the resulting graphs G_i^r, we will apply a labeling scheme for reporting the length of an s-to-t path in $G \setminus \{f\}$ that is with additive error at most $\frac{\varepsilon}{2} \cdot r$ with respect to the length of the shortest such path. By item (2) of the Lemma, the distance reported for one of the three graphs containing s at the scale r satisfying $2\mathsf{dist}_{G\setminus\{f\}}(s, t) \geq r \geq \mathsf{dist}_{G\setminus\{f\}}(s, t)$, is a $(1 + \varepsilon)$ multiplicative approximation of the correct answer, and no distance reported for any graph containing s will be smaller than the correct answer.

[7]The statement of item (3) in [Tho04] only concerns shortest paths, but the proof there actually applies to any path of length at most r, as stated here.

Hence, from now on we focus on describing the labeling scheme with additive error εr for a $(3,r)$-layered directed planar graph. The distance labeling scheme generalizes the reachability labeling scheme, but not in a black box manner. We shall follow the overall structure of the reachability labels. In particular, the graph will be recursively decomposed using fundamental cycle separators, but now we use the the $(3,r)$-layered spanning tree T guaranteed by item (3) in Lemma 5.1. Each separator Q still consists of a constant number of directed paths, but now this constant is larger (the spanning tree we use now is 3-layered rather than 2-layered for reachability), and now these paths are shortest paths, each of length at most r. We further break each of these paths into $O(1/\varepsilon)$ subpaths, each of length at most εr. Hence, from now on we treat every separator Q as a set of $O(1/\varepsilon)$ directed shortest paths of length εr each.

To explain the high level idea of our approach, let us describe how to generalize from reachability to approximate distances in the *non-faulty* case. Assume we just want to approximate the length of a shortest path from s to t under the assumption that this shortest path intersects a particular separator path P. In the case of reachability, we could deduce the answer by comparing $\mathsf{first}_{\mathsf{G}}(s,P)$ and $\mathsf{last}_{\mathsf{G}}(P,t)$. To generalize to approximate distances, let $\mathsf{first}^{\alpha}_{\mathsf{G}}(s,P)$ (resp., $\mathsf{last}^{\alpha}_{\mathsf{G}}(t,P)$) denote the first (resp., last) vertex of P that is reachable from s (resp., can reach t) in G via a path of length at most α. We store $\mathsf{first}^{\mathsf{i}\varepsilon\mathsf{r}}_{\mathsf{G}}(s,P)$ and $\mathsf{last}^{\mathsf{i}\varepsilon\mathsf{r}}_{\mathsf{G}}(t,P)$ for every integer $0 \le i \le 1/\varepsilon$. To answer the query, we find i,j minimizing $(i+j)$ such that $\mathsf{first}^{\mathsf{i}\varepsilon\mathsf{r}}_{\mathsf{G}}(s,P) \le_P \mathsf{last}^{\mathsf{j}\varepsilon\mathsf{r}}_{\mathsf{G}}(t,P)$, and return $(i+j+1)\varepsilon r$ as the estimated distance.

This is correct since $\mathsf{first}^{\mathsf{i}\varepsilon\mathsf{r}}_{\mathsf{G}}(s,P) \le_P \mathsf{last}^{\mathsf{j}\varepsilon\mathsf{r}}_{\mathsf{G}}(t,P)$ implies there exists an s-to-t path of length at most $(i+j+1)\varepsilon r$ (the $+1$ term is for going along the path P whose length is at most εr). To see the approximation guarantee, consider the shortest s-to-t path Γ that intersects P. Let α be the length of the prefix of Γ ending at the earliest vertex of P visited by Γ. Let β be the length of the maximal suffix of Γ that is internally disjoint from P. Then, for $i = \lceil \frac{\alpha}{\varepsilon r} \rceil$, and $j = \lceil \frac{\beta}{\varepsilon r} \rceil$, $\mathsf{first}^{\mathsf{i}\varepsilon\mathsf{r}}_{\mathsf{G}}(s,P) \le_P \mathsf{last}^{\mathsf{j}\varepsilon\mathsf{r}}_{\mathsf{G}}(t,P)$, so we will return at most $(i+j+1)\varepsilon r \le \alpha+\beta+3\varepsilon r$, while the length of Γ is at least $\alpha+\beta$.

We note that Thorup's distance labels use the idea of *portals* (a.k.a ε-covers) which results in smaller labels and more efficient query algorithm. Our more brute force approach incurs polynomial factors in $1/\varepsilon$, but allows us to use the ideas of the fault-tolerant reachability mechanism from the previous sections.

Note, however, that our scheme for fault-tolerant reachability requires in some cases to break the s-to-t path into more than two segments (yet still a constant number of segments), and requires additional modifications as we explain in the rest of this section. In the following sections, some of the text is a straightforward adaptation of the corresponding text for reachability implementing the high-level ideas above. However, in several places (which we will highlight) significant changes and additional ideas are required.

5.1 Fault tolerant approximate distance labeling

In this section, we describe our approximate distances labeling scheme. I.e., what to store in the labels so that given the labels of any three vertices s,f,t we can approximate $\mathsf{dist}_{G\setminus\{f\}}(s,t)$. We follow the structure of the reachability labels from Section 4, and adjust it to the more complicated task of approximate distances. Our main result is the following theorem.

Theorem 5.2. *There exists a labeling scheme for a planar graph G that, given vertices s,t,f returns a $(1+\varepsilon)$-approximation of $\mathsf{dist}_{G\setminus\{f\}}(s,t)$. The size of each label is $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$.*

Before defining the analogous of Lemmas 4.2 and 4.3 from Section 4, we introduce the analog of $\mathsf{first}_{\mathsf{F}}(v,P)$. For a graph F, a vertex v, a path P, and a number α, let $\mathsf{first}^{\alpha}_{\mathsf{F}}(v,P)$ denote the first vertex on P that is reachable from v in F via a path of length at most α. We also present the following relaxation of $\mathsf{first}^{\alpha}_{\mathsf{F}}(u, P)$.

Definition 5.3 (δ-first)**.** *Let F be a directed graph with edge weights from $\mathbb{R}^+$, and let P be a path. A vertex $v \in F$ is δ-$\mathsf{first}^{\alpha}_{\mathsf{F}}(u, P)$ if*

1. *$v \leq_P \mathsf{first}^{\alpha}_{\mathsf{F}}(u, P)$ and,*
2. *$\mathsf{dist}_F(u, v) \leq \alpha + \delta$.*

We follow the reduction into two labeling schemes from a vertex to a path in the presence of a failed vertex, one for the case that the failing vertex is not on the path, and the other for the case that it is.

Lemma 5.4. *There exists a labeling scheme $\mathcal{L}^{\mathtt{s}\xrightarrow{\mathtt{f}}\mathtt{P}} = \mathcal{L}^{\mathtt{s}\xrightarrow{\mathtt{f}}\mathtt{P}}_{G,P,r,\alpha,\varepsilon}$ where G is a planar graph equipped with a decomposition tree $\mathcal{T}$, P is a path in $\mathcal{T}$ with $\mathsf{len}(P) = 0$, and $r, \alpha, \varepsilon \in \mathbb{R}^+$ such that $\alpha \leq r$, and the length of every separator in $\mathcal{T}$ is bounded by r. Let s and $f \notin P$ be two vertices of G that are not an ancestor apex of one another, and such that P is an ancestor path of both s and f. Given the labels of s and f, one can compute the index on P of some vertex b that is εr-$\mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s, P)$. In this labeling scheme, the only vertices that store a label are those that have P as an ancestor. The size of each label stored by such a vertex is $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$.*

Lemma 5.5. *There exists a labeling scheme $\mathcal{L}^{\mathtt{s}\rightarrow\mathtt{P}\setminus\{\mathtt{f}\}} = \mathcal{L}^{\mathtt{s}\rightarrow\mathtt{P}\setminus\{\mathtt{f}\}}_{G,P,r,\alpha,\varepsilon}$ where G is a planar graph equipped with a decomposition tree $\mathcal{T}$, and P is a path in $\mathcal{T}$ such that both endpoints of P lie on the same face of G and $\mathsf{len}(P) = 0$ and $r, \alpha, \varepsilon \in \mathbb{R}^+$ such that $\alpha \leq r$, and the length of every separator in $\mathcal{T}$ is bounded by r. Let s and $f \in P$ be two vertices of G that are not an ancestor apex of one another, and such that P is an ancestor path of both s and f. Given the labels of s and f, one can compute the indices on P of some vertices b_1 and b_2 that are εr-$\mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s, P_1)$ and εr-$\mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s, P_2)$, respectively, where P_1 (resp. P_2) is the prefix (resp. suffix) of P that precedes (resp. follows) f, excluding f. In this labeling scheme, the only vertices that store a label are those that have P as an ancestor path. The size of each label stored by such a vertex is $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$.*

We prove the above two lemmas in Sections 5.2 and 5.3. Given these lemmas, we now prove our main theorem.

Proof of Theorem 5.2. Let G_{rev} be the graph obtained from G by reversing all edges (G_{rev} has exactly the same decomposition tree as G, but the direction of each path is reversed). Let $\varepsilon' = \frac{\varepsilon}{10}$ be an approximation parameter, and for every $r \in \mathbb{R}^+$ let $\Gamma_r = \{i\varepsilon' r \mid i \in [\lceil \frac{1}{\varepsilon'} \rceil]\}$.

Let v be a vertex. For each r that is a power of 2 between 1 and $2nM$, for every i such that $v \in G^r_i$, and for every $\alpha, \beta \in \Gamma_r$ the label of v stores the following:

1. For each piece H in the recursive decomposition $\mathcal{T}$ of G^r_i such that $v \in H \setminus \partial H$, v stores its label in the standard (non-faulty) approximate distance labeling of Thorup for $H \setminus \partial H$.
2. For every ancestor piece H in G^r_i, v stores $\mathsf{first}^{\beta}_{\mathsf{H}}(v, P)$ for each of the $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$ paths P of ∂H.
3. Using Lemma 5.4, for every ancestor path P of v in the recursive decomposition $\mathcal{T}$, v stores $\mathcal{L}^{\mathtt{s}\xrightarrow{\mathtt{f}}\mathtt{P}}_{G',P,r,\alpha,\varepsilon'}(v)$ where G' is obtained from G^r_i by setting $\mathsf{len}(P) = 0$.

4. Using Lemma 5.5, for every ancestor piece H of v in $\mathcal{T}$, for every path P of the separator C of H, v stores $\mathcal{L}^{\mathsf{s}\longrightarrow\mathsf{P}\setminus\{\mathsf{f}\}}_{H^{\times P},P,r,\alpha,\varepsilon'}(v)$, where $H^{\times P}$ is the graph obtained from $H \setminus \partial H$ by making an incision along all the edges of C other than those of P and setting $\mathsf{len}(P) = 0$. Note that, because of the incision, the endpoints of P lie on a single face of $H^{\times P}$, so Lemma 5.5 indeed applies.

5. For every ancestor apex a of v, for every ancestor path P of v, v stores in its label $\mathsf{first}^{\alpha}_{\mathsf{G}^r_i\setminus\{\mathsf{a}\}}(v, P)$, and $\mathsf{first}^{\alpha}_{\mathsf{G}^r_i\setminus\{\mathsf{v}\}}(a, P)$. If a lies on P then let P_2 be the suffix of P after a (not including f). v also stores $\mathsf{first}^{\alpha}_{\mathsf{G}^r_i\setminus\{\mathsf{a}\}}(v, P_2)$. Similarly, if v lies on P then let P_2 be the suffix of P after v (not including v). v also stores $\mathsf{first}^{\alpha}_{\mathsf{G}^r_i\setminus\{\mathsf{v}\}}(a, P_2)$.

6. v additionally stores all the above items in the graph $(G^r_i)_{rev}$ instead of the graph G^r_i.

Size. Each vertex participates in $O(\log Mn)$ graphs G^r_i, and $|\Gamma_r| = O(1/\varepsilon)$. Thus, for each G^r_i such that $v \in G^r_i$ and for every $\alpha, \beta \in \Gamma_r$, v has only $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$ ancestor pieces, paths and apices, and by Lemmas 5.4 and 5.5, all items above sum up to a label of size $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$.

Decoding and Correctness. Let R be a shortest path from s to t in $G \setminus \{f\}$. By the second property of Lemma 5.1, there exists a graph G^r_i such that R is contained in G^r_i. If $f \notin G^r_i$ we query Thorup's non-faulty approximate distances labels for G^r_i (stored in item (1)). This will output a $(1 + \varepsilon')$-approximation of $\mathsf{dist}_{G^r_i}(s, t) = \mathsf{dist}_{G\setminus\{f\}}(s, t)$. Otherwise, let $\hat{H}$ be the rootmost piece in $\mathcal{T}$ whose separator Q separates t and f. Let H be a child piece of $\hat{H}$ that contains t (if both children of $\hat{H}$ contain t then, if one of the children does not contain f we choose H to be that child). Note that by choice of H, $f \notin H \setminus \partial H$.

We assume without loss of generality that $s \in \hat{H}$. We handle the other case symmetrically to the description below, by swapping the roles of s and t and working in $(G^r_i)_{rev}$ instead of in G^r_i.

Observe that by definition of H and of separation, $f \in \partial H$ iff $f \in Q$. Consider first the case when R does not touch ∂H. i.e., s, t and R are all contained in $H \setminus \partial H$, and f is not contained in $H \setminus \partial H$. In this case, querying Thorup's non-faulty labels for $H \setminus \partial H$ (stored in item (1)) will output a $(1 + \varepsilon')$-approximation of $\mathsf{dist}_{H\setminus\partial H}(s, t) = \mathsf{dist}_{G\setminus\{f\}}(s, t)$.

To treat the case where the replacement path R touches ∂H, we separately handle the cases where $f \notin Q$ and $f \in Q$.

When $f \notin Q$ (and so, $f \notin \partial H$). In this case, R must have a suffix contained in H, and this suffix is unaffected by the fault f. Let a be the last vertex on R which is on Q, and let α and β be the smallest numbers in Γ_r with $\alpha \geq \mathsf{len}(R[s, a])$ and $\beta \geq \mathsf{len}(R[a, t])$. Let P be the subpath of Q that contains a. If R exists then $\mathsf{first}^{\alpha}_{\mathsf{G}^r_i\setminus\{\mathsf{f}\}}(s, P) <_P \mathsf{first}^{\beta}_{\mathsf{H}_{\mathsf{rev}}}(t, P)$. The vertex $\mathsf{first}^{\beta}_{\mathsf{H}_{\mathsf{rev}}}(t, P)$ is stored in item (2) of the label of t in $(G^r_i)_{rev}$. Notice that by the rootmost choice of $\hat{H}$, H is an ancestor piece of t, so t indeed stores $\mathsf{first}^{\beta}_{\mathsf{H}_{\mathsf{rev}}}(t, P)$. It thus remains only to describe how to find $\mathsf{first}^{\alpha}_{\mathsf{G}^r_i\setminus\{\mathsf{f}\}}(s, P)$ from the labels of s and f. If either s or f store $\mathsf{first}^{\alpha}_{\mathsf{G}^r_i\setminus\{\mathsf{f}\}}(s, P)$ in item (5), we are done. Otherwise neither s nor f is an ancestor apex of the other, and since both s and f are in $\hat{H}$, P is indeed an ancestor path of both s and f, so, by Lemma 5.4, a vertex a' that is an $\varepsilon'$$\mathsf{r}$-$\mathsf{first}^{\alpha}_{\mathsf{G}^r_i\setminus\{\mathsf{f}\}}(s, P)$ can be obtained from $\mathcal{L}^{\mathsf{s}\xrightarrow{\mathsf{f}}\mathsf{P}}_{G',P,r,\alpha,\varepsilon'}(s)$ (stored in item (3) of the label of s) and $\mathcal{L}^{\mathsf{s}\xrightarrow{\mathsf{f}}\mathsf{P}}_{G',P,r,\alpha,\varepsilon'}(f)$ (stored in item (3) of the label of f). The decoding algorithm will recognize that indeed $a' \leq_P \mathsf{first}^{\beta}_{\mathsf{H}_{\mathsf{rev}}}(t, P)$ and deduce that there exists a path of length at most $(\alpha+\varepsilon' r)+\varepsilon' r+\beta$ from s to t in $G\setminus\{f\}$. Thus, the distance we output is

$$\begin{aligned}(\alpha+\varepsilon' r)+\varepsilon' r+\beta &\le (\alpha+\beta)+2\varepsilon' r\\ &\le |R|+4\varepsilon' r\\ &\le |R|+8\varepsilon'|R|\\ &\le (1+\varepsilon)|R| = (1+\varepsilon)\mathsf{dist}_{G\setminus\{f\}}(s,t).\end{aligned}$$

When $f\in Q$. Let P be the path of Q that contains f. Consider first the case where the path R touches some path $\hat{P}\neq P$ of Q or of the boundary of some ancestor of H. Since boundary paths are vertex disjoint, $f\in P$ implies $f\notin\hat{P}$. Hence, we can obtain a vertex a' that is an $\varepsilon'\mathsf{r\text{-}first}^{\alpha}_{\mathsf{G^r_i}\setminus\{\mathsf{f}\}}(s,\hat{P})$ in a similar manner to the case $f\notin Q$ above, with $\hat{P}$ taking the role of P. In an analogous manner, we can obtain a vertex b' that is a $\varepsilon'\mathsf{r\text{-}first}^{\alpha}_{(\mathsf{G^r_i})_{\mathsf{rev}}\setminus\{\mathsf{f}\}}(t,\hat{P})$ as we just found a', but with $\hat{P}$ taking the role of P, t taking the role of s, and $(G^r_i)_{rev}$ taking the role of G^r_i (we cannot use part (2) of the label of s or t in this case because now f does belong to ∂H).

Now consider the case where other than P, R does not touch any path of Q or any path of the boundary of an ancestor of H. In this case, it is valid to use labels in $\hat{H}^{\times P}$ instead of in G^r_i because R does touch $\partial\hat{H}$, and only crosses Q at P. Let P_1 and P_2 be the prefix and suffix obtained from P by deleting f. If either s or f is an ancestor apex of one another then $\mathsf{first}^{\alpha}_{\mathsf{G^r_i}\setminus\{\mathsf{f}\}}(s,P_2)$ is stored in item (5) of either s or t, and, if $\mathsf{first}^{\alpha}_{\mathsf{G^r_i}\setminus\{\mathsf{f}\}}(s,P_1)$ exists, then it is equal to $\mathsf{first}^{\alpha}_{\mathsf{G^r_i}\setminus\{\mathsf{f}\}}(s,P)$, which is also stored in item (5) of either s or t. (If $\mathsf{first}^{\alpha}_{\mathsf{G^r_i}\setminus\{\mathsf{f}\}}(s,P)$ is not a vertex of P_1 then $\mathsf{first}^{\alpha}_{\mathsf{G^r_i}\setminus\{\mathsf{f}\}}(s,P_1)$ does not exist.) If neither s nor t is an ancestor apex of the other, then P is an ancestor path of both s,t and f. Let a be the last vertex on R which is on P, and let α and β be the smallest numbers in Γ_r where $\alpha\ge\mathsf{len}(R[s,a])$ and $\beta\ge\mathsf{len}(R[a,t])$. We use the labels $\mathcal{L}^{\mathsf{s}\to\mathsf{P}\setminus\{\mathsf{f}\}}_{\hat{H}^{\times P},P,r,\alpha,\varepsilon'}(s)$, $\mathcal{L}^{\mathsf{s}\to\mathsf{P}\setminus\{\mathsf{f}\}}_{\hat{H}^{\times P},P,r,\alpha,\varepsilon'}(f)$, $\mathcal{L}^{\mathsf{s}\to\mathsf{P}\setminus\{\mathsf{f}\}}_{(\hat{H}^{\times P})_{rev},P,r,\beta,\varepsilon'}(f)$ and $\mathcal{L}^{\mathsf{s}\to\mathsf{P}\setminus\{\mathsf{f}\}}_{(\hat{H}^{\times P})_{rev},P,r,\beta,\varepsilon'}(t)$ to obtain the following vertices:

1. a_1 which is an $\varepsilon'\mathsf{r\text{-}first}^{\alpha}_{\hat{\mathsf{H}}^{\times\mathsf{P}}}(s,P_1)$
2. a_2 which is an $\varepsilon'\mathsf{r\text{-}first}^{\alpha}_{\hat{\mathsf{H}}^{\times\mathsf{P}}}(s,P_2)$,
3. b_1 which is an $\varepsilon'\mathsf{r\text{-}first}^{\beta}_{(\hat{\mathsf{H}}^{\times\mathsf{P}})_{\mathsf{rev}}}(s,P_1)$
4. b_2 which is an $\varepsilon'\mathsf{r\text{-}first}^{\beta}_{(\hat{\mathsf{H}}^{\times\mathsf{P}})_{\mathsf{rev}}}(s,P_2)$

The decoding algorithm will recognize that indeed we have $a_1\le_P b_1$ (if $a\in P_1$) or $a_2\le_P b_2$ (if $a\in P_2$) and deduce that there exists a path of length at most $(\alpha+\varepsilon' r)+\varepsilon' r+(\beta+\varepsilon' r)$ from s to t in $G\setminus\{f\}$. Thus, the distance we output is

$$\begin{aligned}(\alpha+\varepsilon' r)+\varepsilon' r+(\beta+\varepsilon' r) &\le (\alpha+\beta)+3\varepsilon' r\\ &\le |R|+5\varepsilon' r\\ &\le |R|+10\varepsilon'|R|\\ &\le (1+\varepsilon)|R| = (1+\varepsilon)\mathsf{dist}_{G\setminus\{f\}}(s,t).\end{aligned}$$

Notice that the decoding algorithm does not know a-priori the correct values of G^r_i, α and β. Thus, the algorithm iterates over all possible values and returns the minimum distance found. Clearly, every distance found for some values implies a path in $G\setminus\{f\}$, thus the algorithm will always return at least $\mathsf{dist}_{G\setminus\{f\}}(s,t)$, and as we proved it will always be at most $(1+\varepsilon)\mathsf{dist}_{G\setminus\{f\}}(s,t)$. □

5.2 The $\mathcal{L}^{\mathsf{s}\xrightarrow{\mathsf{f}}\mathsf{P}}$ labeling

In this section we provide a labeling scheme $\mathcal{L}^{\mathsf{s}\xrightarrow{\mathsf{f}}\mathsf{P}}$, proving Lemma 5.4. Our labeling scheme makes use of the following two auxiliary labeling schemes:

Lemma 5.6. *There exists a (trivial) labeling scheme $\mathcal{L}^{\mathsf{s}\to\mathsf{P}'} = \mathcal{L}^{\mathsf{s}\to\mathsf{P}'}_{H,P,\alpha}$ where H is a planar graph and P is a 0-length path. Given the label of a vertex a, one can retrieve the vertex $\mathsf{first}^{\alpha}_{\mathsf{H}}(a,P)$. The size of each label is $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$.*

Lemma 5.7. *There exists a labeling scheme $\mathcal{L}^{\mathsf{P}'\xrightarrow{\mathsf{f}}\mathsf{P}} = \mathcal{L}^{\mathsf{P}'\xrightarrow{\mathsf{f}}\mathsf{P}}_{H,P,P',\alpha,\varepsilon}$ where H is a planar graph, P and P' are two 0-length paths and $\alpha,\varepsilon \in \mathbb{R}^+$. Given the labels of a vertex a on P' and a vertex f not in $P' \cup P$, one can retrieve the index on P of some vertex $b \in P$ such that b is an $\varepsilon\alpha$-$\mathsf{first}^{\alpha}_{\mathsf{H}\setminus\mathsf{f}}(a,P)$. The size of each label is $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$.*

The proof of Lemma 5.6 is trivial and the proof of Lemma 5.7 appears in Section 7.

Conceptually, Lemma 5.4 is obtained by composing Lemmas 5.6 and 5.7, as we describe in the following overview. However, the concept of composing two label schemes introduces several technical details, making the proof presented below appear more intricate.

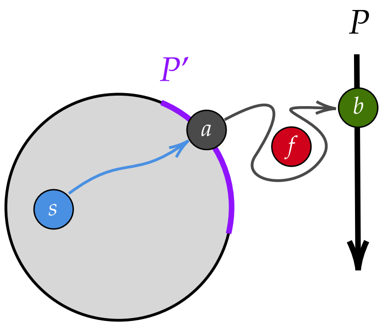


Figure 14: An lustration of a path from s to $b = \mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s,P)$. The blue subpath is from s to $a \in P'$, and the gray subpath is from a to $b \in P$. We use $\mathcal{L}^{\mathsf{s}\to\mathsf{P}'}$ to find a good middle point for a on P', and $\mathcal{L}^{\mathsf{P}'\xrightarrow{\mathsf{f}}\mathsf{P}}$ to find a good middle point for b on P.

Overview (see Figure 14). Let R be a shortest path from s to $b = \mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s,P)$ in $G \setminus \{f\}$, and let Q be a separator that separates s and f. We are interested in a special vertex on R: the first vertex a^* on R that is in Q, and specifically on some P' which is an εr-subpath of Q. We 'guess' estimations β,γ for the lengths of $R[s,a^*]$ and $R[a^*,b]$, respectively. In this overview, it is useful to consider a vertex m that is εr-$\mathsf{first}^{\beta}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s,P)$ as a sufficiently good 'middle point. Specifically, if one aims to reach an early vertex on P from s within a budget α, requiring the path to pass through m and allowing the budget to exceed α by εr does not worsen the result on P. Using the labels $\mathcal{L}^{\mathsf{s}\to\mathsf{P}'}$ with budget β, we find a sufficiently good 'middle point' a for replacing a^*. Starting from a, we use the labels $\mathcal{L}^{\mathsf{P}'\xrightarrow{\mathsf{f}}\mathsf{P}}$ with budget γ to find a vertex earlier on P than $\mathsf{first}^{\gamma}_{\mathsf{G}\setminus\{\mathsf{f}\}}(a,P) \le_P \mathsf{first}^{\gamma}_{\mathsf{G}\setminus\{\mathsf{f}\}}(a^*,P)$.

Since each middle point is reachable with a budget increase of $O(\varepsilon r)$, the final point is reachable with a budget increase of $O(\varepsilon r)$. Since every middle point is 'reasonably good', and the budget used in each phase is an estimation of the length of the corresponding subpath of R, it is guaranteed that the final destination is at least as good as b.

We are now ready to provide the formal proof of Lemma 5.4.

Lemma 5.4. *There exists a labeling scheme* $\mathcal{L}^{\mathsf{s}\xrightarrow{\mathsf{f}}\mathsf{P}} = \mathcal{L}^{\mathsf{s}\xrightarrow{\mathsf{f}}\mathsf{P}}_{G,P,r,\alpha,\varepsilon}$ *where G is a planar graph equipped with a decomposition tree $\mathcal{T}$, P is a path in $\mathcal{T}$ with* $\mathsf{len}(P) = 0$, *and $r, \alpha, \varepsilon \in \mathbb{R}^+$ such that $\alpha \le r$, and the length of every separator in $\mathcal{T}$ is bounded by r. Let s and $f \notin P$ be two vertices of G that are not an ancestor apex of one another, and such that P is an ancestor path of both s and f. Given the labels of s and f, one can compute the index on P of some vertex b that is* $\varepsilon r\text{-}\mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s,P)$. *In this labeling scheme, the only vertices that store a label are those that have P as an ancestor. The size of each label stored by such a vertex is* $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$.

Proof. Let $\varepsilon' = \frac{\varepsilon}{4} \in \Theta(\varepsilon)$ be an approximation parameter and let $\Gamma = \{i\varepsilon'\alpha \mid i \in [\lceil \frac{1}{\varepsilon'} \rceil]\}$. For a subpath P' of a separator in the fully recursive decomposition of G, we denote as $G^0_{P'}$ the graph obtained from G by setting the weight of every edge in P' to be 0. For every $\gamma \in \Gamma$ let $G^1_{P',\gamma}$ be the graph G where the label of each vertex v is set to be $\mathcal{L}^{\mathsf{P'}\xrightarrow{\mathsf{f}}\mathsf{P}}_{G^0_{P'},P,P',\gamma,\varepsilon'}(v)$. We also define $G^2_{P',\gamma}$ as the subgraph of $G^1_{P',\gamma}$ induced only on vertices below the separator of P' in the fully recursive decomposition of G.

The Labeling. For every vertex v such that P is an ancestor of v in $\mathcal{T}$, the label of v stores for every triplet (P', β, γ) such that P' is an $\varepsilon' r$-subpath ancestor of v and $\beta, \gamma \in \Gamma$, the label $\mathcal{L}^{\mathsf{s}\to\mathsf{P'}}_{G^2_{P',\gamma},P',\beta}(v)$ and the label of v in $G^1_{P',\gamma}$. Moreover, the label of v stores the identifiers of the ancestor pieces of v in the full recursive decomposition of G. Finally, for every ancestor separator Q of v below P, the label of v stores $b^{\beta}_{Q} = \mathsf{first}^{\beta}_{\mathsf{G'_Q}}(v,P)$ where G'_Q is the subgraph below the separator of Q, and the label $\mathcal{L}^{\mathsf{P'}\xrightarrow{\mathsf{f}}\mathsf{P}}_{G,P,P,\gamma,\varepsilon'}(b^{\beta}_{Q})$. [8]

Size. There are $O(\log n)$ ancestor separators of v in the fully recursive decomposition. Each separator is partitioned into $O(\frac{1}{\varepsilon})$ subpaths, so overall there are $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$ options for P' for each vertex v. It follows from $|\Gamma| = O(\frac{1}{\varepsilon})$ that there are $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$ triplets (P', β, γ). For every such triplet, the label stores an $\mathcal{L}^{\mathsf{s}\to\mathsf{P'}}$-type label in $G^2_{P',\gamma}$ and the label of a vertex in $G^2_{P',\gamma}$. Each vertex in $G^2_{P',\gamma}$ is labeled with a $\mathcal{L}^{\mathsf{P'}\xrightarrow{\mathsf{f}}\mathsf{P}}$-type label, which is of size $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$ due to Lemma 5.7. Due to Lemma 5.6, the size of $\mathcal{L}^{\mathsf{s}\to\mathsf{P'}}$ is $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$, which should be multiplied by another $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$ factor due to vertices of $G_{P',\gamma}$ each having labels of size $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$. Finally, the height of the recursive decomposition is logarithmic, so storing the identifiers of ancestor pieces of v and storing b^{β}_{Q} and a $\mathcal{L}^{\mathsf{P'}\xrightarrow{\mathsf{f}}\mathsf{P}}$-type label of b^{β}_{Q} per ancestor separator Q of v increases the size of the label by $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$.

Decoding. Given the labels of two vertices s and f such that P is an ancestor of s and f, and none of s and f is an apex ancestor of the other in $\mathcal{T}$. The algorithm finds an $\varepsilon r\text{-}\mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s,P)$ vertex as follows. The algorithm starts by finding the highest separator Q in the recursive decomposition that has s in one side, and f strictly in the other side. The label of s stores $b^{\beta}_{Q} = \mathsf{first}^{\alpha}_{\mathsf{G'_Q}}(s,P)$ and $\mathcal{L}^{\mathsf{P'}\xrightarrow{\mathsf{f}}\mathsf{P}}_{G,P,P,\gamma,\varepsilon}(b^{\beta}_{Q})$ for every $\beta, \gamma \in \Gamma$. The label of f stores $\mathcal{L}^{\mathsf{P'}\xrightarrow{\mathsf{f}}\mathsf{P}}_{G,P,P,\gamma,\varepsilon}(f)$. We use the two labels to obtain $b^{\beta,\gamma}_{Q}$ that is a $\varepsilon'\gamma\text{-}\mathsf{first}^{\gamma}_{\mathsf{G}\setminus\{\mathsf{f}\}}(b^{\beta}_{Q},P)$.

[8]Notice that $\mathsf{first}^{\alpha}_{\mathsf{G'_Q}}(v,P)$ is a slight abuse of notation, since P is not contained in G'_Q. Note that the original definition of this notation can be applied similarly even if P is only partially contained in the graph. In this context, $\mathsf{first}^{\alpha}_{\mathsf{G'_Q}}(v,P)$ is the first vertex of P obtainable from v via a path in G'_Q with length at most α

Recall that Q is partitioned into $O(\frac{1}{\varepsilon})$ subpaths with length at most $\varepsilon' r$ each. For each such subpath P', and for every $\beta, \gamma \in \Gamma$ such that $\beta + \gamma \leq (1+2\varepsilon')\alpha$, the algorithm uses $\mathcal{L}^{\mathsf{s} \to \mathsf{P}'}_{G^2_{P',\gamma}, P', \beta}(s)$ to find $a_{P',\beta} = \mathsf{first}^{\beta}_{\mathsf{G}'_{\mathsf{Q}}}(s, P')$ where G'_Q is the subgraph induced by vertices below Q in the fully recursive decomposition. More precisely, the label $\mathcal{L}^{\mathsf{s} \to \mathsf{P}'}_{G^2_{P',\gamma}, P', \beta}(s)$ allows the algorithm to retrieve $\ell_a = \mathcal{L}^{\mathsf{P}' \xrightarrow{\mathsf{f}} \mathsf{P}}_{G^0_{P'}, P, P', \gamma, \varepsilon'}(a_{P',\beta})$. Note that, the label of f contains $\ell_f = \mathcal{L}^{\mathsf{P}' \xrightarrow{\mathsf{f}} \mathsf{P}}_{G^0_{P'}, P, P', \gamma, \varepsilon'}(f)$. The algorithm uses the labels ℓ_a and ℓ_f to retrieve a vertex $b_{P',\beta,\gamma}$ on P that is an $\varepsilon'\gamma$-$\mathsf{first}^{\gamma}_{\mathsf{G}^0_{\mathsf{P}'} \setminus \{\mathsf{f}\}}(a_{P',\beta}, P)$. Finally, the algorithm returns the vertex $b = \min_{\leq P}\{b_{P',\beta,\gamma}, b^{\beta,\gamma}_Q \mid \beta, \gamma \in \Gamma, P' \textit{ is a subpath of } Q, \beta + \gamma \leq (1+2\varepsilon')\alpha\}$. That is, the first vertex on P that was found over all choices of P', β and γ, or the first $b^{\beta,\gamma}_Q$ found on P.

Correctness. We show that when decoding the labels of two vertices s and f, we indeed return a vertex b that is an $\varepsilon \mathsf{r}$-$\mathsf{first}^{\alpha}_{\mathsf{G} \setminus \{\mathsf{f}\}}(s, P)$. We start by showing that for every candidate x found by the decoding algorithm for b, we have $\mathsf{dist}_{G \setminus \{f\}}(s, x) \leq \alpha + \varepsilon r$. Let Q be the lowest separator in the decomposition such that s is in one side of Q and f is strictly in the other side. Considered a vertex $b^{\beta,\gamma}_Q$ that is a candidate for being b returned by the decoding algorithm. Recall that $b^{\beta,\gamma}_Q$ is an $\varepsilon'\gamma$-$\mathsf{first}^{\gamma}_{\mathsf{G} \setminus \{\mathsf{f}\}}(b^{\beta}_Q, P)$ with $b^{\beta}_Q = \mathsf{first}^{\beta}_{\mathsf{G}'_{\mathsf{Q}}}(s, P)$. By definition, we have $\mathsf{dist}_{G'_Q}(s, b^{\beta}_Q) \leq \beta$ and since $f \notin G'_Q$ we have $\mathsf{dist}_{G \setminus \{f\}}(s, b^{\beta}_Q) \leq \beta$. By definition, $\mathsf{dist}_{G \setminus \{f\}}(b^{\beta}_Q, b^{\beta,\gamma}_Q) \leq (1+\varepsilon')\gamma$. By triangle inequality we have $\mathsf{dist}_{G \setminus \{f\}}(s, b^{\beta,\gamma}_Q) \leq \beta + \gamma + \varepsilon'\gamma \leq \alpha + 2\varepsilon'\alpha + \varepsilon'\alpha \leq (1+\varepsilon)\alpha$.

We now deal with candidates obtained as $b_{P',\beta,\gamma}$. Recall that $b_{P',\beta,\gamma}$ is an $\varepsilon'\gamma$-$\mathsf{first}^{\gamma}_{\mathsf{G}^0_{\mathsf{P}'} \setminus \{\mathsf{f}\}}(a_{P',\beta}, P)$ with $a_{P',\beta} = \mathsf{first}^{\beta}_{\mathsf{G}'_{\mathsf{Q}}}(s, P')$. Since $f \notin G'_Q$ we have $\mathsf{dist}_{G \setminus \{f\}}(s, a_{P',\beta}) \leq \mathsf{dist}_{G'_Q}(s, a_{P',\beta}) \leq \beta$. Since $b_{P',\beta,\gamma}$ is $\varepsilon'\gamma$-$\mathsf{first}^{\gamma^*}_{\mathsf{G}^0_{\mathsf{P}'} \setminus \{\mathsf{f}\}}(a_{P',\beta}, P)$, we have $\mathsf{dist}_{G^0_{P'} \setminus \{f\}}(a_{P',\beta}, b_{P',\beta,\gamma}) \leq \gamma + \varepsilon'\gamma$. Since $G^0_{P'}$ is obtained from G by reducing the weight of all edges of P' to 0, and since the total length of P' in G is at most $\varepsilon' r$, we have that $\mathsf{dist}_{G \setminus \{f\}}(a_{P',\beta}, b_{P',\beta,\gamma}) \leq \gamma + \varepsilon'\gamma + \varepsilon' r$. From triangle inequality we get $\mathsf{dist}_{G \setminus \{f\}}(s, b_{P',\beta,\gamma}) \leq \beta + \gamma + \varepsilon'\gamma + \varepsilon' r \leq \alpha + \varepsilon'(2\alpha + \gamma + r) \leq \alpha + 4\varepsilon' r = \alpha + \varepsilon r$ as required.

We are now left with the task of proving $b \leq_P \mathsf{first}^{\alpha}_{\mathsf{G} \setminus \{\mathsf{f}\}}(s, P)$. Let $b^* = \mathsf{first}^{\alpha}_{\mathsf{G} \setminus \{\mathsf{f}\}}(s, P)$ and let R be a shortest path from s to b^* in $G \setminus \{f\}$. Let a^* be the first vertex on R that is also in Q. If there is no a^*, let b' be the first vertex on R that is on P and let $\beta^* = \min\{x \in \Gamma \mid x \geq \mathsf{len}(R[s, b']\}$ and $\gamma^* = \min\{x \in \Gamma \mid x \geq \mathsf{len}(R[b', b^*]\}$. Note that $\beta^* + \gamma^* \leq (1+2\varepsilon')\alpha$ and therefore the decoding algorithm computed some vertex b^{β^*,γ^*}_Q as a candidate for b. Recall that b^{β^*,γ^*}_Q is an $\varepsilon'\gamma$-$\mathsf{first}^{\gamma}_{\mathsf{G} \setminus \{\mathsf{f}\}}(b^{\beta^*}_Q, P)$ vertex for $b^{\beta^*}_Q = \mathsf{first}^{\beta^*}_{\mathsf{G}'_{\mathsf{Q}}}(s, P)$. Note that $R[s, b']$ is a path with length at most β^* that is completely contained in G'_Q. Therefore, $b^{\beta^*}_Q = \mathsf{first}^{\beta^*}_{\mathsf{G}'_{\mathsf{Q}}}(s, P) \leq_P b'$. Since $P[b^{\beta^*}_Q, b'] \cdot R[b', b^*]$ is a path of length at most γ^* in $G \setminus \{f\}$, we also have $b^{\beta^*,\gamma^*}_Q \leq_P \mathsf{first}^{\gamma^*}_{\mathsf{G} \setminus \{\mathsf{f}\}}(b', P) \leq_P b^*$. This concludes the case where a^* does not exist, since we have $b \leq_P b^*$ due to b being the $\leq_P$ minimum in a set containing b^{β^*,γ^*}_Q.

We now treat the case where a^* exists. Let P' be the subpath of Q that contains a^*. Let $\beta^* = \min\{x \in \Gamma \mid x \geq \mathsf{len}(R[s, a^*])\}$ and $\gamma^* = \min\{x \in \Gamma \mid x \geq \mathsf{len}(R[a^*, b^*])\}$, and notice that $\beta^* + \gamma^* \leq \mathsf{len}(R) + 2\varepsilon'\alpha \leq (1+2\varepsilon')\alpha$. Therefore, the decoding algorithm computed some b_{P',β^*,γ^*} that is $\varepsilon'\gamma^*$-$\mathsf{first}^{\gamma^*}_{\mathsf{G}^0_{\mathsf{P}'} \setminus \{\mathsf{f}\}}(a_{P',\beta^*}, P)$ with $a_{P',\beta^*} = \mathsf{first}^{\beta^*}_{\mathsf{G}'_{\mathsf{Q}}}(s, P')$.

Since $R[s, a^*]$ is a path from s to P' in G'_Q of length at most β^*, we have $a_{P',\beta^*} \leq_{P'} a^*$. Since $P'[a_{P',\beta^*}, a^*] \cdot R[a^*, b^*]$ is a path from a_{P',β^*} to b^* in $G^0_{P'} \setminus \{f\}$ with length at most γ^* (recall that the length of P' in $G^0_{P'}$ is zero), we have $b_{P',\beta^*,\gamma^*} \leq_P b^*$. Due to the minimality of the returned vertex b on P across all choices of P', β and γ, we have that $b \leq_P b_{P',\beta^*,\gamma^*} \leq_P b^*$ as required. $\square$

5.3 The $\mathcal{L}^{\mathsf{s}\to\mathsf{P}\setminus\{\mathsf{f}\}}$ labeling

In this section we provide a labeling scheme $\mathcal{L}^{\mathsf{s}\to\mathsf{P}\setminus\{\mathsf{f}\}}$, proving Lemma 5.5. Our labeling scheme makes use of Lemma 5.6 and the following two auxiliary labeling schemes:

Lemma 5.8. *There exists a labeling scheme $\mathcal{L}^{\mathsf{P}\to\mathsf{P}\setminus\mathsf{f}} = \mathcal{L}^{\mathsf{P}\to\mathsf{P}\setminus\mathsf{f}}_{G,P,\alpha,\varepsilon}$ where G is a planar graph, P is a 0-length path of G whose two endpoints lie on the same face and $\alpha, \varepsilon \in \mathbb{R}^+$. Given the labels of two vertices b and f on P let P_1 and P_2 be the prefix and suffix of P before and after f (without f), respectively. One can compute the indices on P of some vertices b_1 and b_2 that are $\varepsilon\alpha$-$\mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(b, P_1)$ and $\varepsilon\alpha$-$\mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(b, P_2)$, respectively. The size of each label is $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$.*

Lemma 5.9. *There exists a labeling scheme $\mathcal{L}^{\mathsf{P}'\to\mathsf{P}\setminus\mathsf{f}} = \mathcal{L}^{\mathsf{P}'\to\mathsf{P}\setminus\mathsf{f}}_{H,P,P',\alpha,\varepsilon}$ where H is a planar graphs H with a 0-length path P' and a path P without outgoing edges which lies on a single face, such that $P \cap P' = \emptyset$, and $\alpha, \varepsilon \in \mathbb{R}^+$. For $f \in P$ let P_1 and P_2 be the prefix and suffix of P before and after f (without f), respectively. Given the labels of two vertices $a \in P'$ and $f \in P$, one can retrieve two vertices b_1 and b_2 which are $\varepsilon\alpha$-$\mathsf{first}^{\alpha}_{\mathsf{H}\setminus\{\mathsf{f}\}}(a, P_1)$ and $\varepsilon\alpha$-$\mathsf{first}^{\alpha}_{\mathsf{H}\setminus\{\mathsf{f}\}}(a, P_2)$, respectively. The size of each label is $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$.*

The proof of Lemma 5.8 appears in Section 6 and the proof of Lemma 5.9 appears in Section 7. We note that given Lemmas 5.6, 5.8 and 5.9 the label of $\mathcal{L}^{\mathsf{s}\to\mathsf{P}\setminus\{\mathsf{f}\}}$ is conceptually simple as we explain in the following high-level overview. Due to certain technical details, the complete proof presented below appears more intricate.

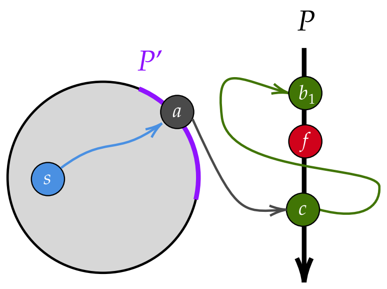


Figure 15: An lustration of a path from s to $b_1 = \mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s, P_1)$. The blue subpath is from s to $a \in P'$, and the gray subpath is from a to $c \in P_2$ and is internally disjoint from P, and the green subpath is from c to $b_1 \in P_1$. We use $\mathcal{L}^{\mathsf{s}\to\mathsf{P}'}$ to find a good middle point for a on P', and $\mathcal{L}^{\mathsf{P}'\to\mathsf{P}\setminus\mathsf{f}}$ to find a good middle point for c on P_2 and $\mathcal{L}^{\mathsf{P}\to\mathsf{P}\setminus\mathsf{f}}$ to find a good middle point for b_1 on P_1.

Overview (see Figure 15). Let R be a shortest path from s to $b_1 = \mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s, P_1)$ in $G \setminus \{f\}$, and let Q be a separator that separates s and f. We are interested in two special vertices on R: the first vertex a^* on R that is in Q, and specifically on some P' which is an εr-subpath of Q, and the first vertex c^* on R after a^* that is on P (say, on P_2). We 'guess' estimations β, γ, δ for the lengths of $R[s, a^*]$, $R[a^*, c^*]$ and $R[c^*, b_1]$, respectively. In this overview, it is useful to consider a vertex m that is εr-$\mathsf{first}^{\beta}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s, P)$ as a sufficiently good 'middle point. Specifically, if one aims to reach an early vertex on P from s within a budget α, requiring the path to pass through m and allowing the budget to exceed α by εr does not worsen the result on P. Using the labels $\mathcal{L}^{\mathsf{s}\to\mathsf{P}'}$ with budget β, we find a sufficiently good 'middle point' a for replacing a^*. Starting from a, we use the labels $\mathcal{L}^{\mathsf{P}'\to\mathsf{P}\setminus\mathsf{f}}$ with budget γ to find a sufficiently good 'middle point' replacing c^*. Finally, we use $\mathcal{L}^{\mathsf{P}\to\mathsf{P}\setminus\mathsf{f}}$ with budget δ to find a vertex earlier on P than $\mathsf{first}^{\gamma}_{\mathsf{G}\setminus\{\mathsf{f}\}}(c^*, P_1)$.

Since each middle point is reachable with a budget increase of $O(\varepsilon r)$, the final point is reachable with a budget increase of $O(\varepsilon r)$. Since every middle point is 'reasonably good', and the budget used in each phase is an estimation of the length of the corresponding subpath of R, it is guaranteed that the final destination is at least as good as b_1.

We are now ready to provide the formal proof of Lemma 5.5.

Lemma 5.5. *There exists a labeling scheme $\mathcal{L}^{\mathsf{s}\to\mathsf{P}\setminus\{\mathsf{f}\}} = \mathcal{L}^{\mathsf{s}\to\mathsf{P}\setminus\{\mathsf{f}\}}_{G,P,r,\alpha,\varepsilon}$ where G is a planar graph equipped with a decomposition tree $\mathcal{T}$, and P is a path in $\mathcal{T}$ such that both endpoints of P lie on the same face of G and $\mathsf{len}(P) = 0$ and $r, \alpha, \varepsilon \in \mathbb{R}^+$ such that $\alpha \leq r$, and the length of every separator in $\mathcal{T}$ is bounded by r. Let s and $f \in P$ be two vertices of G that are not an ancestor apex of one another, and such that P is an ancestor path of both s and f. Given the labels of s and f, one can compute the indices on P of some vertices b_1 and b_2 that are εr-$\mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s, P_1)$ and εr-$\mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s, P_2)$, respectively, where P_1 (resp. P_2) is the prefix (resp. suffix) of P that precedes (resp. follows) f, excluding f. In this labeling scheme, the only vertices that store a label are those that have P as an ancestor path. The size of each label stored by such a vertex is $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$.*

Proof. Let $\varepsilon' = \frac{\varepsilon}{7} \in \Theta(\varepsilon)$ be an approximation parameter and let $\Gamma = \{i\varepsilon'\alpha \mid i \in [\lceil \frac{1}{\varepsilon'} \rceil]\}$. For a subpath P' of a separator in $\mathcal{T}$ below P, and for $\beta, \gamma, \delta \in \Gamma$ we define the following graphs.

1. $G^P_{P',\delta}$ is the graph G where the weights of all edges of P' are set to 0, and the label of each vertex v is set to be $\mathcal{L}^{\mathsf{P}\to\mathsf{P}\setminus\mathsf{f}}_{G,P,\delta,\varepsilon'}(v)$ obtained by Lemma 5.8.
2. $G^0_{P',\delta}$ is the graph obtained from $G^P_{P',\delta}$ by removing all outgoing edges of P (the edges of P itself are not removed).
3. $G^1_{P',\delta,\gamma}$ is the graph obtained by setting the label of each vertex v in $G^0_{P',\delta}$ to be $\mathcal{L}^{\mathsf{P}'\to\mathsf{P}\setminus\mathsf{f}}_{G^0_{P',\delta},P,P',\gamma,\varepsilon'}(v)$.
4. $G^2_{P',\delta,\gamma}$ is the induced graph of $G^1_{P',\delta,\gamma}$ only on vertices that are below the separator of P' in the fully recursive decomposition.

The Labeling. For every vertex v below the separator of P in the fully recursive decomposition the label of v stores the index of v in P, if $v \in P$. For every tuple $(P', \beta, \gamma, \delta)$ such that P' is an $\varepsilon' r$-subpath ancestor of v and $\beta, \gamma, \delta \in \Gamma$, the label $\mathcal{L}^{\mathsf{s}\to\mathsf{P}'}_{G^2_{P',\delta,\gamma},P',\beta}(v)$ and the labels of v in $G^2_{P',\delta,\gamma}$ and in $G^P_{P',\delta}$. Moreover, the label of v stores the identifiers of the ancestor pieces of v in $\mathcal{T}$. Finally, for every maximal consecutive subpath P^* of P in $G'_{P'}$, and $\gamma, \delta \in \Gamma$ the label of v stores $b^{\gamma}_{Q,P^*} = \mathsf{first}^{\gamma}_{\mathsf{G}'_{\mathsf{P}'}}(v, P^*)$ where Q is the separator from which P' originates and $G'_{P'}$ is the subgraph below Q, and the label $\mathcal{L}^{\mathsf{P}\to\mathsf{P}\setminus\mathsf{f}}_{G,P,\gamma,\varepsilon'}(b^{\gamma}_{Q,P^*})$.

Size. There are $O(\log n)$ ancestor separators of v in $\mathcal{T}$. Each separator is partitioned into $O(\frac{1}{\varepsilon})$ subpaths, so overall there are $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$ options for P' for each vertex v. It follows from $|\Gamma| = O(\frac{1}{\varepsilon})$ that there are $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$ tuples $(P', \beta, \gamma, \delta)$. For every such tuple, the label stores an $\mathcal{L}^{\mathsf{s}\to\mathsf{P}'}$-type label in $G^2_{P',\delta,\gamma}$ and the label of v in $G^2_{P',\delta,\gamma}$ and in G^P_{δ}. Due to Lemma 5.6, the size of $\mathcal{L}^{\mathsf{s}\to\mathsf{P}'}$ is

$\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$. In our construction, the size should be multiplied by another two $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$ factors since the $\mathcal{L}^{\mathsf{s}\to\mathsf{P}'}$ label is applied to the graph $G^1_{P',\delta,\gamma}$ in which every vertex is labeled with a $\mathcal{L}^{\mathsf{P}'\to\mathsf{P}\setminus\mathsf{f}}$ type label (which is of size $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$ due to Lemma 5.9), and these $\mathcal{L}^{\mathsf{P}'\to\mathsf{P}\setminus\mathsf{f}}$ labels are applied to the graph $G^P_{P',\delta}$ in which the label of each vertex is a $\mathcal{L}^{\mathsf{P}\to\mathsf{P}\setminus\mathsf{f}}$-type label (which is of size $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$ due to Lemma 5.8). Finally, the height of the recursive decomposition is logarithmic, and in every ancestor piece of v below P, the number of maximal consecutive subpaths of P is bounded by the number of apices ancestors of v which is $\tilde{O}(1)$. It follows that storing for every ancestor piece of v, for every maximal consecutive subpath P^* of P in the piece and for every $\gamma, \delta \in \Gamma$ the vertex b^γ_{Q,P^*} and an $\mathcal{L}^{\mathsf{P}\to\mathsf{P}\setminus\mathsf{f}}$-type label of b^γ_{Q,P^*} increases the size of the label by $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$.

Decoding. Given the labels of two vertices s and $f \in P$ that are not an ancestor apex of one another, and such that P is an ancestor path of both s and f, the algorithm finds a vertex b_1 that is εr-$\mathsf{first}^\alpha_{\mathsf{G}\setminus\{\mathsf{f}\}}(s, P_1)$ and a vertex b_2 that is εr-$\mathsf{first}^\alpha_{\mathsf{G}\setminus\{\mathsf{f}\}}(s, P_2)$ as follows. The algorithm starts by finding the highest separator Q in the recursive decomposition that has s in one side, and f strictly in the other side (this can be done since both s and f store all ancestor pieces). Recall that Q is partitioned into $O(\frac{1}{\varepsilon})$ subpaths of length at most $\varepsilon' r$ each. For each such subpath P', and for every $\beta, \gamma, \delta \in \Gamma$ such that $\beta + \gamma + \delta \leq (1 + 3\varepsilon')\alpha$, the algorithm uses $\mathcal{L}^{\mathsf{s}\to\mathsf{P}'}_{G^2_{P',\delta,\gamma}, P', \beta}(s)$ to find $a_{P',\beta} = \mathsf{first}^\beta_{\mathsf{G}^2_{\mathsf{P}',\delta,\gamma}}(s, P')$. More precisely, the label $\mathcal{L}^{\mathsf{s}\to\mathsf{P}'}_{G^2_{P',\delta,\gamma}, P', \beta}(s)$ allows the algorithm to retrieve $\ell_a = \mathcal{L}^{\mathsf{P}'\to\mathsf{P}\setminus\mathsf{f}}_{G^0_{P',\delta}, P, P', \gamma, \varepsilon'}(a_{P',\beta})$. Note that the label of f contains $\ell_f = \mathcal{L}^{\mathsf{P}'\xrightarrow{\mathsf{f}}\mathsf{P}}_{G^0_{P',\delta}, P, P', \gamma, \varepsilon'}(f)$. The algorithm uses the labels ℓ_a and ℓ_f to retrieve vertices $b^1_{P',\beta,\gamma}$ and $b^2_{P',\beta,\gamma}$ on P_1 (resp. on P_2) such that $b^1_{P',\beta,\gamma}$ is an $\varepsilon'\gamma$-$\mathsf{first}^\gamma_{\mathsf{G}^0_{\mathsf{P}',\delta}\setminus\{\mathsf{f}\}}(a_{P',\beta}, P_1)$ and $b^2_{P',\beta,\gamma}$ is an $\varepsilon'\gamma$-$\mathsf{first}^\gamma_{\mathsf{G}^0_{\mathsf{P}',\delta}\setminus\{\mathsf{f}\}}(a_{P',\beta}, P_2)$. Again, since the vertices of the graph $G^0_{P',\delta}$ are labeled, the algorithm actually obtains $\mathcal{L}^{\mathsf{P}\to\mathsf{P}\setminus\mathsf{f}}_{G,P,\delta,\varepsilon'}$ labels of $b^1_{P',\beta,\gamma}$ and of $b^2_{P',\beta,\gamma}$, denote these labels as ℓ^1_b and ℓ^2_b respectively. Recall that the label of f stores $\ell'_f = \mathcal{L}^{\mathsf{P}\to\mathsf{P}\setminus\mathsf{f}}_{G,P,\delta,\varepsilon'}(f)$. The algorithm uses the labels ℓ^1_b, ℓ^2_b and ℓ'_f to obtain:

- a vertex $b^{1,1}_{P',\beta,\gamma,\delta}$ that is $\varepsilon'\delta$-$\mathsf{first}^\delta_{\mathsf{G}\setminus\{\mathsf{f}\}}(b^1_{P',\beta,\gamma}, P_1)$,
- a vertex $b^{1,2}_{P',\beta,\gamma,\delta}$ that is $\varepsilon'\delta$-$\mathsf{first}^\delta_{\mathsf{G}\setminus\{\mathsf{f}\}}(b^1_{P',\beta,\gamma}, P_2)$,
- a vertex $b^{2,1}_{P',\beta,\gamma,\delta}$ that is $\varepsilon'\delta$-$\mathsf{first}^\delta_{\mathsf{G}\setminus\{\mathsf{f}\}}(b^2_{P',\beta,\gamma}, P_1)$,
- a vertex $b^{2,2}_{P',\beta,\gamma,\delta}$ that is $\varepsilon'\delta$-$\mathsf{first}^\delta_{\mathsf{G}\setminus\{\mathsf{f}\}}(b^2_{P',\beta,\gamma}, P_2)$.

Let $b^\gamma_{Q,1} = \min_{\leq P}\{b^\gamma_{Q,P^*} \mid P^*$ **is a maximal consecutive subpath of** P **in** G'_Q, $b^\gamma_{Q,P^*} \in P_1\}$ and $b^\gamma_{Q,2} = \min_{\leq P}\{b^\gamma_{Q,P^*} \mid P^*$ **is a maximal consecutive subpath of** P **in** G'_Q, $b^\gamma_{Q,P^*} \in P_2\}$. Notice that $b^\gamma_{Q,1}$ and $b^\gamma_{Q,2}$ can be computed using the index of f in P to classify each b^γ_{Q,P^*} either to P_1 or to P_2. The algorithm uses the labels $\mathcal{L}^{\mathsf{P}\to\mathsf{P}\setminus\mathsf{f}}_{G,P,\delta,\varepsilon'}(b^\gamma_{Q,1})$, $\mathcal{L}^{\mathsf{P}\to\mathsf{P}\setminus\mathsf{f}}_{G,P,\delta,\varepsilon'}(b^\gamma_{Q,1})$, and $\mathcal{L}^{\mathsf{P}\to\mathsf{P}\setminus\mathsf{f}}_{G,P,\delta,\varepsilon'}(f)$ to obtain the vertices:

1. $b^{\gamma,\delta}_{Q,1,1}$ that is an $\varepsilon'\delta$-$\mathsf{first}^\delta_{\mathsf{G}\setminus\{\mathsf{f}\}}(b^\gamma_{Q,1}, P_1)$.
2. $b^{\gamma,\delta}_{Q,1,2}$ that is an $\varepsilon'\delta$-$\mathsf{first}^\delta_{\mathsf{G}\setminus\{\mathsf{f}\}}(b^\gamma_{Q,1}, P_2)$.
3. $b^{\gamma,\delta}_{Q,2,1}$ that is an $\varepsilon'\delta$-$\mathsf{first}^\delta_{\mathsf{G}\setminus\{\mathsf{f}\}}(b^\gamma_{Q,2}, P_1)$.

4. $b^{\gamma,\delta}_{Q,2,2}$ that is an $\varepsilon'\delta$-$\mathsf{first}^{\delta}_{\mathsf{G}\setminus\{\mathsf{f}\}}(b^{\gamma}_{Q,2}, P_2)$.

The algorithm sets the vertex $b_1 = \min_{\leq P}\big(\{b^{x,1}_{P',\beta,\gamma,\delta} \mid \beta,\gamma,\delta \in \Gamma, P' \textit{ is a subpath of } Q, \beta+\gamma+\delta \leq (1+3\varepsilon')\alpha, x \in \{1,2\}\} \cup \{b^{\gamma,\delta}_{Q,x,1} \mid \gamma+\delta \leq (1+2\varepsilon')\alpha, x \in \{1,2\}\}\big)$ i.e. the first vertex on P_1 that was found across all choices of β,γ,δ and a subpath P', or some $b^{\gamma,\delta}_{Q,x,1}$. The algorithm also sets $b_2 = \min_{\leq P}\big(\{b^{x,2}_{\beta,\gamma,\delta} \mid \beta,\gamma,\delta \in \Gamma, P' \textit{ is a subpath of } Q, \beta+\gamma+\delta \leq (1+3\varepsilon')\alpha, x \in \{1,2\}\} \cup \{b^{\gamma,\delta}_{Q,x,2} \mid \gamma+\delta \leq (1+2\varepsilon')\alpha, x \in \{1,2\}\}\big)$. That is, the first vertex on P_2 that was found across all guesses of β,γ,δ and a subpath P' of Q, or $b^{\gamma,\delta}_{Q,x,2}$. The algorithm returns b_1 and b_2.

Correctness. We show that when decoding the labels of two vertices s and f, we indeed return a vertex b_1 that is $\varepsilon\mathsf{r}$-$\mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s, P_1)$. The proof that b_2 is $\varepsilon\mathsf{r}$-$\mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s, P_2)$ is similar. We start by showing that for every candidate b' the algorithm considers for being b_1, we have $\mathsf{dist}_{G\setminus\{f\}}(s, b') \leq \alpha + \varepsilon r$.

Let Q be the lowest separator in the decomposition such that s is in one side of Q and f is strictly in the other side, and let P^* be a maximal consecutive subpath of P contained in G'_Q. We start by focusing of a candidate b' of the form $b^{\gamma,\delta}_{Q,x,1}$. Recall that $\gamma+\delta \leq (1+2\varepsilon')\alpha$ and that $b^{\gamma,\delta}_{Q,x,1}$ is an $\varepsilon'\gamma$-$\mathsf{first}^{\gamma}_{\mathsf{G}\setminus\{\mathsf{f}\}}(b^{\gamma}_{Q,x}, P_1)$ vertex for $b^{\gamma}_{Q,x}$ which is the first vertex of P that can be reached on P_1 in G'_Q with budget γ. By definition, we have $\mathsf{dist}_{G'_Q}(s, b^{\gamma}_{Q,1}) \leq \gamma$, and since $f \notin G'_Q$ we have $\mathsf{dist}_{G\setminus\{f\}}(s, b^{\gamma}_{Q,x}) \leq \gamma$. We also have $\mathsf{dist}_{G\setminus\{f\}}(b^{\gamma}_{Q,x}, b^{\gamma,\delta}_{Q,x,1}) \leq (1+\varepsilon')\delta$. It follows from triangle inequality that $\mathsf{dist}_{G\setminus\{f\}}(s, b^{\gamma,\delta}_{Q,x,1}) \leq \gamma + (1+\varepsilon')\delta \leq \alpha + 3\varepsilon'\alpha \leq (1+\varepsilon)\alpha$ as required.

We now treat candidates of the form $b^{x,1}_{P',\beta,\gamma,\delta}$ for some $\beta,\gamma,\delta \in \Gamma$ and $x \in \{1,2\}$. Recall that:

1. $b^{x,1}_{P',\beta,\gamma,\delta}$ is a $\varepsilon'\delta$-$\mathsf{first}^{\delta}_{\mathsf{G}\setminus\{\mathsf{f}\}}(b^{x}_{P',\beta,\gamma}, P_1)$.

2. $b^{x}_{P',\beta,\gamma}$ is a $\varepsilon'\gamma$-$\mathsf{first}^{\gamma}_{\mathsf{G}^0_{\mathsf{P}',\delta}\setminus\{\mathsf{f}\}}(a_{P',\beta}, P_x)$.

3. $a_{P',\beta}$ is $\mathsf{first}^{\beta}_{\mathsf{G}^2_{\mathsf{P}',\delta,\gamma}}(s, P')$

Recall that $f \notin G^2_{P',\delta,\gamma}$ since $G^2_{P',\delta,\gamma}$ only contains vertices in the side of s of the separator of Q in the full recursive decomposition. Also notice that in $G^2_{P',\delta,\gamma}$ the length of the path P' is 0, and in G the length of P' is at most $\varepsilon' r$. It follows that $\mathsf{dist}_{G\setminus\{f\}}(s, a_{P',\beta}) \leq \mathsf{dist}_{G^2_{P',\delta,\gamma}}(s, a_{P',\beta}) + \varepsilon' r \leq \beta + \varepsilon' r$. Due to the same reasoning, we also have $\mathsf{dist}_{G\setminus\{f\}}(a_{P',\beta}, b^{x}_{P',\beta,\gamma}) \leq \mathsf{dist}_{G^0_{P',\delta}\setminus\{f\}}(a_{P,\beta}, b^{x}_{P',\beta,\gamma}) + \varepsilon' r \leq \gamma + \varepsilon'\gamma + \varepsilon' r$. Directly from the definition of $\varepsilon'\delta$-$\mathsf{first}^{\delta}_{\mathsf{G}\setminus\{\mathsf{f}\}}(b^{x}_{P',\beta,\gamma}, P_1)$ we have $\mathsf{dist}_{G\setminus\{f\}}(b^{x}_{P',\beta,\gamma}, b^{x,1}_{P',\beta,\gamma,\delta}) \leq \delta + \varepsilon'\delta$.

From triangle inequality and $\beta+\gamma+\delta \leq (1+3\varepsilon')\alpha$ we get $\mathsf{dist}_{G\setminus\{f\}}(s, b^{x,1}_{P',\beta,\gamma,\delta}) \leq (\beta+\varepsilon' r) + (\gamma+\varepsilon'\gamma+\varepsilon' r) + (\delta+\varepsilon'\delta) \leq (\beta+\gamma+\delta) + \varepsilon'(2r+\gamma+\delta) \leq \alpha + \varepsilon'(2r+3\alpha+\gamma+\delta) \leq \alpha + 7\varepsilon' r = \alpha + \varepsilon r$ as required.

We are now left with the task of proving $b_1 \leq_P \mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s, P_1)$. Notice that we have $b_1 \in P_1$ since b_1 can either be $b_{Q,1}$ which is by definition on P_1, or $b^{x,1}_{\beta,\gamma,\delta}$ which is a P_1 output of a $\mathcal{L}^{\mathsf{P}\to\mathsf{P}\setminus\mathsf{f}}$-type label.

Let $b^* = \mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(s, P_1)$ and let R be a shortest path from s to b^* in $G \setminus \{f\}$. Let a^* be the first vertex on R that is also in Q, and let c^* be the first vertex on $R[a^*..b^*]$ that is on P. Notice that a^* (and therefore, c^*) do not necessarily exist, but if a^* exists, so does c^*.

Case 1: a^* does not exist Let b' be the first vertex on R that is on P. Let $\gamma^* = \min\{x \in \Gamma \mid x \geq \mathsf{len}(R[s,b']\}$ and $\delta^* = \min\{x \in \Gamma \mid x \geq \mathsf{len}(R[b',b^*]\}$ and $b' \in P_x$ for $x \in \{1,2\}$. Notice that b' must be on one of the maximal consecutive subpaths of P that are contained in G'_Q, we denote this maximal subpath as P^*. Since $R[s,b']$ is a path with length at most γ in G'_Q from s to b', we have $b^{\gamma^*}_{Q,P^*} = \mathsf{first}^{\gamma}_{\mathsf{G'_Q}}(s,P^*) \leq_P b'$. In particular, $b^{\gamma^*}_{Q,x} \leq_P b'$ as an $\leq_P$ minimum in a set containing P^*. Notice that $\gamma^* + \delta^* \leq (1+2\varepsilon')\alpha$, so the algorithm considered a candidate $b^{\gamma^*,\delta^*}_{Q,x,1}$ that is an $\varepsilon'\delta^*$-$\mathsf{first}^{\delta^*}_{\mathsf{G}\setminus\{\mathsf{f}\}}(b^{\gamma^*}_{Q,x},P_1)$. Since $P_1[b^{\gamma^*}_{Q,x},b']R[b',b^*]$ is a path in $G \setminus \{f\}$ from $b^{\gamma^*}_{Q,x}$ to b^* of length at most δ^*, we have that $b^{\gamma^*,\delta^*}_{Q,x,1} \leq_P \mathsf{first}^{\gamma^*}_{\mathsf{G}\setminus\{\mathsf{f}\}}(b^{\gamma^*}_{Q,x},P_1) \leq_P b^*$.

This concludes the proof of this case, as $b_1 \leq_P b^{\gamma^*,\delta^*}_{Q,x,1}$ as the $\leq_P$ minimum of a set containing the latter.

Case 2: a^* exists, and $c^* \in P_1$ Let P' be the subpath of Q that contains a^*. Let:

1. $\beta^* = \min\{x \in \Gamma \mid x \geq \mathsf{len}(R[s,a^*])\}$,
2. $\gamma^* = \min\{x \in \Gamma \mid x \geq \mathsf{len}(R[a^*,c^*])\}$, and
3. $\delta^* = \min\{x \in \Gamma \mid x \geq \mathsf{len}(R[c^*,b^*])\}$.

Notice that $\beta^*+\gamma^*+\delta^* \leq \mathsf{len}(R)+3\varepsilon'\alpha$. Therefore, the decoding algorithm computed some $b^{1,1}_{\beta^*,\gamma^*,\delta^*}$ such that:

1. $b^{1,1}_{\beta^*,\gamma^*,\delta^*}$ is a $\varepsilon'\delta^*$-$\mathsf{first}^{\delta^*}_{\mathsf{G}\setminus\{\mathsf{f}\}}(b^1_{\beta^*,\gamma^*},P_1)$.
2. $b^1_{\beta^*,\gamma^*}$ is a $\varepsilon'\gamma^*$-$\mathsf{first}^{\gamma^*}_{\mathsf{G}^0_{\mathsf{P'},\delta^*}\setminus\{\mathsf{f}\}}(a_{\beta^*},P_1)$.
3. a_{β^*} is $\mathsf{first}^{\beta^*}_{\mathsf{G}^2_{\mathsf{P'},\delta^*,\gamma^*}}(s,P')$.

Notice that we omit P' from the subscript of the vertices listed above (i.e. $b^{1,1}_{P',\beta^*,\gamma^*,\delta^*}$ is written as $b^{1,1}_{\beta^*,\gamma^*,\delta^*}$). This is done for convenience, as this notation will not be used in the current context with a subpath other than P'.

Recall that $G^2_{P',\gamma^*,\delta^*}$ is a vertex labeled version of G'_Q, with the weights of the edges of P' set to 0. Since $R[s,a^*]$ is a path from s to P' in $G'_Q \setminus \{f\} = G'_Q$ with length at most β^*, it is in particular a path from s to $a^* \in P'$ in $G^2_{P',\gamma^*,\delta^*}$ with length at most β^* and therefore we have $a_{\beta^*} \leq_{P'} a^*$.

Recall that G^0_{P',δ^*} is a vertex labeled version of G with the weights of all edges of P' set to 0 without outgoing edges from P. Since $R[a^*,c^*]$ is a path from a^* to c^* in $G \setminus \{f\}$ with length at most γ^* that do not use outgoing edges of P (as it is internally disjoint from P), we have that $P'[a_{\beta^*},a^*] \cdot R[a^*,c^*]$ is a path from a_{β^*} to $c^* \in P_1$ in $G^0_{P',\delta^*} \setminus \{f\}$ with length at most γ^* (also recall that $f \notin P'$). We therefore have $b^1_{\beta^*,\gamma^*} \leq_P c^*$.

Furthermore, consider the path $P_1[b^1_{\beta^*,\gamma^*},c^*] \cdot R[c^*,b^*]$. It is a path with length at most δ^* (recall that $\mathsf{len}(P)=0$) in $G \setminus \{f\}$. Therefore, we have $b^{1,1}_{\beta^*,\gamma^*,\delta^*} \leq_P b^*$. Due to the minimality of the returned vertex b_1 on P across all values of β,γ,δ and all subpaths P', we have that $b_1 \leq_P b^{1,1}_{\beta^*,\gamma^*,\delta^*} \leq_P b^*$ as required.

Case 3: a^* exists and $c^* \in P_2$ This proof for this case is completely identical to the proof of the previous case, replacing $b^{1,1}_{\beta^*,\gamma^*,\delta^*}$ with $b^{2,1}_{\beta^*,\gamma^*,\delta^*}$. □

6 The $\mathcal{L}^{\mathsf{P}\to\mathsf{P}\setminus\mathsf{f}}$ Labeling (Proof of Lemma 5.8)

In this section we prove Lemma 5.8 by extending the ideas of Section 4.4 from reachability to approximate distances. We recall the settings of Lemma 5.8. G is a planar graph, P is a 0-length path of G whose two endpoints lie on the same face and $\alpha, \varepsilon \in \mathbb{R}^+$. Our goal is to assign $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$-sized labels to the vertices of P such that, given the labels of two vertices b and f of P, one can compute the indices on P of some vertices b_1 and b_2 that are $\varepsilon\alpha$-$\mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(b, P_1)$ and $\varepsilon\alpha$-$\mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(b, P_2)$, respectively. Throughout, P_1 and P_2 denote the prefix and suffix of P before and after vertex f (without f) respectively.

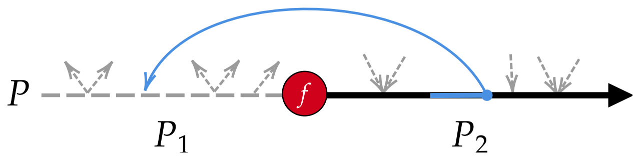


Figure 16: An illustration of G_1. We are interested in P_2 to P_1 paths (like the blue path) that may start with some edges of P_2 and then continue with a subpath which is internally disjoint from P. Formally, this is achieved by removing all in-going edges to P_2 and all out-going edges from P_1 (the removed edges are displayed in gray in the figure).

We define G_1 (resp. G_2) to be the graph obtained from $G \setminus \{f\}$ by first removing all in-going edges to vertices of P_2 (resp. P_1), except for the edges of P_2 (resp. P_1) itself, and then removing all out-going edges from vertices of P_1 (resp. P_2), including the edges of P_1 (resp. P_2), see Figure 16. Intuitively, G_1 (resp. G_2) is a version of $G \setminus \{f\}$ in which every path from P_2 (resp. P_1) to P_1 (resp. P_2) starts with some subpath of P_2 (resp. P_1) and is then internally disjoint from P.

We divide our task into four auxiliary labeling schemes.

1. $\mathcal{L}^{\mathsf{P_2}\to\mathsf{P_1}\to\mathsf{P}}_{\beta,\gamma}$ - Given the labels of two vertices $f <_P b$, one can obtain $\mathsf{first}^{\gamma}_{\mathsf{G}\setminus\{\mathsf{f}\}}(x, P_1)$ and $\mathsf{first}^{\gamma}_{\mathsf{G}\setminus\{\mathsf{f}\}}(x, P_2)$ for a vertex x that is $\varepsilon\beta$-$\mathsf{first}^{\beta}_{\mathsf{G_1}}(b, P_1)$. We introduce a labeling scheme for this problem in Section 6.1.

2. $\mathcal{L}^{\mathsf{P_1}\to\mathsf{P_2}\to\mathsf{P}}_{\beta,\gamma}$ - Given the labels of two vertices $b <_P f$, one can obtain $\mathsf{first}^{\gamma}_{\mathsf{G}\setminus\{\mathsf{f}\}}(x, P_1)$ and $\mathsf{first}^{\gamma}_{\mathsf{G}\setminus\{\mathsf{f}\}}(x, P_2)$ for a vertex x that is $\varepsilon\beta$-$\mathsf{first}^{\beta}_{\mathsf{G_2}}(b, P_2)$. The labeling scheme of $\mathcal{L}^{\mathsf{P_2}\to\mathsf{P_1}\to\mathsf{P}}_{\beta,\gamma}$ (Section 6.1) can be adjusted to solve this problem.

3. $\mathcal{L}^{\mathsf{P_2}\to\mathsf{P_2}}_{\beta}$ - Given the labels of two vertices $f <_P b$ one can compute $\varepsilon\beta$-$\mathsf{first}^{\beta}_{\mathsf{G}\setminus(\mathsf{P_1}\cup\{\mathsf{f}\})}(b, P_2)$. We introduce a labeling scheme for this problem in Section 6.2.

4. $\mathcal{L}^{\mathsf{P_1}\to\mathsf{P_1}}_{\beta}$- Given the labels of two vertices $b <_P f$ one can compute $\varepsilon\beta$-$\mathsf{first}^{\beta}_{\mathsf{G}\setminus(\mathsf{P_2}\cup\{\mathsf{f}\})}(b, P_1)$. The labeling scheme of $\mathcal{L}^{\mathsf{P_2}\to\mathsf{P_2}}_{\beta}$ (Section 6.2) can be adjusted to solve this problem.

Exploiting the labeling schemes for the auxiliary tasks, we prove the following lemma (see Figure 17).

Lemma 5.8. *There exists a labeling scheme* $\mathcal{L}^{\mathsf{P}\to\mathsf{P}\setminus\mathsf{f}} = \mathcal{L}^{\mathsf{P}\to\mathsf{P}\setminus\mathsf{f}}_{G,P,\alpha,\varepsilon}$ *where G is a planar graph, P is a 0-length path of G whose two endpoints lie on the same face and $\alpha, \varepsilon \in \mathbb{R}^+$. Given the labels of two vertices b and f on P let P_1 and P_2 be the prefix and suffix of P before and after f (without f),*

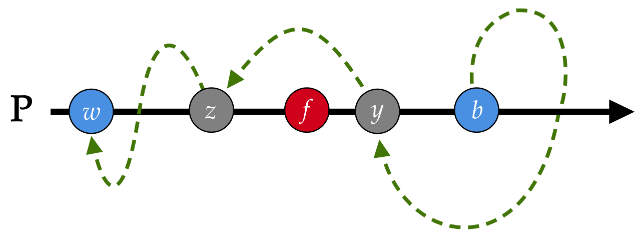


Figure 17: An illustration of a path R from b to $w = \mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(b, P_1)$. The first subpath is from b to $y \in P_2$ (without using P_1), the second subpath is from y to $z \in P_1$ using a path that is internally disjoint from P except for a prefix. Finally, the third subpath from z to $w \in P_1$ is a path in $G \setminus \{f\}$. The first part is approximated using $\mathcal{L}^{\mathsf{P_2 \to P_2}}$, the second and third parts are approximated using $\mathcal{L}^{\mathsf{P_2 \to P_1 \to P}}$.

respectively. One can compute the indices on P of some vertices b_1 and b_2 that are $\varepsilon\alpha$-$\mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(b, P_1)$ and $\varepsilon\alpha$-$\mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(b, P_2)$, respectively. The size of each label is $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$.

Proof. Let $\varepsilon' = \varepsilon/6$ and let $\Gamma = \{i\varepsilon'\alpha \mid i \in [0, \lceil 1/\varepsilon' \rceil]\}$ be the set of multiples of $\varepsilon'\alpha$. For every $(\gamma, \delta) \in \Gamma^2$ we define the labeled graph $G_{\gamma,\delta}$ as a copy of G in which every vertex $v \in V$ is labeled with $\mathcal{L}^{\mathsf{P_2 \to P_1 \to P}}_{\gamma,\delta}(v)$ and $\mathcal{L}^{\mathsf{P_1 \to P_2 \to P}}_{\gamma,\delta}(v)$, where the labeles are computed with approximation factor ε'. For every $v \in V$, the label $\mathcal{L}^{\mathsf{P \to P \setminus f}}(v)$ stores for every $(\beta, \gamma, \delta) \in \Gamma^3$ the labels of $\mathcal{L}^{\mathsf{P_2 \to P_2}}_{\beta}(v)$ and $\mathcal{L}^{\mathsf{P_1 \to P_1}}_{\beta}(v)$ computed with approximation factor ε' on the graph $G_{\gamma,\delta}$, and also the label of v in $G_{\gamma,\delta}$.

Size. By Lemmas 6.6 and 6.14 we have that the size of each label of $\mathcal{L}^{\mathsf{P_2 \to P_1 \to P}}$, $\mathcal{L}^{\mathsf{P_1 \to P_2 \to P}}$, $\mathcal{L}^{\mathsf{P_2 \to P_2}}$ and $\mathcal{L}^{\mathsf{P_1 \to P_1}}$ is $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$. Thus, the size of $\mathcal{L}^{\mathsf{P \to P \setminus f}}(v)$ is $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$. Note that $\mathcal{L}^{\mathsf{P_2 \to P_2}}$ and $\mathcal{L}^{\mathsf{P_1 \to P_1}}$ labels on the graph $G_{\beta,\delta}$, in which vertices have labels of size $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$, still have size $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$.

Decoding. Given the labels of b and f, if $f <_P b$ the computation of the indices on P of some vertices b_1 and b_2 that are $\varepsilon\alpha$-$\mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(b, P_1)$ and $\varepsilon\alpha$-$\mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(b, P_2)$, respectively, is done as follows. Iterate over all triplets $(\beta, \gamma, \delta) \in \Gamma^3$ such that $\beta + \gamma + \delta \leq (1 + 3\varepsilon')\alpha$. The algorithm uses $\mathcal{L}^{\mathsf{P_2 \to P_2}}_{\beta}(b)$ and $\mathcal{L}^{\mathsf{P_2 \to P_2}}_{\beta}(f)$ to compute some vertex $b'_{\beta,\gamma,\delta} \in G_{\gamma,\delta}$ which is $\varepsilon'\beta$-$\mathsf{first}^{\beta}_{\mathsf{G}\setminus(\mathsf{P_1}\cup\{\mathsf{f}\})}(b, P_2)$. Then, using $\mathcal{L}^{\mathsf{P_2 \to P_1 \to P}}_{\gamma,\delta}(f)$ and $\mathcal{L}^{\mathsf{P_2 \to P_1 \to P}}_{\gamma,\delta}(b'_{\beta,\gamma,\delta})$ we compute $b^1_{\beta,\gamma,\delta} = \mathsf{first}^{\delta}_{\mathsf{G}\setminus\{\mathsf{f}\}}(x_{\beta,\gamma,\delta}, P_1)$ and $b^2_{\beta,\gamma,\delta} = \mathsf{first}^{\delta}_{\mathsf{G}\setminus\{\mathsf{f}\}}(x_{\beta,\gamma,\delta}, P_2)$ for a vertex $x_{\beta,\gamma,\delta}$ that is $\varepsilon'\gamma$-$\mathsf{first}^{\gamma}_{\mathsf{G_1}}(b'_{\beta,\gamma,\delta}, P_1)$. Finally, b_1 and b_2 are the earliest vertices of P among all $b^1_{\beta,\gamma,\delta}$ and $b^2_{\beta,\gamma,\delta}$, respectively. The case where $b <_P f$ is decoded in a similar way, using $\mathcal{L}^{\mathsf{P_1 \to P_1}}$ and $\mathcal{L}^{\mathsf{P_1 \to P_2 \to P}}$.

Correctness. First notice that for every (β, γ, δ) we have $\mathsf{dist}_{G\setminus\{f\}}(b, b^1_{\beta,\gamma,\delta}) \leq \mathsf{dist}_{G\setminus\{f\}}(b, b'_{\beta,\gamma,\delta}) + \mathsf{dist}_{G\setminus\{f\}}(b'_{\beta,\gamma,\delta}, x_{\beta,\gamma,\delta}) + \mathsf{dist}_{G\setminus\{f\}}(x_{\beta,\gamma,\delta}, b^1_{\beta,\gamma,\delta})$. By definition of $\mathcal{L}^{\mathsf{P_2 \to P_2}}$ we have $\mathsf{dist}_{G\setminus\{f\}}(b, b'_{\beta,\gamma,\delta}) \leq (1+\varepsilon')\beta$ and by definition of $\mathcal{L}^{\mathsf{P_2 \to P_1 \to P}}$ we have $\mathsf{dist}_{G\setminus\{f\}}(b'_{\beta,\gamma,\delta}, x_{\beta,\gamma,\delta}) \leq (1+\varepsilon')\gamma$ and $\mathsf{dist}_{G\setminus\{f\}}(x_{\beta,\gamma,\delta}, b^1_{\beta,\gamma,\delta}) \leq (1+\varepsilon')\delta$. Thus, $\mathsf{dist}_{G\setminus\{f\}}(b, b^1_{\beta,\gamma,\delta}) \leq (1+3\varepsilon')\alpha + 3\varepsilon'\alpha = (1+6\varepsilon')\alpha = (1+\varepsilon)\alpha$. By similar arguments, $\mathsf{dist}_{G\setminus\{f\}}(b, b^2_{\beta,\gamma,\delta}) \leq (1+6\varepsilon')\alpha = (1+\varepsilon)\alpha$, as required.

It remains to prove that $b_1 \leq_P \mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(b, P_1)$ and $b_2 \leq_P \mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(b, P_2)$. We prove the former, the proof of the latter is similar. Let R be a shortest path from b to $w = \mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(b, P_1)$ in $G \setminus \{f\}$ (see Figure 17. Let z be the first vertex on R that is on P_1 and let y be the last vertex on $R[b, z]$ which is on P_2. Let $\beta = \min\{q \in \Gamma \mid q \geq \mathsf{len}(R[b, y])\}$, let $\gamma = \min\{q \in \Gamma \mid q \geq \mathsf{len}(R[y, z])\}$ and let

$\delta = \min\{q \in \Gamma \mid q \geq \mathsf{len}(R[z,w])\}$. Notice that $\beta+\gamma+\delta \leq \alpha + 3\varepsilon'\alpha$. We will show that $b^1_{\beta,\gamma,\delta} \leq_P w$, the claim then follows from the minimality (w.r.t $\leq_P$) of b_1 on P across all choices of β, γ, δ.

First, $b'_{\beta,\gamma,\delta} \leq_P \mathsf{first}^{\beta}_{\mathsf{G}\setminus(\mathsf{P_1}\cup\{\mathsf{f}\})}(b, P_2) \leq_P y$ since y can be reached in $G \setminus (P_1 \cup \{f\})$ from b with a path ($R[b,y]$) of length at most β. Second, $x_{\beta,\gamma,\delta} \leq_P \mathsf{first}^{\gamma}_{\mathsf{G_1}}(b'_{\beta,\gamma,\delta}, P_1) \leq_P z$ since z can be reached in G_1 from $b'_{\beta,\gamma,\delta}$ with a path ($P[b'_{\beta,\gamma,\delta}, y] \cdot R[y,z]$) of length at most γ. Finally, $b^1_{\beta,\gamma,\delta} \leq_P \mathsf{first}^{\delta}_{\mathsf{G}\setminus\{\mathsf{f}\}}(x_{\beta,\gamma,\delta}, P_1) \leq_P w$ since w can be reached in $G \setminus \{f\}$ from $x_{\beta,\gamma,\delta}$ with a path ($P[x_{\beta,\gamma,\delta}, z] \cdot R[z,w]$) of length at most δ. ☐

6.1 The $\mathcal{L}^{\mathsf{P_2}\to\mathsf{P_1}\to\mathsf{P}}$ labeling

In this section we focus on a vertex $f \in P$. We denote $G' = G^1$. For a vertex $b \in P_2$ let D_b be a shortest path from b to $\mathsf{first}^{\beta}_{\mathsf{G'}}(b, P_1)$. If $\mathsf{first}^{\beta}_{\mathsf{G'}}(b, P_1) = null$ we say that b *cannot bypass* f. This section is mostly dedicated to proving the following lemma, which leads to a simple labeling scheme for $\mathcal{L}^{\mathsf{P_2}\to\mathsf{P_1}\to\mathsf{P}}$.

Lemma 6.1. *There are sequences $I = (b_1 <_{P_2} b_2 <_{P_2} \ldots <_{P_2} b_t) \subseteq P_2$ and $V = (v_1, v_2, \ldots, v_t) \subseteq P_1$ such that:*

1. *For every vertex $b \in P_2$ we have $\min_{\leq_{P_1}}\{v_i \mid b_i \geq_{P_2} b\}$ is an $\varepsilon\beta$-$\mathsf{first}^{\beta}_{\mathsf{G'}}(b, P_1)$.*
2. *Every vertex $b >_{P_2} b_t$ cannot bypass f.*
3. *$t \in O(1/\varepsilon)$.*

For $v \in P_2$ we denote $v_{first} = \mathsf{first}^{\beta}_{\mathsf{G'}}(v, P_1)$. Let B_f be the set of all pairs (v, v_{first}), that are maximal with respect to inclusion (i.e. B_f does not include a pair (v, v_{first}) if there exists (u, u_{first}) with $v <_P u$ and $u_{first} \leq_P v_{first}$). By the following claim, we can assume that for every $b \in P_2$, the path D_b starts with a prefix $P[b..u]$ and procedes with D_u such that $(u, u_{first}) \in B_f$.

Claim 6.2. *Let $b >_P f$ be a vertex that can bypass f. There is a path from b to b_{first} beginning with a subpath of P followed by some D_u such that $(u, u_{first}) \in B_f$.*

Proof. Since G' does not contain in-going edges into vertices of P_2 (except for the edges of P_2) and does not contain out-going edges from vertices of P_1 it must be that $D_b = P[b,u] \cdot D_b[u, b_{first}]$ for some vertex $u \geq_P b$ such that $D_b[u, b_{first}]$ is internally disjoint from P. Notice that $\mathsf{len}(D_b[u, b_{first}]) = \mathsf{len}(D_b) \leq \beta$ since $\mathsf{len}(P) = 0$, therefore $u_{first} \leq_P b_{first}$. Moreover, $\mathsf{len}(P[b,u] \cdot D_u) = \mathsf{len}(D_u) \leq \beta$ and therefore, it must be that $u_{first} \geq_P b_{first}$, thus $b_{first} = u_{first}$.

Now, let us fix D_b as a b-to-b_{first} path of length at most β that maximizes u (with respect to the $\leq_P$ order). (Notice that D_b is not necessarily a shortest path.) Assume by contradiction that $(u, u_{first}) \notin B_f$, so there is a vertex $v \in P_2$ such that $(v, v_{first}) \in B_f$ and $v_{first} \leq_P u_{first}$ and $u <_P v$. However, in this case $\mathsf{len}(P[b,v] \cdot D_v) \leq \mathsf{len}(D_v) \leq \beta$ which implies $v_{first} \geq_P b_{first} = u_{first}$, and therefore $v_{first} = b_{first}$. The path $P[b,v] \cdot D_v$ contradicts the maximality of u. ☐

Since the endpoints of P lie on the same face, a path D_b that emanates P to the right (resp. left) of P must enter P from the right (resp. left) of P. Let B^{left}_f and B^{right}_f be the set of left and right pairs in B_f, respectively. Let G'_{right} (G'_{left}) be the induced graph from P, restricted only to vertices v such that there is a path from P, that is internally disjoint from P, to v that emanates P to the right (resp. left).

We prove Lemma 6.1 for the graph G'_{right} (instead of G'). Applying the same argument for G'_{left} would produce a second pair of sequences I, V. It is straightforward to merge the two I's sequences and the two V's sequences obtained for both cases to complete the proof of Lemma 6.1, since any P_2 to P_1 path is contained in either G'_{right} or G'_{left}.

An algorithm for $\varepsilon = 1$. We start with a warm-up solution for $\varepsilon = 1$ (i.e., a 2-approximation). In this case, we show sequences $I = (b_1)$ and $V = (v_1)$. We define v_1 and b_1 as follows. v_1 is the first vertex of P_1 such that there exists a $(b_{v_1}, v_1) \in B_f^{right}$. b_1 is the last vertex of P_2 such that there is a pair $(b_1, (b_1)_{first}) \in B_f^{right}$. We show that the sequences I and V satisfy the lemma for $\varepsilon = 1$. Notice that the demand $t \in O(1/\varepsilon)$ is trivial in the current context as ε and t are both constants. Additionally, notice that from the definition of b_1, every vertex $b >_{P_2} b_1$ cannot bypass f.

Claim 6.3. *If $f <_P b \leq_P b_1$ then v_1 is a β-$\mathsf{first}^{\beta}_{\mathsf{G'}}(b, P_1)$.*

Proof. Notice that $b_{first} >_{P_1} v_1$ by Claim 6.2 and the definition of v_1. We will show that there is a path of length at most 2β from b_1 to v_1, which concludes the claim as every b can reach v via b_1 with a path of length 2β, making v a β-$\mathsf{first}^{\beta}_{\mathsf{G'}}(b, P_1)$ vertex.

Consider the paths D_{b_1} and $D_{b_{v_1}}$. Since both paths emanate P to the right, and since $v_1 \leq_P (b_1)_{first} <_P b_{v_1} \leq_P b_1$ (due to both pairs being in B_f), it must be the case that D_{b_1} intersects with $D_{b_{v_1}}$, say at vertex z. It follows that the path $D_{b_1}[b_1, z] \cdot D_{b_v}[z, v_1]$ is a path of length at most 2β from b_1 to v_1, as required. □

An algorithm for any ε. We now extend the above solution from $\varepsilon = 1$ to any $\varepsilon > 0$. We will partition B_f^{right} into $O(1/\varepsilon)$ subsets. For ease of presentation let us denote $B = B_f^{right}$.

Let $A_1 = (b_1, p_1)$ and $A_2 = (b_2, p_2)$ with $b_1 >_{P_2} b_2$ be two different pairs in B. We first note that due to the maximality with respect to inclusion $p_2 <_P p_1 <_P f <_P b_2 <_P b_1$. Thus, we say that A_1 and A_2 *cross* each other, and denote it by $A_1 \not\rightarrow A_2$. If there is a path S from b_1 to p_2 that is internally disjoint from P and is to the right of P (by definition of G'_{right}) of length at most $(1+\varepsilon)r$ then we say that A_1 and A_2 *good-cross* each other, and denote it as $A_1 \overset{\text{good}}{\not\rightarrow} A_2$. Otherwise, we say that they bad-cross each other and denote it by $A_1 \overset{\text{bad}}{\not\rightarrow} A_2$.

Lemma 6.4. *B can be partitioned into $O(1/\varepsilon)$ subsets $B_1, B_2, \ldots, B_t$ such that every two pairs in the same B_i good-cross each other.*

Proof. We present an algorithm that partitions B. The algorithm runs in phases, creating the subset B_i in the i'th phase. At the beginning of every phase, we initialize a set $B_{good} = \emptyset$. During a phase, we examine the pairs of B in decreasing $<_P$ order of their first vertex. When examining a pair A, we check if A good-crosses all pairs in B_{good}, and if so we add A into B_{good} and remove A from B. At the end of the i'th phase, we set $B_i = B_{good}$ and append B_i to the partition and if $B = \emptyset$ the algorithm terminates. Clearly $|B_{good}| \geq 1$, so the algorithm halts with a partition of the initial B. It should also be clear from the construction of B_{good} at every iteration that two pairs in the same B_i good-cross each other.

To conclude the proof, we show that the number of phases $t \leq 1/\varepsilon + 1$. By definition, for every $i \in [2..t]$ there is a pair $A_i = (b_i, p_i)$ in B_i and a pair $A_{i-1} = (b_{i-1}, p_{i-1})$ in B_{i-1} with $b_{i-1} >_{P_2} b_i$ such that $A_{i-1} \overset{\text{bad}}{\not\rightarrow} A_i$. Let A_t be some pair in B_t and $A_{t-1}, A_{t-2}, \ldots, A_1$ be a sequence of pairs with $A_{i-1} \in B_{i-1}$ being an arbitrary pair satisfying the above with respect to $A_i \in B_i$, starting with A_t.

We claim that for every $i \in [1..t]$ there is a path from b_i to p_1 in G'_{right} of length at most $(1 - (i-1)\varepsilon)\beta$ (see Figure 18). The claim is trivial for $i = 1$. We thus assume the claim holds for i, and prove it for $i+1$. Let P_1 be the path from b_i to p_1 and let $P_2 = D_{b_{i+1}}$ be the path corresponding to the pair (b_{i+1}, p_{i+1}). Since $p_{i+1} \leq_P p_1 <_P b_{i+1} <_P b_i$, and since paths only emanate to the right of P in G'_{right}, we must have that P_1 and P_2 intersect at some vertex z. Since $A_i \overset{\text{bad}}{\not\rightarrow} A_{i+1}$, any path from b_i to p_{i+1} in G'_{right} is of length more than $(1+\varepsilon)\beta$. In particular the path $S = P_1[b_i, z] \cdot P_2[z, p_{i+1}]$ is of length more than $(1+\varepsilon)\beta$. Let $S' = P_2[b_{i+1}, z] \cdot P_1[z, p_1]$. We show that $\mathsf{len}(S') \leq (1 - i\varepsilon)\beta$. By the induction hypothesis, $\mathsf{len}(P_1) \leq (1 - (i-1)\varepsilon)\beta$. By the definition

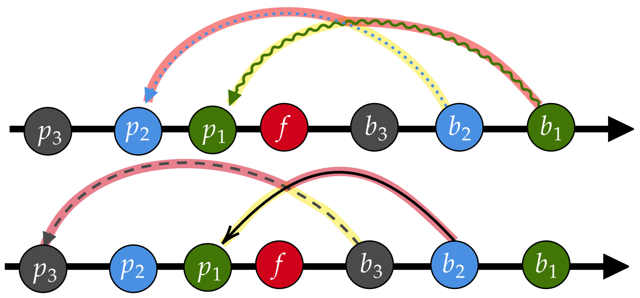


Figure 18: An illustration for the proof of Lemma 6.4. In the top figure, the green wavy path corresponds to A_1 and the blue dotted path corresponds to A_2. The sum of the green and blue paths is bounded by 2β. Since $A_1 \overset{\text{bad}}{\not\to} A_2$, the highlighted red path from b_1 to p_2 is expensive: its length is at least $(1+\varepsilon)\beta$. Since the yellow and red paths sum up to $\mathsf{len}(A_1) + \mathsf{len}(A_2)$, it follows that the highlighted yellow path from b_2 to p_1 is cheap: its length is at most $(1-\varepsilon)\beta$.
The bottom figure demonstrates the next inductive step. Now, the cheap b_2-to-p_1 path from the previous step is shown in solid black. The dashed path from b_3 to p_3 represents A_3. As in the top picture, we decompose A_3 and the cheap path into a highlighted red b_2-to-p_3 path and a highlighted yellow b_3-to-p_1 path. It follows from $A_1 \overset{\text{bad}}{\not\to} A_2$ that the red path is of length more than $(1+\varepsilon)\beta$. Since we started with a cheap b_2-to-p_1 path, the sum of red and yellow is now $2\beta - \varepsilon\beta$, which leads to the length of yellow being at most $\beta - 2\varepsilon\beta$.

of B we have $\mathsf{len}(P_2) \le \beta$. Notice that $\mathsf{len}(P_1) + \mathsf{len}(P_2) = \mathsf{len}(S) + \mathsf{len}(S')$. Since $\mathsf{len}(S) > (1+\varepsilon)\beta$ we have $\mathsf{len}(S') \le \beta + (1-(i-1)\varepsilon)\beta - (1+\varepsilon)\beta = (1-i\varepsilon)\beta$, as required.

To conclude, if $t > 1/\varepsilon + 1$ then there is a path from b_t to p_1 of negative length, a contradiction. $\square$

We proceed to define the sequences I and V. Let $B_1, B_2, \dots B_t$ be the partition obtained by Lemma 6.4. For every $i \in [t]$, let p'_i be the first vertex of P such that there exists a pair $(\cdot, p'_i) \in B_i$. Let b'_i be the last vertex of P such that there is a $(b'_i, \cdot) \in B_i$. Assume that the B_i's are indexed such that $b'_1 <_P b'_2 <_P \dots <_P b'_t$ [9]. We define $I = (f, b'_1, b'_2 \dots, b'_t)$ and $V = (p'_1, p'_2, \dots p'_t)$.

It follows immediately from Lemma 6.4 that the lengths of the sequences are $O(\frac{1}{\varepsilon})$. Additionally, note that each b'_i is a last vertex on P in its respective B_i, and b'_t is the last vertex on P among all b'_i's. Since the sets B_i's form a partition of B, it holds that b'_t is the last vertex of P such that there is a pair $(b'_t, \cdot)$ in P. From the definition of B, it means that every $b >_{P_2} b'_t$ cannot bypass f. We prove via the following claim that I and V satisfy the remaining property of Lemma 6.1.

Claim 6.5. *For every vertex $b \in P_2$ we have that $\min_{\le_{P_1}} \{v'_i \mid b'_i \ge_{P_2} b\}$ is a $\varepsilon\beta$-$\mathsf{first}^{\beta}_{\mathsf{G}'}(b, P_1)$*

Proof. First, we claim that for every $b'_i \ge_{P_2} b$, we have $\mathsf{dist}_{G'}(b, v'_i) \le (1+\varepsilon)\beta$. Recall that b'_i and v'_i are obtained as components of two pairs $(b'_i, \cdot)$ and $(\cdot, v'_i)$ in B_i. Since both pairs are in B_i, they good-cross each other. Therefore, $\mathsf{dist}_{G'}(b'_i, v'_i) \le (1+\varepsilon)\beta$ via some shortest path D_i. We thus have that $P[b, b'_i] \cdot D_i$ is a path from b to v'_i of length at most $(1+\varepsilon)\beta$.

By Claim 6.2 there is a pair $(u, b_{first}) \in B$ such that the path $P[b, u] \cdot D_u$ is of length at most β. Let B_j be the set that contains the pair (u, b_{first}). Recall that $(b_j, \cdot) \in B_j$ is the pair with maximal vertex b_j (with respect to $\le_{P_2}$) in B_j, and $(\cdot, v_j)$ is the pair with minimal (with respect to $\le_{P_1}$) vertex v_j in B_j. It follows that $b_j \ge_{P_2} u \ge_{P_2} b$ and $v_j \le_{P_1} b_{first}$. We have therefore shown that

[9]The algorithm described in the proof of Lemma 6.4 would output B_i's exactly in the reverse of this order

there is $b_j \in I$ such that $b_j \geq_{P_2} b$ and $v_j \leq_{P_1} b_{first}$. Therefore, in particular, the vertex $b_i \geq_{P_2} b \in I$ that minimizes v_i also has $v_i \leq_{P_1} b_{first}$.

In conclusion, for $v = \min_{\leq_{P_1}}\{v_i \mid b_i \geq_{P_2} b\}$, we have shown that $\mathsf{dist}_{G'}(b, v) \leq (1+\varepsilon)\beta$, and that $v \leq_{P_1} b_{first}$. Therefore, v is an $\varepsilon\beta\text{-}\mathsf{first}^{\beta}_{\mathsf{G'}}(b, P_1)$ as required. ◻

This concludes the proof of Lemma 6.1, thus implying a simple labeling scheme for $\mathcal{L}^{\mathsf{P_2 \to P_1 \to P}}_{\beta,\gamma}$:

Lemma 6.6. *There exists a labeling scheme $\mathcal{L}^{\mathsf{P_2 \to P_1 \to P}}_{\beta,\gamma}$ with labels of size $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$.*

Proof. Every vertex $u \in P$ stores its index in P. Moreover, u also stores the sequences I_u and V_u as defined in Lemma 6.1. Additionally, for every $v \in V_u$ the label of u stores $\mathsf{first}^{\gamma}_{\mathsf{G\setminus\{u\}}}(v, P_1)$ and $\mathsf{first}^{\gamma}_{\mathsf{G\setminus\{u\}}}(v, P_2)$. Lemma 6.1 implies that the length of a label is $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$.

When two labels $\mathcal{L}^{\mathsf{P_2 \to P_1 \to P}}_{\beta}(b)$ and $\mathcal{L}^{\mathsf{P_2 \to P_1 \to P}}_{\beta}(f)$ are given, one can compute $x = \min_{\leq_{P_1}}\{v_i \mid b_i \geq_{P_2} b\}$ using the label of f and the index of b, and by the first property of Lemma 6.1 it is guaranteed that x is an $\varepsilon\beta\text{-}\mathsf{first}^{\beta}_{\mathsf{G'}}(b, P_1)$, and returns $\mathsf{first}^{\gamma}_{\mathsf{G\setminus\{f\}}}(x, P_1)$ and $\mathsf{first}^{\gamma}_{\mathsf{G\setminus\{f\}}}(x, P_2)$, that were stored explicitly, as required. ◻

6.2 The $\mathcal{L}^{\mathsf{P_2 \to P_2}}$ labeling

In this section, we provide a labeling scheme for P such that given the labels of two vertices $f <_P b$ one can compute $\varepsilon\beta\text{-}\mathsf{first}^{\beta}_{\mathsf{G\setminus(P_1 \cup \{f\})}}(b, P_2)$.

A labeling scheme for $\varepsilon = 1$. We start with a warm-up solution for $\varepsilon = 1$ (i.e., when the goal is to compute $\beta\text{-}\mathsf{first}^{\beta}_{\mathsf{G\setminus(P_1 \cup \{f\})}}(b, P_2)$).

For any vertex $v \in P$, let v_{last} be the last vertex in P that has a path to v in G of length at most β that does not touch any vertex of P before v. The pair (v_{last}, v) is called a *detour*. Let D be the set of all detours (v_{last}, v) for $v \in P$. Recall that the *size* of the detour (v_{last}, v) is the number of vertices in $P[v, v_{last}]$. We partition the set of detours D into exponential classes of sizes, as follows: For any $i \in [0, \lceil \log_{1.1} |P| + 1 \rceil]$ let D_i be the set of all detours whose size is in the range $[1.1^i, 1.1^{i+1})$. Let $x = 1.1^i$. We further partition every D_i into subsets: For any k, let $D_{i,k}$ be the subset of D_i that contains all the detours (v_{last}, v) where both v and v_{last} are in the subpath $W_{i,k} = P[0.1 \cdot k \cdot x, 0.1 \cdot k \cdot x + 1.2x]$ of P (which we call a *window*). An important property is that any vertex v of P is contained in $O(1)$ windows for a specific D_i and in $O(\log n)$ windows in total.

Claim 6.7 (Properties of the subsets)**.**

- *Any vertex v is contained in $O(\log n)$ windows.*
- *Let $m_{i,k} = W_{i,k}[\lceil |W|/2 \rceil]$ be the middle of $W_{i,k}$. For any detour $(v_{last}, v) \in D_{i,k}$ we have $v <_P m_{i,k} <_P v_{last}$.*

Let $H_{i,k}$ be the (unweighted) graph that is composed of $W_{i,k}$ enriched with an edge (v_{last}, v) for every detour $(v_{last}, v) \in D_{i,k}$. For two detours (u, v) and (x, y) with $u > x$ we say that the detours cross each other, and denote $(u, v) \not\leftharpoondown (x, y)$ if $u >_P x >_P v >_P y$. The following claim is used here only for $H = H_{i,k}$, we prove a stronger claim, regarding subgraphs of $H_{i,k}$, which will be useful later.

Claim 6.8. *Let H be a subgraph of $H_{i,k}$. If there is a u-to-v path S in H that does not touch any vertex of P before v, then there is a u-to-v path in H that does not touch any vertex of P before v and uses either one or two detours. If it uses two detours, then those detours must cross.*

Proof. Let $e_1 = (b_{last}, b)$ be a detour with the latest b_{last} in S, and let $e_2 = (a_{last}, v)$ be the last detour used by S. If $b = v$ then the path that goes from u to b_{last} on P and then on e_1 concludes the proof. Similarly, if $b_{last} = a_{last}$ then the path that goes from u to a_{last} and uses e_2 concludes the proof. Otherwise, we have $v <_P b$ and $a_{last} <_P b_{last}$. By Claim 6.7 we have $b <_P m_{i,k}$ and $m_{i,k} <_P a_{last}$. Therefore, $v <_P b <_P a_{last} <_P b_{last}$ and so the path that goes from u to b_{last}, on P uses e_1, then goes from b to a_{last} on P, and then uses e_2 contains exactly two crossing detours. $\square$

The following claim is a direct consequence of Claim 6.8.

Claim 6.9. *Let $v <_P u$ be two vertices in $W_{i,k}$. Then, if v is reachable from u in $H_{i,k}$ with a path that does not touch any vertex before v then there is a path in G from u to v that does not touch any vertex before v and is of length at most 2β.*

Proof. By Claim 6.8 there is a path S from u to v that uses at most two detours of $H_{i,k}$. By definition, each such detour corresponds to a path of length at most β in G. Moreover, since the total length of P is 0, we have that by replacing each edge in S with its corresponding path in G, we get a path in G of length at most 2β that does not touch any vertex in P before v. $\square$

Using Claim 6.9, we can use the auxiliary reachability procedure of Section 4.4 for every set $D_{i,k}$ in order to obtain our 2-approximation: Every $v \in P$ for every i, k such that $v \in W_{i,k}$, stores its reachability label in $H_{i,k}$, as well as the minimum vertex a such that there is a detour $(a_{last}, a) \in D_{i,k}$ with $a \leq_P v \leq_P a_{last}$, if exists. Recall that by Claim 6.7 every vertex of P is contained in $O(\log n)$ windows so the size of v's labels is $\tilde{O}(1)$.

We now explain how to use the information stored in the labels to complete our task. Let f and b be two vertices with $f <_P b$, and let $p = \mathsf{first}^{\beta}_{\mathsf{G}\setminus(\mathsf{P_1}\cup\{\mathsf{f}\})}(b, P_2)$. Consider the detour (p_{last}, p). This detour appears in some set $D_{i,k}$ (if (p_{last}, p) appears in more than one $D_{i,k}$, consider one arbitrary such set). It is clear that $p \leq_P b \leq_P p_{last}$ and therefore $b \in W_{i,k}$. If f is not in this window, then the label of b stores p explicitly as the minimum vertex that can be reached by a detour starting after b. Otherwise, if f is in the window, then we have the labels of the auxiliary reachability procedure of Section 4.4 for both f and b, which means we can find the first $p' >_P f$ that is reachable from b in $H_{i,k}$. Notice that $p' \leq_P p$ since $(p_{last}, p) \in D_{i,k}$ implies that p is reachable from b in $H_{i,k}$. It follows from Claim 6.9 that p' is a β-$\mathsf{first}^{\beta}_{\mathsf{G}\setminus(\mathsf{P_1}\cup\{\mathsf{f}\})}(b, P_2)$.

A labeling scheme for any ε. We are now ready to solve the problem for any ε. Let $A_1 = (u_1, v_1)$ and $A_2 = (u_2, v_2)$ with $u_1 > u_2$ be two detours in D. If $A_1 \not\leftarrow A_2$, and every path from u_1 to v_2 that does not touch any vertex before v_2 is of length more than $(1+\varepsilon)\beta$, then we say that A_1 bad-crosses A_2 and denote $A_1 \overset{\text{bad}}{\not\leftarrow} A_2$. Otherwise, we say that A_1 good-crosses A_2 and denote $A_1 \overset{\text{good}}{\not\leftarrow} A_2$. Notice that if A_1 does not cross A_2, then $A_1 \overset{\text{good}}{\not\leftarrow} A_2$.

From now on, we focus on a single $D_{i,k}$, and use the notation $D = D_{i,k}$, $m = m_{i,k}$, $W = W_{i,k}$ and $H = H_{i,k}$.

Lemma 6.10. *D can be partitioned into $t = O(1/\varepsilon^4)$ subsets $D_1, D_2, \ldots, D_t$ such that if $A_1, A_2 \in D_i$ then $A_1 \overset{good}{\not\leftarrow} A_2$.*

Proof. We present an algorithm that partitions D (this is exactly the same algorithm as in the proof of Lemma 6.4, but the definition of a bad-cross in this context is different). The algorithm runs in phases, creating the subset D_i in the i'th phase. At the beginning of every phase, we initialize a set $D_{good} = \emptyset$. During a phase, we examine the detours (u, v) of D in decreasing order of their first coordinate u. When examining a detour A, we check if A good-crosses all detours in D_{good}, and if so we add A into D_{good} and remove A from D. At the end of the i'th phase, we set $D_i = D_{good}$ and append D_i to the partition and if $D = \emptyset$ the algorithm terminates. Clearly $|D_{good}| \geq 1$, so the algorithm halts with a partition of the initial D.

By definition, for a detour $A_t = (u_t, v_t)$ in D_t, since $A_t \notin D_{t-1}$, there is a detour $A_{t-1} = (u_{t-1}, v_{t-1}) \in D_{t-1}$ with $u_{t-1} > u_t$ and $A_{t-1} \overset{\text{bad}}{\not\to} A_t$. Given A_{i+1} (with $i \geq 1$) we define $A_i = (u_i, v_i)$ in a similar way. Thus, we have a sequence $A_1, A_2, \ldots, A_t$ of detours such that $A_i \in D_i$ and $A_i \overset{\text{bad}}{\not\to} A_{i+1}$.

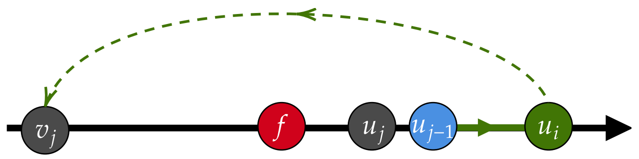


Figure 19: An illustration of the proof of Claim 6.11. The green dashed arrow represents a path of length at most $(1+\varepsilon)\beta$ from u_i to v_j that does not use a vertex before v_j on P. Such a path exists due to the assumption that $A_i \overset{\text{good}}{\not\to} A_j$. By starting with $P[u_{j-1}, u_i]$ (displayed as a thick green portion of P), we obtain a path of the same length from u_{j-1} to v_j that does not use any vertex before v_j, a contradiction to $A_j \overset{\text{bad}}{\not\to} A_{j-1}$.

Claim 6.11. *For any $1 \leq i < j \leq t$ we have $A_i \overset{bad}{\not\to} A_j$.*

Proof. By Claim 6.7, m is a position such that for any detour $(u, v) \in D$ we have $u >_P m >_P v$. Recall that $A_{k-1} \overset{\text{bad}}{\not\to} A_k$ means in particular that $A_{k-1} \not\to A_k$ thus, $u_{k-1} >_P u_k >_P m >_P v_{k-1} >_P v_k$. By simple induction $u_1 >_P u_2 >_P \ldots >_P u_t >_P m >_P v_1 >_P v_2 >_P \ldots >_P v_t$. Thus, $A_i \not\to A_j$. Assume by contradiction that $A_i \overset{\text{good}}{\not\to} A_j$, then there is a path S from u_i to v_j that does not touch any vertex before v_j and is of length is at most $(1+\varepsilon)\beta$. The path $S' = P[u_{j-1}, u_i] \cdot S$ (see Figure 19) is a path from u_{j-1} to v_j that does not use any vertex before v_j and $\mathsf{len}(S') = 0 + \mathsf{len}(S) \leq (1+\varepsilon)\beta$, a contradiction to $A_{j-1} \overset{\text{bad}}{\not\to} A_j$. □

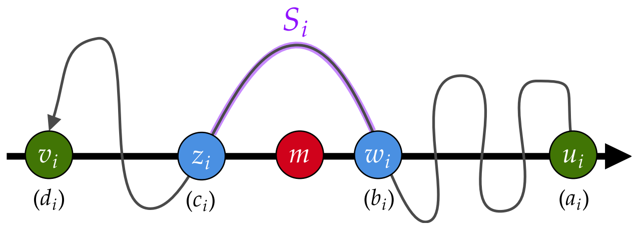


Figure 20: A demonstration of w_i, z_i and S_i. P_i is black u_i-to-v_i path.

For every detour $A_i = (u_i, v_i)$ in the sequence, we define the vertices w_i and z_i as follows (see Figure 20). Let P_i be the path corresponding to the detour A_i in G. The vertex z_i is defined to be the first vertex visited by P_i that is on P and has $z_i <_P m$ (notice that z_i is well defined because $v_i <_P m$ is in P_i). The vertex w_i is the vertex of P that precedes z_i in P_i (note that w_i is well defined because $u_i >_P m >_P z_i$ is the first vertex of P_i). Additionally, notice that $w_i \geq m$ due to the minimality of z_i in P_i. We also define $S_i = P_i[w_i, z_i]$, and observe that S_i is internally disjoint from P. If S_i is to the left (resp. right) of P, we say that (w_i, z_i) is a left (resp. right) pair. Consider the sequence $W = (w_1, z_1), (w_2, z_2), \ldots, (w_t, z_t)$.

By Erdős–Szekeres [ES35], there is a subsequence W' of W with $|W'| \geq \sqrt{t}$ such that the w_i's of the pairs in W' are monotone (either non-increasing or non-decreasing). By applying Erdős–Szekeres again, we have that there is a subsequence W'' of W' such that the z_i values are monotone and $|W''| \geq \sqrt[4]{t}$. Finally, each of the pairs in W'' is either a left or a right pair. We assume w.l.o.g that most of the pairs in W'' are left pairs. It follows that there is a subsequence C of W'' such that both w_i and z_i values are monotone, all pairs in C are left pairs, and $|C| \geq |W''|/2 \geq \frac{\sqrt[4]{t}}{2}$. To conclude the proof of Lemma 6.10, it remains only to prove that $|C| \leq \frac{1}{\varepsilon} + 1$, since then $\frac{1}{\varepsilon} + 1 \geq \frac{\sqrt[4]{t}}{2}$.

Claim 6.12. $|C| \leq 1/\varepsilon + 1$.

Proof. Let $C = (b_1, c_1), (b_2, c_2), \ldots (b_{|C|}, c_{|C|})$. For every $i \in [|C|]$ such that $(b_i, c_i) = (w_j, z_j)$ we denote $a_i = u_j$, $d_i = v_j$ (see Figure 20). From now on, we abuse notation by using A_i, P_i and S_i to refer to A_j, P_j and S_j respectively.

Recall that both b_i and c_i are monotone. We distinguish between two cases:

Case 1: either the b_i's or the c_i's are non-decreasing. We assume the c_i's are non-decreasing (the proof for the b_i's is symmetric). In this case, we have $a_1 \leq_P a_2 \leq_P \cdots \leq_P a_{|C|} < m \leq_P c_{|C|} \leq_P c_{|C|-1} \leq_P \cdots \leq_P c_1 \leq_P d_1 \leq_P d_2 \leq_P \cdots \leq_P d_{|C|}$.

We will prove by induction that for every $i \in [|C|]$ we have $\mathsf{len}(P_i[a_i, c_i]) \leq (1 - (i-1)\varepsilon)\beta$. For $i = 1$ the claim follows since $\mathsf{len}(P_1[a_1, c_1]) \leq \mathsf{len}(P_1) \leq \beta$. We assume the claim holds for i, and prove it holds for $i+1$. Notice that (by Claim 6.11) for any $i < |C|$ we have that the $A_i \overset{\text{bad}}{\not\to} A_{i+1}$. This means that the length of any path from a_i to d_{i+1} that does not use any vertex of P before d_{i+1} is more than $(1+\varepsilon)\beta$. In particular, for the path $S = P_i[a_i, c_i] \cdot P[c_i, c_{i+1}] \cdot P_{i+1}[c_{i+1}, d_{i+1}]$ we have $\mathsf{len}(S) > (1+\varepsilon)\beta$. Due to the induction hypothesis, $\mathsf{len}(P_i[a_i, c_i]) \leq (1 - (i-1)\varepsilon)\beta$. Therefore, we have $(1 - (i-1)\varepsilon)\beta + 0 + \mathsf{len}(P_{i+1}[c_{i+1}, d_{i+1}]) \geq (1+\varepsilon)\beta$ which leads to $\mathsf{len}(P_{i+1}[c_{i+1}, d_{i+1}]) \geq i\varepsilon r$. Finally, we have $\mathsf{len}(P_{i+1}[a_{i+1}, c_{i+1}]) = \mathsf{len}(P_{i+1}) - \mathsf{len}(P_{i+1}[c_{i+1}, d_{i+1}]) \leq \beta - i\varepsilon\beta = (1 - i\varepsilon)\beta$ as required. If $|C| > 1/\varepsilon + 1$ we get that $\mathsf{len}(P_{|C|}[a_{|C|}, c_{|C|}])$ is negative, a contradiction.

Case 2: both the b_i's and the c_i's are decreasing. We make the following claim: For every $i \in [|C|]$ there is a path E_i from a_i to c_1 with $\mathsf{len}(E_i) \leq (1 - (i-1)\varepsilon)r$. Moreover, b_i is on E_i and $E_i[b_i, c_1]$ is internally disjoint from P and is to the left of P. The claim holds for $i = 1$ by setting $E_1 = P_1[a_1, c_1]$. We assume the claim holds for i, and prove it for $i+1$.

Recall that P_{i+1} is the path corresponding to the detour (a_{i+1}, d_{i+1}) and $b_i >_P b_{i+1} \geq_P m >_P c_1 >_P c_{i+1}$. Since both $E_i[b_i, c_1]$ and $S_{i+1} = P_{i+1}[b_{i+1}, c_{i+1}]$ are internally disjoint from P and go to the left of P, they must intersect at some vertex $z \notin P$.

Since $A_i \overset{\text{bad}}{\not\to} A_{i+1}$, any path from a_i to d_{i+1} that does not use any vertex before d_{i+1} is of length more than $(1+\varepsilon)\beta$. In particular, for the path $S = E_i[a_i, z] \cdot P_{i+1}[z, d_{i+1}]$ we have $\mathsf{len}(S) > (1+\varepsilon)\beta$. Let $S' = P_{i+1}[a_{i+1}, z] \cdot E_i[z, c_1]$. Notice that S' is a path from a_{i+1} to c_1 that goes through b_{i+1} and $S'[b_{i+1}, c_1]$, is internally disjoint from P, and is to the left of P. It remains to show that $\mathsf{len}(S') \leq (1 - i\varepsilon)\beta$. By the induction hypothesis, $\mathsf{len}(E_i) \leq (1 - (i-1)\varepsilon)\beta$. By definition of a detour we have $\mathsf{len}(P_{i+1}) \leq \beta$. Notice that $\mathsf{len}(E_i) = \mathsf{len}(E_i[a_i, z]) + \mathsf{len}(E_i[z, c_1])$ and $\mathsf{len}(P_{i+1}) = \mathsf{len}(P_{i+1}[a_{i+1}, z]) + \mathsf{len}(P_{i+1}[z, d_{i+1}])$. Therefore $\mathsf{len}(E_i) + \mathsf{len}(P_{i+1}) = \mathsf{len}(S) + \mathsf{len}(S')$. Since $\mathsf{len}(S) > (1+\varepsilon)\beta$ we have $\mathsf{len}(S') \leq \beta + (1 - (i-1)\varepsilon)\beta - (1+\varepsilon)\beta = (1 - i\varepsilon)\beta$, as required. If $|C| > 1/\varepsilon + 1$, then there is a path from a_t to c_1 with negative length, a contradiction. □

This concludes the proof of Lemma 6.10 □

Let $\mathcal{D} = D_1, D_2, \ldots, D_t$ be the partition of D obtained by Lemma 6.10. For $D_i \in \mathcal{D}$, let H_{D_i} be the graph that contains W (the window corresponding to D) and all the detours of D_i (each detour

$(v_{last}, v) \in D_i$ corresponds to an edge from v_{last} to v in H_{D_i}). The following claim is an extention of Claim 6.9 for general ε.

Claim 6.13. *Let $D_i \in \mathcal{D}$ be a set and let $v < u$ be two vertices in W. Then, if v is reachable from u in H_{D_i} with a path not touching any vertex before v, then there is a path in G from u to v that does not touch any vertex before v and is of length at most $(1+\varepsilon)\beta$.*

Proof. By Claim 6.8 there is a path S in H_{D_i} from u to v that uses either a single detour, or two crossing detours. In the case where S uses a single detour, the corresponding path in G is trivially of length at most β. In the other case, let $A_1 \not\to A_2$ be the two detours. Since $A_1, A_2 \in D_i$, by definition of the partition it must be that $A_1 \overset{\text{good}}{\not\to} A_2$, and therefore there is a path in G of length at most $(1+\varepsilon)\beta$ from u_1 to $v_2 = v$ where $A_1 = (u_1, v_1)$ and $A_2 = (u_2, v_2)$, as required. □

Lemma 6.14. *There exists a labeling scheme $\mathcal{L}_{\beta}^{\mathsf{P_2} \to \mathsf{P_2}}$ with labels of size $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$.*

Proof. We are using the auxiliary reachability procedure of Section 4.4 in our labeling. For every $D = D_{i,k}$ for every subset of the partition $D_j \in \mathcal{D}$, each vertex in $W_{i,k}$ stores the auxiliary reachability label of v in H_{D_i}. Additionally, every vertex v in $W_{i,k}$ stores the minimum vertex a such that there is a detour $(a_{last}, a) \in D$ with $a \leq_P v \leq_P a_{last}$, if exists. Recall that by Claim 6.7 every vertex of P is contained in $O(\log n)$ windows, and by Lemma 6.10 every set is partitioned into $O(1/\varepsilon^4)$ subsets, so the accumulated size of labels kept in v is $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$.

We now explain how to use the information stored in the labels to complete our task. Let f and b be two vertices on P with $f <_P b$, and let $p = \mathsf{first}^{\beta}_{\mathsf{G}\setminus(\mathsf{P_1}\cup\{\mathsf{f}\})}(b, P_2)$. Consider the detour (p_{last}, p). This detour appears in some set $D_{i,k}$ (i and k can be determined by the size of (p_{last}, p) and the index of p). (If (p_{last}, p) appears in more than one $D_{i,k}$, consider one arbitrary such set.) Moreover, there exists a subset D_j of $D_{i,k}$, obtained via Lemma 6.10 with $(p_{last}, p) \in D_j$. It is clear that $p \leq_P b \leq_P p_{last}$ and therefore b is in $W_{i,k}$. If $f \in W_{i,k}$, then the label of b stores p explicitly as the minimum vertex that can be reached by a detour starting after b. Otherwise, if $f \in W_{i,k}$, then we have the auxiliary reachability labels of both f and b in H_{D_j}, which means we can find the first $p' >_P f$ that is reachable from b in H_{D_i}. Notice that $p' \leq_P p$ since $(p_{last}, p) \in D_j$ implies that p is reachable from b in H_{D_j}. It follows from Claim 6.13 that p' is an $\varepsilon\beta$-$\mathsf{first}^{\beta}_{\mathsf{G}\setminus(\mathsf{P_1}\cup\{\mathsf{f}\})}(b, P_2)$. □

7 The $\mathcal{L}^{\mathsf{P'} \xrightarrow{\mathsf{f}} \mathsf{P}}$ Labeling (Proof of Lemma 5.7)

In this section we prove Lemma 5.7. The setting in this section is as follows. G (for the ease of presentation we use here G as the name of the graph) is a graph, P and P' are two 0-length paths and $\alpha, \varepsilon \in \mathbb{R}^+$. Our goal is to develop a labeling scheme such that given the labels of a vertex a on P' and a vertex f not in $P' \cup P$, one can retrieve the index on P of some vertex $b \in P$ such that b is an $\varepsilon\alpha$-$\mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(a, P)$.

Similarly to the case of reachability, we shall define a set of canonical paths that facilitate the distribution of information about avoiding failed vertices. The choice of these canonical paths is more elaborate. The canonical paths are no longer disjoint, but an important feature we maintain is that every vertex f of G lies on a small number of canonical paths, so we can afford to store in f's label information about these paths when f fails. We work with an estimation γ that is a multiple of $\varepsilon\alpha$, such that $\gamma - \varepsilon\alpha < \mathsf{dist}_G(a, b) \leq \gamma$. Notice that, $\mathsf{first}^{\gamma}_{\mathsf{G}}(P, a) \leq_P b$ since $\gamma \geq \mathsf{dist}_G(a, b)$. We define $L_\gamma = \{(u, \mathsf{first}^{\gamma}_{\mathsf{G}}(u, P)) \mid u \in P' \text{ and } \mathsf{dist}_G(u, \mathsf{first}^{\gamma}_{\mathsf{G}}(u, P)) > \gamma - \varepsilon\alpha\}$.

We define a set $C = C_\gamma$ of canonical paths (which we call γ-legitimate paths) as follows.

Definition 7.1 (γ-legitimate set of paths)**.** *A set C of paths in G is a γ-*legitimate set of paths for *L if it has the following properties:*

- *For each $S \in C$ we have $\mathsf{len}(S) \le (1+\varepsilon)\gamma$ and $S = S_1 \cdot S_2$ such that both S_1 and S_2 are shortest paths.*
- *For every $(c, c_{first}) \in L$ let $u \ge_{P'} c$ be the first vertex of P' with a path $(u \rightsquigarrow u') \in C$. If there are several such paths, let u' be the first (in P) among all vertices u'. It holds that u exists, and u' is a $\varepsilon\gamma$-$\mathsf{first}^{\gamma}_{\mathsf{G}}(c, P)$.*
- *For every vertex f of G there are $O(1/\varepsilon)$ paths in C going through f.*

We call the paths in C *canonical.* For every $c \in P'$ let $u \ge_{P'} c$ be the first vertex of P' with a path $(u \rightsquigarrow u') \in C$ (if exists). If there are several such paths, let u' be the first (in P) among all values u'. If u exists, we say that $(u \rightsquigarrow u')$ is *the canonical path of c.*

In Section 7.1 we prove the following lemma.

Lemma 7.2. *For every $G, P', P, \varepsilon, \gamma, L$ there is a γ-legitimate set of paths.*

Denote such a set by $C = C_\gamma$. We partition P' into maximal contiguous intervals of vertices c with $(c, \cdot) \in L_\gamma$ that have the same canonical path. We call these intervals the canonical γ-intervals of P' and P. For every such interval I of vertices p with the same canonical path $S^\gamma_I = (x \rightsquigarrow y)$, we call S^γ_I the canonical path of interval I.

In Section 7.2, we prove the following.

Lemma 7.3. *For a graph G, two paths P and P' of length 0, a path $A = (u \rightsquigarrow u')$ from the last vertex of P' to u' which is the first vertex of P and numbers $\alpha, \varepsilon \in \mathbb{R}^+$ such that:*

1. *$\mathsf{len}(A) \le (1+\varepsilon)\alpha$.*
2. *$A = A_1 \cdot A_2$ such that A_1 and A_2 are shortest paths in G.*

There exists a labeling scheme $\mathcal{L}^{\mathsf{a}\rightsquigarrow\mathsf{P}} = \mathcal{L}^{\mathsf{a}\rightsquigarrow\mathsf{P}}_{G,P',A,P,\alpha,\varepsilon}(v)$ such that given the labels of two vertices $a \in P'$ and $f \in A \setminus P'$, one can obtain a vertex $b \in P$ such that $\mathsf{dist}_{G\setminus\{f\}}(a,b) \le (1+\varepsilon)\alpha$ and $b \le_P \mathsf{first}^{\alpha}_{\mathsf{G}\setminus\mathsf{A}[\ldots\mathsf{f}]}(a,P)$ or conclude that $\mathsf{first}^{\alpha}_{\mathsf{G}\setminus\mathsf{A}[\ldots\mathsf{f}]}(a,P) = null$. The size of each label is $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$.

We are now ready to prove Lemma 5.7, which we restate here.

Lemma 5.7. *There exists a labeling scheme $\mathcal{L}^{\mathsf{P}'\xrightarrow{\mathsf{f}}\mathsf{P}} = \mathcal{L}^{\mathsf{P}'\xrightarrow{\mathsf{f}}\mathsf{P}}_{H,P,P',\alpha,\varepsilon}$ where H is a planar graph, P and P' are two 0-length paths and $\alpha, \varepsilon \in \mathbb{R}^+$. Given the labels of a vertex a on P' and a vertex f not in $P' \cup P$, one can retrieve the index on P of some vertex $b \in P$ such that b is an $\varepsilon\alpha$-$\mathsf{first}^{\alpha}_{\mathsf{H}\setminus\mathsf{f}}(a,P)$. The size of each label is $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$.*

Proof. Let $\varepsilon' = \varepsilon/2$ be an approximation factor and let $\Gamma = \{i\varepsilon'\alpha \mid i \in [0, \lceil 1/\varepsilon' \rceil]\}$ be the set of multiples of $\varepsilon'\alpha$. For every $\gamma \in \Gamma$ let C_γ be a set of γ-canonical paths, computed with respect to ε' (such a set exists due to Lemma 7.2).

For every vertex $v \in P'$ and for every $\gamma \in \Gamma$ such that $(v, \cdot) \in L_\gamma$ the label of v stores: The endpoints (u_γ, u'_γ) of its canonical path $S_{I_v} \in C_\gamma$. In addition v stores $\mathcal{L}^{\mathsf{a}\rightsquigarrow\mathsf{P}}_{G,P'[1,u_\gamma],S_{I_v},P[u'_\gamma,|P|],\alpha,\varepsilon'}(v)$

For every vertex $v \in G \setminus P'$ and for every $\gamma \in \Gamma$ and for every $S = (u \rightsquigarrow u') \in C_\gamma$ such that $v \in S$ the label of v stores: $\mathsf{first}^{\alpha+2\varepsilon'\alpha}_{\mathsf{G}\setminus\{\mathsf{v}\}}(u,P)$ and $\mathcal{L}^{\mathsf{a}\rightsquigarrow\mathsf{P}}_{G,P'[1,u],S,P[u',|P|],\alpha,\varepsilon'}(v)$.

Size. By Definition 7.1 every $v \in G$ is on $O(1/\varepsilon')$ canonical paths. Moreover, for every canonical path, the size of $\mathcal{L}^{\mathsf{a} \rightsquigarrow \mathsf{P}}(v)$ is $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$ by Lemma 7.3. Thus, the total size of every $\mathcal{L}^{\mathsf{P}' \xrightarrow{\mathsf{f}} \mathsf{P}}(v)$ is $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$.

Decoding. Given the labels of $a \in P'$ and $f \in G \setminus P'$ one can compute the index on P of some vertex b such that b is an $\varepsilon\alpha$-$\mathsf{first}^{\alpha}_{\mathsf{G}\setminus\mathsf{f}}(a, P)$ as follows. For every $\gamma \in \Gamma$ such that $(v, \cdot) \in L_\gamma$ let $S_\gamma = (u_\gamma, u'_\gamma)$ be the canonical path of a in C_γ. If $f \notin S_\gamma$ (which can be obtained from the label of f) then $b_\gamma = u'_\gamma$. Otherwise, let b^1_γ be a vertex on P such that $\mathsf{dist}_{G\setminus\{f\}}(a, b^1_\gamma) \le (1 + \varepsilon')\alpha$ and $b^1_\gamma \le_P \mathsf{first}^{\alpha}_{\mathsf{G}\setminus\mathsf{S}_\gamma[\ldots\mathsf{f}]}(a, P)$ or set $b^1_\gamma = \mathit{null}$ if $\mathsf{first}^{\alpha}_{\mathsf{G}\setminus\mathsf{S}_\gamma[\ldots\mathsf{f}]}(a, P) = \mathit{null}$, by using $\mathcal{L}^{\mathsf{a} \rightsquigarrow \mathsf{P}}$. Let b^2_γ be $\mathsf{first}^{\alpha+2\varepsilon'\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(u_\gamma, P)$ (stored in the label of f). Let $b_\gamma = \min_{\le_P}\{b^1_\gamma, b^2_\gamma\}$. Finally, we return $b = \min_{\le_P}\{b_\gamma \mid \gamma \in \Gamma\}$.

Correctness. First, we show that $\mathsf{dist}_{G\setminus\{f\}}(a, b) \le (1 + 2\varepsilon')\alpha = (1 + \varepsilon)\alpha$.

- If $b = b_\gamma = u'_\gamma$ it must be that $f \notin S_\gamma$ and therefore by Definition 7.1 (first property) we have $\mathsf{dist}_{G\setminus\{f\}}(a, u'_\gamma) \le \mathsf{len}(P[a, u_\gamma] \cdot S_\gamma) \le (1 + \varepsilon')\gamma \le (1 + \varepsilon)\alpha$.
- If $b = b^1_\gamma$ for some γ, then by Lemma 7.3 we have $\mathsf{dist}_{G\setminus\{f\}}(a, b^1_\gamma) \le (1 + \varepsilon')\alpha \le (1 + \varepsilon)\alpha$.
- Otherwise, if $b = b^2_\gamma = \mathsf{first}^{(1+2\varepsilon')\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(u_\gamma, P)$, by definition $\mathsf{dist}_{G\setminus\{f\}}(a, b^2_\gamma) \le (1+2\varepsilon')\alpha = (1+\varepsilon)\alpha$.

It remains to prove that $b \le_P \mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(a, P)$. Let $b' = \mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(a, P)$ and let R be a shortest path from a to b' in $G \setminus \{f\}$. Let $\gamma = x \in \Gamma \mid x \ge \mathsf{dist}_G(a, b')\}$, and let $S_\gamma = (u_\gamma, u'_\gamma)$ be the canonical path of a in C_γ. We distinguish between three cases:

- If $f \notin S_\gamma$, by Definition 7.1 (second property) $b_\gamma = u'_\gamma \le_P \mathsf{first}^{\gamma}_{\mathsf{G}}(a, P) \le_P b'$ by definition of γ.
- If $R \cap S_\gamma[u_\gamma, f] = \emptyset$, then b^1_γ is a vertex on P such that $b^1_\gamma \le_P \mathsf{first}^{\alpha}_{\mathsf{G}\setminus\mathsf{S}_\gamma[\mathsf{u}_\gamma,\mathsf{f}]}(a, P) \le_P b'$ since there exists a path (R) in $G \setminus S_\gamma[u_\gamma, f]$ from a to b' of length at most α.
- Otherwise (if $R \cap S_\gamma[u_\gamma, f) \ne \emptyset$) we have that $b^2_\gamma \le_P b'$ by the following claim.

Claim 7.4. $b^2_\gamma = \mathsf{first}^{\alpha+2\varepsilon'\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(u_\gamma, P) \le_P \mathsf{first}^{\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(a, P) = b'$.

Proof. Recall that $(a, \mathsf{first}^{\gamma}_{\mathsf{G}}(a, P)) \in L_\gamma$ which means that $\mathsf{dist}_G(a, \mathsf{first}^{\gamma}_{\mathsf{G}}(a, P)) > \gamma - \varepsilon\alpha$. Assume by contradiction that $b' <_P b^2_\gamma$. Let z be some vertex in $R \cap S_\gamma[u_\gamma, f)$. Let $\ell = \mathsf{len}(S_\gamma[u_\gamma, z])$. It must be that $\mathsf{len}(R[u_\gamma, z]) \le \ell - 2\varepsilon'\alpha$ since otherwise, $S_\gamma[u_\gamma, z] \cdot R[z, b']$ is a path of length at most $\alpha + 2\varepsilon'\alpha$, which contradicts the minimality of $\mathsf{first}^{\alpha+2\varepsilon'\alpha}_{\mathsf{G}\setminus\{\mathsf{f}\}}(u_\gamma, P)$ on P. Thus, $\mathsf{len}(R[u, z]) \le \ell - 2\varepsilon'\alpha$. Now, consider the path $R[a, z] \cdot S_\gamma[z, u'_\gamma]$ (in G). This is a path of length at most $(1+\varepsilon')\gamma - 2\varepsilon'\alpha \le \gamma - \varepsilon'\alpha$ from a to u'_γ in G. Therefore, $u'_\gamma \ge_P \mathsf{first}^{\gamma}_{\mathsf{G}}(a, P)$. By Definition 7.1 (second property) it must be that $u'_\gamma \le_P \mathsf{first}^{\gamma}_{\mathsf{G}}(a, P)$ which leads to $u'_\gamma = \mathsf{first}^{\gamma}_{\mathsf{G}}(a, P)$. We have shown $\mathsf{dist}_G(a, \mathsf{first}^{\gamma}_{\mathsf{G}}(a, P)) < \gamma - \varepsilon\alpha$ this contradicts $(a, \mathsf{first}^{\gamma}_{\mathsf{G}}(a, P)) \in L_\gamma$. □

In each case we have shown that $b_\gamma \le_P b'$, and therefore by the minimality of b on P, we have $b \le_P b'$. □

7.1 Selecting canonical paths

In this section we solve the following problem. We are given a weighted directed planar graph G with two paths P' and P where the total length of each path is 0. In addition, we are given a length bound γ, and an approximation factor $\varepsilon > 0$. Finally, we are given a set $L \subseteq P' \times P$ of pairs (u, u_{first}) where $u \in P'$ and $u_{first} = \mathsf{first}^{\gamma}_{\mathsf{G}}(u, P)$. Notice that L has the following properties:

1. **Uniqueness:** For every $u \in P'$, there is at most one pair (u, u_{first}) in L.
2. **Proximity:** Each pair $(u, u_{first}) \in L$ has $\mathsf{dist}_G(u, u_{first}) \leq \gamma$.
3. **Monotonicity:** For every two pairs $(u, u_{first}), (v, v_{first}) \in L$, we have that if $u <_{P'} v$ then $u_{first} \leq_P v_{first}$.

Our goal in this problem is to find a *γ-legitimate set of canonical paths for L* as defined in Definition 7.1:

Definition 7.1 (γ-legitimate set of paths). *A set C of paths in G is a* γ-legitimate set of paths for *L if it has the following properties:*

- *For each $S \in C$ we have $\mathsf{len}(S) \leq (1+\varepsilon)\gamma$ and $S = S_1 \cdot S_2$ such that both S_1 and S_2 are shortest paths.*
- *For every $(c, c_{first}) \in L$ let $u \geq_{P'} c$ be the first vertex of P' with a path $(u \rightsquigarrow u') \in C$. If there are several such paths, let u' be the first (in P) among all vertices u'. It holds that u exists, and u' is a $\varepsilon\gamma$-$\mathsf{first}^{\gamma}_{\mathsf{G}}(c, P)$.*
- *For every vertex f of G there are $O(1/\varepsilon)$ paths in C going through f.*

We call the paths in C *canonical* paths.

The rest of this section is dedicated to the proof of Lemma 7.2.

Lemma 7.2. *For every $G, P', P, \varepsilon, \gamma, L$ there is a γ-legitimate set of paths.*

For every $(u, u_{first}) \in L$ let A_u be a shortest path from u to u_{first} (notice that $\mathsf{len}(A_u) \leq \gamma$). For every $u <_{P'} v$ if $A_u \cap A_v \neq \emptyset$ we say that A_u crosses[10] A_v and denote $A_u \not\leftarrow A_v$. For every $A_u \not\leftarrow A_v$ (with $u <_{P'} v$) let $z_{u,v}$ be the first vertex on A_v which is also on A_u (see Figure 21). Let $P_{u,v} = A_v[v, z_{u,v}]$ and $S_{u,v} = A_u[z_{u,v}, u_{first}]$ be the prefix of A_v until $z_{u,v}$ and the suffix of A_u from $z_{u,v}$, respectively. Let $A_{u,v} = P_{u,v} \cdot S_{u,v}$. If $\mathsf{len}(A_{u,v}) \leq (1+\varepsilon)\gamma$ we say that A_u good-crosses A_v and we denote $A_u \overset{\text{good}}{\not\leftarrow} A_v$; otherwise A_u bad-crosses A_v, and we denote $A_u \overset{\text{bad}}{\not\leftarrow} A_v$.

Algorithm. Consider the pairs (u, u_{first}) in L to be ordered according to the $<_{P'}$ order of u. We construct the set C using the following algorithm. The algorithm initializes $C = \emptyset$ and a pair (a, a_{first}) to be the first pair in L. The algorithm repeats the following procedure until the halting condition is met. The algorithm finds the last pair $(b, b_{first}) \in L$ such that $A_a \overset{\text{good}}{\not\leftarrow} A_b$. We distinguish between two cases: If $a \neq b$, the algorithm inserts $A_{a,b}$ to C and sets $(a, a_{first}) \leftarrow (b, b_{first})$. Otherwise, if $a = b$, the algorithm inserts $A_{a,b} = A_a$ to C and sets (a, a_{first}) to be the pair in L following (a, a_{first}); if there is no such pair, the algorithm halts and returns C.

We prove that C is indeed a γ-legitimate set of canonical paths.

[10] A more natural terminology would be to say that A_u intersects A_v. However, due to a similarity to Sections 6.1 and 6.2 we prefer to use the same terminology of crossing here.

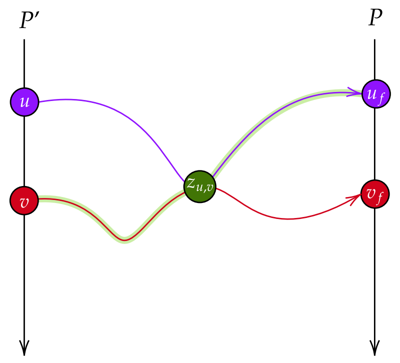


Figure 21: An illustration of the path $A_{u,v} = P_{u,v} \cdot S_{u,v}$.

Claim 7.5. *For each $S \in C$ we have $\mathsf{len}(S) \leq (1+\varepsilon)\gamma$ and $S = S_1 \cdot S_2$ such that both S_1 and S_2 are shortest paths.*

Proof. Let $S \in C$, by the algorithm $S = A_{a,b}$ for some vertices a and b such that $A_a \overset{\text{good}}{\not\to} A_b$. By definition of $\overset{\text{good}}{\not\to}$ we have $\mathsf{len}(S) = \mathsf{len}(A_{a,b}) \leq (1+\varepsilon)\gamma$. Moreover, $S = A_{a,b} = P_{a,b} \cdot S_{a,b}$ and both $P_{a,b}$ and $S_{a,b}$ are shortest paths (as they are subpaths of shortest paths). $\square$

Claim 7.6. *For every $(c, c_{first}) \in L$ let $u \geq_{P'} c$ be the first vertex of P' with a path $(u \rightsquigarrow u') \in C$. If there are several such paths, let u' be the first (in P) among all vertices u'. It holds that u exists, and u' is a $\varepsilon\gamma$-$\mathsf{first}^{\gamma}_{\mathsf{G}}(c, P)$.*

Proof. Let (w, w_{first}) be the last pair in L. Notice that the algorithm necessarily iterates the last pair, and therefore adds a path starting in $w \geq_{P'} c$ to C. Therefore, the vertices u and u' exist. Let $A_{a,b} = (u \rightsquigarrow u')$. In particular $u = b$ and $u' = a_{first}$. Notice that $b \geq_{P'} c$. Moreover it must be that $a \leq_{P'} c$ since otherwise C would also contain some other path $A_{x,a}$ or $A_{x,a-1}$ (where $(a-1,\cdot) \in L$ is the pair preceding (a, a_{first}) in L and $x \in P'$) contradicting the minimality of b. It follows from the monotonicity of L that $u' = a_{first} \leq_P c_{first} = \mathsf{first}^{\gamma}_{\mathsf{G}}(c, P)$.

Consider the path $S' = P'[c,b] \cdot A_{a,b}$ from c to a_{first}. The path S' has length $\mathsf{len}(S') = 0 + \mathsf{len}(A_{a,b}) \leq (1+\varepsilon)\gamma$ by Claim 7.5. It follows that a_{first} is an $\varepsilon\gamma$-$\mathsf{first}^{\gamma}_{\mathsf{G}}(c, P)$. $\square$

It remains to prove the last property. For a set $C' \subseteq \{A_u \mid (u, u_{first}) \in L\}$ and a vertex f we say that C' is an *f-bad* set if f is on every path in C', and every two different paths in C' bad-cross each other. We prove the following helpful lemma regarding f-bad sets.

Lemma 7.7. *Let C' be an f-bad set for some vertex f of G. It must hold that $|C'| \leq 1/\varepsilon + 1$*

Proof. Let $C' = A_{a_1}, A_{a_2}, \ldots, A_{a_t}$ be the paths in C' ordered such that $a_1 \leq_{P'} a_2 \leq_{P'} \ldots \leq_{P'} a_t = a_{|C'|}$. We claim that for every $i \in [t]$, we have $\mathsf{len}(A_{a_i}[f, (a_i)_{first}]) \leq (1-(i-1)\varepsilon)\gamma$.

For $i = 1$, this is true since f is on A_{a_1} and therefore $\mathsf{len}(A_{a_1}[f, (a_1)_{first}] \leq \mathsf{len}(A_{a_1}) \leq \gamma$. We thus assume the claim holds for i, and prove it for $i+1$.

Consider the path $S^* = A_{a_{i+1}}[a_{i+1}, f] \cdot A_{a_i}[f, (a_i)_{first}]$. We first claim that $\mathsf{len}(S^*) \geq \mathsf{len}(A_{a_i,a_{i+1}})$. Recall that $z = z_{a_i,a_{i+1}}$ is the first vertex in $A_{a_{i+1}}$ that is also on A_{a_i}. Notice that $f \geq_{A_{a_{i+1}}} z$. There are two cases to consider:

- if $f \geq_{A_{a_i}} z$ (see Figure 22): Since both A_{a_i} and $A_{a_{i+1}}$ are shortest paths, we can assume that $A_{a_{i+1}}[z, f] = A_{a_i}[z, f]$. Therefore,

$$A_{a_i,a_{i+1}} = A_{a_{i+1}}[a_{i+1}, z] \cdot A_{a_i}[z, (a_i)_{first}]$$

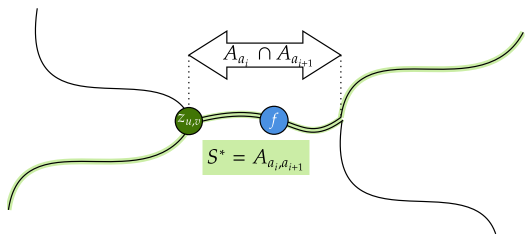


Figure 22: An illustration of the case $f \geq_{A_{a_i}} z$ where $S^* = A_{a_i,a_{i+1}}$.

$$
\begin{aligned}
&= A_{a_{i+1}}[a_{i+1}, z] \cdot A_{a_i}[z, f] \cdot A_{a_i}[f, (a_i)_{first}] \\
&= A_{a_{i+1}}[a_{i+1}, z] \cdot A_{a_{i+1}}[z, f] \cdot A_{a_i}[f, (a_i)_{first}] \\
&= A_{a_{i+1}}[a_{i+1}, f] \cdot A_{a_i}[f, (a_i)_{first}] = S^*
\end{aligned}
$$

Hence, $\mathsf{len}(S^*) = \mathsf{len}(A_{a_i,a_{i+1}})$.

- if $f \leq_{A_{a_i}} z$ (see Figure 23): In this case, there exists two subpaths, $A_{a_i}[f, z]$ and $A_{a_{i+1}}[z, f]$ So, we have

$$
\begin{aligned}
S^* &= A_{a_{i+1}}[a_{i+1}, f] \cdot A_{a_i}[f, (a_i)_{first}] \\
&= A_{a_{i+1}}[a_{i+1}, z] \cdot A_{a_{i+1}}[z, f] \cdot A_{a_i}[f, z] \cdot A_{a_i}[z, (a_i)_{first}] \\
\text{and} \quad A_{a_i,a_{i+1}} &= A_{a_{i+1}}[a_{i+1}, z] \qquad\qquad \cdot \qquad\qquad A_{a_i}[z, (a_i)_{first}]
\end{aligned}
$$

Therefore, $\mathsf{len}(S^*) = \mathsf{len}(A_{a_i,a_{i+1}}) + \mathsf{len}(A_{a_{i+1}}[z, f] \cdot A_{a_i}[f, z]) \geq \mathsf{len}(A_{a_i,a_{i+1}})$.

Since A_{a_i} and $A_{a_{i+1}}$ are two different paths in C', it must be that $A_{a_i} \overset{\text{bad}}{\not\sim} A_{a_{i+1}}$ which means $\mathsf{len}(A_{a_i,a_{i+1}}) > (1+\varepsilon)\gamma$. Therefore, $\mathsf{len}(S^*) > (1+\varepsilon)\gamma$.

By definition, $\gamma \geq \mathsf{len}(A_{a_{i+1}}) = \mathsf{len}(A_{a_{i+1}}[a_{i+1}, f]) + \mathsf{len}(A_{a_{i+1}}[f, (a_{i+1})_{first}])$, which implies

$$\mathsf{len}(A_{a_{i+1}}[f, (a_{i+1})_{first}]) \leq \gamma - \mathsf{len}(A_{a_{i+1}}[a_{i+1}, f]) \tag{1}$$

By the induction hypothesis $\mathsf{len}(A_{a_i}[f, (a_i)_{first}]) \leq (1 - (i-1)\varepsilon)\gamma$ which implies

$$0 \leq (1 - (i-1)\varepsilon)\gamma - \mathsf{len}(A_{a_i}[f, (a_i)_{first}]) \tag{2}$$

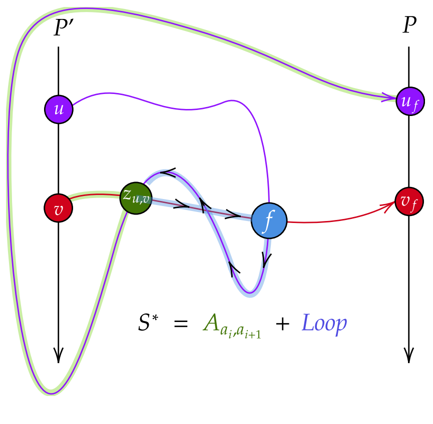


Figure 23: An illustration of the case $f \leq_{A_{a_i}} z$.

By summing Equations (1) and (2), we obtain

$$\begin{aligned}\mathsf{len}(A_{a_{i+1}}[f,(a_{i+1})_{first}]) &\le \gamma + (1-(i-1)\varepsilon)\gamma - \mathsf{len}(A_{a_{i+1}}[a_{i+1},f]) - \mathsf{len}(A_{a_i}[f,(a_i)_{first}])\\ &\le (2-(i-1)\varepsilon)\gamma - \mathsf{len}(S^*)\\ &< (2-(i-1)\varepsilon)\gamma - (1+\varepsilon)\gamma = (1-i\varepsilon)\gamma\end{aligned}$$

as required.

Assume by contradiction that $t = |C'| > 1/\varepsilon + 1$, and deduce that $A_{a_t}[f,(a_t)_{first}]$ is a path of negative length, in contradiction. Thus, $|C'| \le 1/\varepsilon + 1$ as required. □

Equipped with Lemma 7.7, we are finally ready to prove the last property.

Lemma 7.8. *For every vertex f of G there are $O(1/\varepsilon)$ canonical paths going through f.*

Proof. Let f be a vertex of G and let $C_f = \{A_{a,b} \in C \mid f \text{ is on } A_{a,b}\}$. Recall that for every $A_{a,b} \in C_f$, $A_{a,b} = P_{a,b} \cdot S_{a,b}$. We define two subsets $C_f^P = \{A_{a,b} \in C_f \mid f \in P_{a,b}\}$ and $C_f^S = \{A_{a,b} \in C_f \mid f \in S_{a,b}\}$. Note that $C_f^P \cup C_f^S = C_f$.

We make the following claim.

Claim 7.9. $|C_f^S| \le 2/\varepsilon + 2$.

Proof. We partition C_f^S into two sets C_E and C_O. C_E and C_O contain canonical paths of C_f^S that were added to C in even and odd steps of the algorithm, respectively. We proceed by proving that $C'_E = \{A_u \mid A_{u,v} \in C_E\}$ is f-bad, which implies that $|C_E| = |C'_E| \le 1/\varepsilon + 1$ via Lemma 7.7. The size of C_O can be bounded using identical arguments, which yields the claim.

Let $C_E = A_{a_1,b_1}, A_{a_2,b_2}, \ldots, A_{a_t,b_t}$ be the canonical paths of C_E ordered such that $a_1 \le_{P'} a_2 \le_{P'} \ldots \le_{P'} a_t$. Clearly, all the paths in C'_E intersect in f, so we just need to show that every pair of different path forms a bad-cross. We start by observing that if the algorithm adds $A_{a',b'}$ to C, and two iterations later adds $A_{a'',b''}$ to C, it must be the case that $a'' >_{P'} b'$. It follows from the monotonicity of the a values of $A_{a,b}$ added by the algorithm, and by the fact that C_E contains only paths that were added on an even step of the algorithm that $b_i <_{P'} a_j$ for every $i < j \in [t]$.

We claim that for every $i < j \in [t]$, we have $A_{a_i} \overset{\text{bad}}{\not\to} A_{a_j}$. This follows from $A_{a_i} \not\to A_{a_j}$ (due to the intersection of S_{a_i} and S_{a_j} in f), and from b_i ($<_{P'} a_j$) being the last vertex on P' that participates in a pair of L and has $A_{a_i} \overset{\text{good}}{\not\to} A_{b_i}$. □

In addition, we make the following claim.

Claim 7.10. $|C_f^P| \le 3/\varepsilon + 3$.

Proof. The proof is very similar to the proof of Claim 7.9, but require a some subtle adjustments.

We partition C_f^P into three sets C_0, C_1 and C_2. C_0, C_1 and C_2 contain canonical paths of C_f^P that were added to C in steps numbered with 0, 1, and 2 modulo 3 of the algorithm, respectively (i.e. C_1 contains path added in steps $1, 4, 7, \ldots$). We proceed by proving that $C'_0 = \{A_v \mid A_{u,v} \in C_0\}$ is f-bad, which implies that $|C_0| = |C'_0| \le 1/\varepsilon + 1$ via Lemma 7.7. The sizes of C_1 and C_2 can be bounded using identical arguments, which yields the claim.

Let $C_0 = A_{a_1,b_1}, A_{a_2,b_2}, \ldots, A_{a_t,b_t}$ be the canonical paths of C_0 ordered such that $a_1 \le_{P'} a_2 \le_{P'} \ldots \le_{P'} a_t$. Clearly, all the paths in C'_0 intersect in f, so we just need to show that every pair of different path forms a bad-cross.

We claim that for every $i < j \in [t]$, we have $A_{b_i} \overset{\text{bad}}{\not\to} A_{b_j}$, by considering two cases.

Case 1: $a_i = b_i$**.** In this case, no pair (a', b') in L after $(a_i, (a_i)_{first})$ has $A_{a_i} \overset{\text{good}}{\not\to} A_{a'}$. In particular, $A_{a_i} \overset{\text{bad}}{\not\to} A_{b_j}$ since $a_i <_{P'} a_j \leq_{P'} b_j$. Since $a_i = b_i$, we have $A_{b_i} \overset{\text{bad}}{\not\to} A_{b_j}$ as required.

Case 2: $a_i \neq b_i$**.** Let $3k$ be the iteration number in which A_{a_i,b_i} was added to C. In this case, the algorithm will process the pair $(b_i, (b_i)_{first})$ in iteration $3k+1$. In this iteration, the path $A_{b_i,c}$ would be added to C such that $(c, c_{first}) \in L$ is the last pair such that $A_{b_i} \overset{\text{good}}{\not\to} A_c$. In iteration $3k+2$, some path will be added to C, and all following iterations (in particular, the iteration in which A_{a_j,b_j} was added) will process pairs (a', b') such that $b' \geq_{P'} a' >_{P'} c$. Due to the maximality of c and the fact that A_{b_j} crosses A_{b_i} in f, we must have $A_{b_i} \overset{\text{bad}}{\not\to} A_{b_j}$ as required. ☐

Combining Claims 7.9 and 7.10 we complete the proof of Lemma 7.8. ☐

7.2 The $\mathcal{L}^{\mathsf{a} \rightsquigarrow \mathsf{P}}$ labeling (proof of Lemma 7.3)

In this section, we prove Lemma 7.3; The settings are as follows. We are given a graph G, two paths P and P' of length 0, a path $A = (u \rightsquigarrow u')$ from the last vertex of P' to u' which is the first vertex of P and numbers $\alpha, \varepsilon \in \mathbb{R}^+$ such that:

1. $\mathsf{len}(A) \leq (1+\varepsilon)\alpha$.

2. $A = A_1 \cdot A_2$ such that A_1 and A_2 are shortest paths in G.

For most of this section, we focus on a vertex $a \in P'$. For every vertex $f \in A$, we define $b^f_{suf} = \mathsf{first}^{\alpha}_{\mathsf{G} \setminus \mathsf{A[u,f]}}(a, P)$, where $P[|P|]$ is the last vertex of P. Note that the set of vertices f such that b^f_{suf} is well defined forms a (possibly empty) prefix of A. We denote as z' the last vertex on A such that $b^{z'}_{suf}$ is well defined (if there is no such vertex, we say that $z' = null$). We make the following observation regarding b^f_{suf}.

Observation 7.11. *For every $f_1 \leq_A f_2 \leq_A z'$ we have $b^{f_1}_{suf} \leq_P b^{f_2}_{suf}$.*

Most of the section is devoted to the proof of the following technical lemma.

Lemma 7.12. *If $z' \neq null$, there are sequences $F = (u = f_1 \leq_A f_2 \leq_A \ldots \leq_A f_t = z')$ and $B = (b_1, b_2, \ldots, b_{t-1})$, with $t = O(1/\varepsilon)$, such that for every $f_i \leq_A f <_A f_{i+1}$, we have:*

1. $b^f_{suf} \geq_P b_i$.

2. $\mathsf{dist}_{G \setminus \{f\}}(a, b_i) \leq (1+\varepsilon)\alpha$.

Proof. First, we reduce the problem of finding such sequences to the case where A is internally disjoint from P'. Let x be the first vertex on P' that is on A. Let $\tilde{P}' = P'[a, x]$ and $\tilde{A} = A[x, u']$, and assume we have sequences $\tilde{F}$ and $\tilde{B}$ as in the statement of the lemma for $\tilde{P}'$ and $\tilde{A}$ in $G \setminus A[u, x)$. Note that $\tilde{P}'$ and $\tilde{A}$ are internally disjoint due to the definition of x. We claim that the sequences $F = (u) \cdot \tilde{F}$ and $B = (u') \cdot \tilde{B}$ satisfy the lemma, for P' and A, due to the following.

- For every $f \geq_A x$, the sequences $\tilde{B}$ and $\tilde{F}$ are satisfactory for P' and A, as b^f_{suf} in G is the same as b^f_{suf} in $G \setminus A[u, x)$ for $f \geq_A x$, and distances in $G \setminus \{f\}$ are shorter than distances in $G \setminus (A[u, x) \cup f)$.

- For every $f <_A x$ we have that $P'[a, x] \cdot A[x, u']$ is a path of length at most $(1+\varepsilon)\alpha$ to u' in $G \setminus A[u, f]$, and $b^f_{suf} \geq_P u'$ by definition.

So, from now on we assume that A is internally disjoint from P'.

Recall that $A = A_1 \cdot A_2$, and let z be the last vertex of A_1 such that b^z_{suf} is well defined (notice that z' is not necessarily on A_1). We show how to compute sequences F and B satisfying the required conditions for every $f \in A_1[u, z]$. If $z' = z$ (i.e. the last vertex with well defined b^f_{suf} on A is in A_1)- these F and B sequences conclude the lemma. Otherwise, z is the last vertex on A_1. In this case, we apply the same construction for $A_2[z, z']$, and the concatenation of the F and B sequences concludes the lemma. We describe the construction for A_1, the construction for A_2 is similar. Note that if $z = null$ then the lemma follows trivially. We can therefore assume that b^u_{suf} is well defined.

For every vertex $f \in A[u, z']$, let D_f be a shortest path from a to b^f_{suf} in $G \setminus A[u, f]$. Let r_f be the last vertex on D_f that is also on A_1 (note that r_f may be undefined). We denote as ℓ_f the first vertex on A_1 that is also on D_f. Notice that, by definition we have $\ell_f \leq_{D_f} r_f$ and $\ell_f \leq_{A_1} r_f$. Since both A_1 and D_f are shortest paths, we can assume $D_f[\ell_f, r_f] = A_1[\ell_f, r_f]$.

For two vertices $f_1 <_{A_1} f_2$, we say that f_1 crosses f_2, denoted as $f_1 \not\!\!-\, f_2$ if $\ell_{f_2} \in A_1[\ell_{f_1}, r_{f_1}]$. For two crossing vertices $f_1 <_{A_1} f_2$, we say that f_1 bad-crosses f_2 and denote $f_1 \overset{\text{bad}}{\not-} f_2$, if $\mathsf{dist}_{G \setminus A_1[u, f_2]}(a, b^{f_1}_{suf}) > (1 + \varepsilon)\alpha$.

Algorithm. We present the following algorithm (see Algorithm 1 below) that generates sequences $F = (u = f_1, f_2, \ldots, f_t = z)$ and $B = (b_1, b_2, \ldots, b_{t-1})$. The algorithm initializes the following variables.

1. A vertex f initially set to u: f is meant to iterate A_1 from left to right.
2. A vertex R initially set to r_u (or $null$ if r_u is undefined). The vertex R keeps track of the leftmost value of r_f that was encountered so far.
3. b' keeps track of the $b^{f_R}_{suf}$ value of the vertex f_R from which the value R was obtained (even if $R = null$). Initially, b' is set to b^u_{suf}.

Having initialized f,R, and b', the algorithm initializes the sequences as $F = (f)$ and $B = (b^f_{suf})$.

The algorithm runs the following procedure repeatedly until a terminating condition is met. The algorithm finds the vertex x which is the first vertex in $A_1[f, z]$ such that $\mathsf{dist}_{G \setminus A[u,x]}(a, b^f_{suf}) > (1 + \varepsilon)\alpha$. If there is no such x, the algorithm appends z to F and terminates.

Otherwise, the algorithm assigns $f \leftarrow x$, appends f to F and b^f_{suf} to B, and proceeds according to the following cases.

If $D_f \cap A_1 = \emptyset$, the algorithm appends z to F and terminates.

If $D_f \cap A_1 \neq \emptyset$ and $\ell_f >_{A_1} R$, the algorithm appends ℓ_f and z to F (in that order), appends b' to B and terminates. Finally, if $r_f <_{A_1} R$, the algorithm updates $R \leftarrow r_f$ and $b' \leftarrow b^f_{suf}$.

Correctness. Notice that the following invariant holds at any time during the execution of Algorithm 1:

Invariant 7.13. *At the beginning of every iteration (Line 3), for every $f \in F$ we have $R \leq_{A_1} r_f$.*[11]

Let $F = (f_1, f_2, \ldots, f_t)$ and $B = (b_1, b_2, \ldots, b_{t-1})$ be the output of the algorithm.

We make the following helpful claim.

Claim 7.14. *For every $i \in [1..t - 4]$, it holds that $f_i \overset{\text{bad}}{\not-} f_{i+1}$.*

[11]If r_f is undefined we consider z as r_f.

Algorithm 1: Partition A_1

```
    Input  : $A = A_1 \cdot A_2$, $G$, $u$, $u'$, $a$, $z$, $\alpha$, $\varepsilon$
    Output: Sequences $F$ and $B$
1   Initialize $f \leftarrow u$, $R \leftarrow r_f$, $b' \leftarrow b^f_{suf}$, $F \leftarrow (f)$, and $B \leftarrow (b^f_{suf})$;
2   while true do
3       Let $x$ be the first vertex in $A_1[f,z]$ such that $\mathsf{dist}_{G \setminus A[u,x]}(a, b^f_{suf}) > (1+\varepsilon)\alpha$;
4       if $x$ does not exist then
5           Append $z$ to $F$ and return $F, B$;
6       $f \leftarrow x$;
7       Append $f$ to $F$ and $b^f_{suf}$ to $B$;
8       if $D_f \cap A_1 = \emptyset$ then
9           Append $z$ to $F$ and return $F, B$;
10      else
11          if $\ell_f >_{A_1} R$ then
12              Append $\ell_f$, and $z$ to $F$, and $b'$ to $B$, and return $F, B$ ;
13          else
14              if $R >_{A_1} r_f$ then
15                  $R \leftarrow r_f$ and $b' \leftarrow b^f_{suf}$;
```

Proof. Since $i \le t-4$, the vertex f_{i+1} must have been added to F in line Line 7 and both $D_{f_{i+1}} \cap A_1 \neq \emptyset$ and $\ell_{f_{i+1}} \le_{A_1} R$ (otherwise, the algorithm terminates with $t \le i+3$). Due to Invariant 7.13, we have $R \le_{A_1} r_{f_i}$ and therefore $\ell_{f_{i+1}} \le_{A_1} r_{f_i}$. Since f_{i+1} is the first vertex in $A[f_i, z]$ such that $\mathsf{dist}_{G \setminus A[u, f_{i+1}]}(a, b^{f_i}_{suf})) > (1+\varepsilon)\alpha$, we must have $f_{i+1} \ge_{A_1} \ell_{f_i}$. Otherwise, D_{f_i} is a path from a to $b^{f_i}_{suf}$ in $G \setminus A[u, f_{i+1}]$ of length $\mathsf{len}(D_{f_i}) \le \alpha$. We have shown that $\ell_{f_{i+1}} \in A_1[\ell_{f_i}, r_{f_i}]$ and therefore $f_i \not\to f_{i+1}$. From the definition of f_{i+1} (Line 3) we have that $\mathsf{dist}_{G \setminus A[u, f_{i+1}]}(a, b^{f_i}_{suf}) > (1+\varepsilon)\alpha$ and therefore $f_i \overset{\text{bad}}{\not\to} f_{i+1}$ as required. ☐

We define a sequence of useful paths in G. For every $f_i \in F$ such that $\ell_{f_i} \in A_1$, we denote $S_i = A[u, \ell_{f_i}] \cdot D_{f_i}[\ell_{f_i}, b^{f_i}_{suf}]$. The paths S_i are instrumental in the proofs of the following claims.

Claim 7.15. *Let $i \in [1..t]$ such that $r_{f_i} \in A_1$ then, for any $f >_{A_1} r_{f_i}$ we have $\mathsf{dist}_{G \setminus \{f\}}(a, b^{f_i}_{suf}) \le \mathsf{len}(S_i) \le (1+\varepsilon)\alpha$.*

Proof. First note that by definitions of S_i and r_{f_i}, we have $A_1[r_{f_i}+1, z] \cap S_i = \emptyset$ (and recall that $A_1[\ell_{f_i}, r_{f_i}] = D_{f_i}[\ell_{f_i}, r_{f_i}]$). Therefore, $P'[a,u] \cdot S_i$ is a path in $G \setminus \{f\}$ for any $f >_A r_{f_i}$ and it follows that $\mathsf{dist}_{G \setminus \{f\}}(a, b^{f_i}_{suf}) \le \mathsf{len}(P') + \mathsf{len}(S_i) = \mathsf{len}(S_i)$.

It remains to prove $\mathsf{len}(S_i) \le (1+\varepsilon)\alpha$. We consider two cases regarding $b^{f_i}_{suf}$: (1) $b^{f_i}_{suf} = u'$, in this case since $D_{f_i}[\ell_{f_i}, u']$ is a shortest path, we have $\mathsf{len}(D_{f_i}[\ell_{f_i}, u']) \le \mathsf{len}(A[\ell_{f_i}, u'])$. Thus, $\mathsf{len}(S_i) = \mathsf{len}(A[u, \ell_{f_i}]) + \mathsf{len}(D_{f_i}[\ell_{f_i}, u']) \le \mathsf{len}(A[u, \ell_{f_i}]) + \mathsf{len}(A[\ell_{f_i}, u']) = \mathsf{len}(A) \le (1+\varepsilon)\alpha$.

The second case is (2) $b^{f_i}_{suf} >_P u'$. From definition of $b^{f_i}_{suf}$ we have that $\mathsf{dist}_{G \setminus A[u, f_i]}(a, u') > \alpha$. In particular, by definition of ℓ_{f_i} we have $\mathsf{len}(D_{f_i}[a, \ell_{f_i}] \cdot A[\ell_{f_i}, u']) > \alpha$. Moreover we have $\mathsf{len}(D_{f_i}[a, \ell_{f_i}] \cdot D_{f_i}[\ell_{f_i}, b^{f_i}_{suf}]) = \mathsf{len}(D_{f_i}) \le \alpha$ and $\mathsf{len}(A[u, \ell_{f_i}] \cdot A[\ell_{f_i}, u']) = \mathsf{len}(A) \le (1+\varepsilon)\alpha$. Combining the above we get $\mathsf{len}(S_i) = \mathsf{len}(A[u, \ell_{f_i}] \cdot D_{f_i}[\ell_{f_i}, b^{f_i}_{suf}]) \le \alpha + (1+\varepsilon)\alpha - \alpha = (1+\varepsilon)\alpha$. ☐

We prove the following claim by induction:

Claim 7.16. *For every $i \in [1, t-4]$ it holds that $\mathsf{len}(S_i) \leq (1-(i-2)\varepsilon)\alpha$.*

Proof. Notice that if $t < 5$ the claim is vacuously true. Notice that for $i \in [1, t-4]$ we have $b_i = b^{f_i}_{suf}$, ℓ_{f_i} and r_{f_i} are well defined and by Claim 7.14 we have $f_i \overset{\text{bad}}{\not\to} f_{i+1}$.

We prove the claim by induction on i. For $i = 1$ the claim $\mathsf{len}(S_1) \leq (1+\varepsilon)\alpha$ follows from Claim 7.15.

We assume the claim holds for i and prove for $i+1 \leq t-4$. By the algorithm (Line 3), $\mathsf{dist}_{G \setminus A[u, f_{i+1}]}(a, b_i) > (1+\varepsilon)\alpha$ and in particular, $\mathsf{len}(D_{f_{i+1}}[a, \ell_{f_{i+1}}] \cdot S_i[\ell_{f_{i+1}}, b_i]) > (1+\varepsilon)\alpha$ (note that $\ell_{f_{i+1}} \in A[\ell_{f_i}, r_{f_i}]$ since $f_i \overset{\text{bad}}{\not\to} f_{i+1}$). Moreover, $\mathsf{len}(S_i[u, \ell_{f_{i+1}}] \cdot S_i[\ell_{f_{i+1}}, b_i]) = \mathsf{len}(S_i) \leq (1-(i-2)\varepsilon)\alpha$ and $\mathsf{len}(D_{f_{i+1}}[a, \ell_{f_{i+1}}] \cdot D_{f_{i+1}}[\ell_{f_{i+1}}, b_{i+1}]) = \mathsf{len}(D_{f_{i+1}}) \leq \alpha$. Combining the above we get $\mathsf{len}(S_{i+1}) = \mathsf{len}(S_{i+1}[u, \ell_{f_{i+1}}] \cdot S_{i+1}[\ell_{f_{i+1}}, b_{i+1}]) \leq \alpha + (1-(i-2)\varepsilon)\alpha - (1+\varepsilon)\alpha = (1-(i-1)\varepsilon)\alpha$ as required. ☐

The following claim is a direct consequence of Claim 7.16.

Claim 7.17. $|F| = O(1/\varepsilon)$

Proof. Assume by contradiction that $t = |F| > 1/\varepsilon + 7$, and deduce by Claim 7.16 that S_i is a path of negative length, in contradiction. Thus, $|F| \leq 1/\varepsilon + 7$ as required. ☐

We prove the following claim.

Claim 7.18. *For every $f \in A_1$ such that $f_i \leq_{A_1} f <_{A_1} f_{i+1}$, we have:*

1. $b^f_{suf} \geq_P b_i$.
2. $\mathsf{dist}_{G \setminus \{f\}}(a, b_i) \leq (1+\varepsilon)\alpha$.

Proof. We consider several cases regarding b_i.

Case 1: $b_i = b^{f_i}_{suf}$. In this case by Observation 7.11 since $f_i \leq_{A_1} f$ we have $b^f_{suf} \geq_P b^{f_i}_{suf} = b_i$. We consider two sub-cases regarding f_{i+1}:

- If f_{i+1} was added in Lines 5, 7 or 9. In each of these lines, f_{i+1} is set to be the first x in $A_1[f_i, z]$ such that $\mathsf{dist}_{G \setminus A[u,x]}(a, b_i) > (1+\varepsilon)\alpha$ or is set to z if there is no such x. Either way, due to $f <_{A_1} f_{i+1}$ we have $\mathsf{dist}_{G \setminus \{f\}}(a, b_i) \leq \mathsf{dist}_{G \setminus A[u,f]}(a, b_i) \leq (1+\varepsilon)\alpha$ as required.

- If $f_{i+1} = \ell_{f_i}$ was added in Line 12, then $\mathsf{dist}_{G \setminus \{f\}}(a, b_i) \leq \mathsf{dist}_{G \setminus A_1[u, \ell_{f_i}-1]}(a, b_i) \overset{(*)}{\leq} \mathsf{len}(D_{f_i}) \leq \alpha \leq (1+\varepsilon)\alpha$, where the inequality (*) follows from $D_{f_i} \cap A_1[u, \ell_{f_i} - 1] = \emptyset$ by defintion of ℓ_{f_i}.

Case 2: b_i was added in Line 12. In this case $f_i = \ell_{f_{i-1}}$ and $f_{i+1} = z$. In addition $b_i = b'$ at this time of the algorithm. Consider the values of R and b' in the iteration of the algorithm where Line 12 was executed. Notice that $b_i = b' = b^{f_j}_{suf}$ and $R = r_{f_j}$ for some $j < i$ (as assigned in either Line 1 or Line 15) such that $r_{f_j} \in A_1$. It follows from Observation 7.11 that $b^f_{suf} \geq_P b^{f_i}_{suf} \geq_P b^{f_j}_{suf} = b_i$ as required. The algorithm executes Line 12 since $f \geq_{A_1} \ell_{f_{i-1}} >_{A_1} R = r_{f_j}$. Finally, by Claim 7.15 we have $\mathsf{dist}_{G \setminus \{f\}}(a, b^{f_j}_{suf}) \leq \mathsf{len}(S_j) \leq (1+\varepsilon)\alpha$. ☐

Combining Claims 7.17 and 7.18, concludes the proof of Lemma 7.12. ☐

We are now ready to present the labeling scheme for $\mathcal{L}^{\mathsf{a} \rightsquigarrow \mathsf{P}}$, proving Lemma 7.3.

Lemma 7.3. *For a graph G, two paths P and P' of length 0, a path $A = (u \rightsquigarrow u')$ from the last vertex of P' to u' which is the first vertex of P and numbers $\alpha, \varepsilon \in \mathbb{R}^+$ such that:*

1. $\mathsf{len}(A) \leq (1+\varepsilon)\alpha$.
2. $A = A_1 \cdot A_2$ *such that* A_1 *and* A_2 *are shortest paths in* G.

There exists a labeling scheme $\mathcal{L}^{\mathsf{a \rightsquigarrow P}} = \mathcal{L}^{\mathsf{a \rightsquigarrow P}}_{G,P',A,P,\alpha,\varepsilon}(v)$ such that given the labels of two vertices $a \in P'$ and $f \in A \setminus P'$, one can obtain a vertex $b \in P$ such that $\mathsf{dist}_{G\setminus\{f\}}(a,b) \leq (1+\varepsilon)\alpha$ and $b \leq_P \mathsf{first}^{\alpha}_{\mathsf{G \setminus A[...f]}}(a,P)$ or conclude that $\mathsf{first}^{\alpha}_{\mathsf{G \setminus A[...f]}}(a,P) = null$. The size of each label is $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$.

Proof. For every vertex $v \in P'$: Let z'_v be the last vertex of A such that $b^{z'}_{suf}$ is well defined. If $z' = null$, the label of v stores a flag. Otherwise, v stores the sequences F and B obtained by applying Lemma 7.12 on v and $b^{z'}_{suf}$. In addition for every vertex $v \in A$: the label of v stores v's index in A. From Lemma 7.12, it is clear that the size of the label is $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$.

Given the labels of $a \in P'$ and $f \in A$, one can obtain a vertex $b \in P$ such that $\mathsf{dist}_{G\setminus\{f\}}(a,b) \leq (1+\varepsilon)\alpha$ and $b \leq_P \mathsf{first}^{\alpha}_{\mathsf{G \setminus A[...f]}}(a,P)$ or conclude that $\mathsf{first}^{\alpha}_{\mathsf{G \setminus A[...f]}}(a,P) = null$, as follows. If $z'_a = null$ (marked in the label of a with a flag) or $f >_A z'_a$ ($z'_a = F[|F|]$), then we conclude that $\mathsf{first}^{\alpha}_{\mathsf{G \setminus A[...f]}}(a,P) = null$. If $f = z'_a$, we simply return $b^{z'_a}_{suf}$. Otherwise, let i be the index such that $f_i \leq_A f <_A f_{i+1}$, return b_i, which by Lemma 7.12 satisfies the requirements. □

8 The $\mathcal{L}^{\mathsf{P' \to P \setminus f}}$ Labeling (Proof of Lemma 5.9)

In this section we prove Lemma 5.9. The settings in this section are as follows. G (for the ease of presentation we use here G as the name of the graph) is a graph, P' is a 0-length path, P is a path without outgoing edges which lies on a single face, and $\alpha, \varepsilon \in \mathbb{R}^+$. For $f \in P$ let P_1 and P_2 be the prefix and suffix of P before and after f (without f), respectively. Our goal is to develop a labeling scheme such that given the labels of vertices $a \in P'$ and $f \in P$, one can retrieve two vertices b_1 and b_2 which are $\varepsilon\alpha$-$\mathsf{first}^{\alpha}_{\mathsf{G\setminus\{f\}}}(a, P_1)$ and $\varepsilon\alpha$-$\mathsf{first}^{\alpha}_{\mathsf{G\setminus\{f\}}}(a, P_2)$, respectively. The size of each label is required to be $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$.

Let z be the first vertex of P'. Let $\Gamma = \{i\varepsilon\alpha \mid i \in [0, \lceil 1/\varepsilon \rceil]\}$ be the set of multiples of $\varepsilon\alpha$. Let F be the graph obtained from G by removing all edges of P. For every $\beta \in \Gamma$ let $P_\beta = \{v \in P \mid \beta - \varepsilon\alpha < \mathsf{dist}_F(z,v) \leq \beta\}$ denote the set of vertices of P whose distance from z in F (which is exactly the distance in G via paths that are **internally disjoint** from P) is in the interval $(\beta - \varepsilon\alpha, \beta]$. Notice that since P lies on a single face, all paths entering P enter from the same side.

For every $a \in P'$ let $P_\beta(a) = \{v \in P_\beta \mid \mathsf{dist}_F(a,v) \leq \alpha\}$ denote the subset of vertices of P_β whose distance from a in F is at most α. By definition $P_\beta(a) \subseteq P_\beta$ and planarity dictates the following:

Claim 8.1. *For any $\beta \in \Gamma$, considering the sets $P_\beta, P_\beta(a)$ as sequences ordered according to the order along P. There is a set of at most two intervals of consecutive vertices of P_β such that: (i) every vertex in $P_\beta(a)$ is in one of these two intervals, and (ii) for every vertex $w \in P_\beta$ in each of these intervals, $\mathsf{dist}_F(a,w) \leq (1+\varepsilon)\alpha$. Finally, (iii) the endpoints of each interval are in P_β.*

Proof. Recall that P lies on a single face of G. Let P° be the cycle of the boundary of the face P lies on. We shall show that with respect to the cyclic order on P°, a single interval in the statement of the claim suffices. This interval consists of at most two intervals of P.

Consider all pairs of consecutive vertices (in the cyclic order along $P_\beta(a)$) $u, v \in P_\beta(a)$. Let C_{uv} be the (undirected) cycle formed by the a-to-u shortest path in F, the a-to-v shortest path in F, and $P^\circ[u,v]$. Since the union of the shortest paths forming the above cycles is a tree (a subtrtee of the shortest path tree rooted at a) whose leaves are all on P, the vertex z is strictly enclosed by at most one of these cycles. Let $C_{u^*v^*}$ be the cycle that strictly encloses z. We choose the interval to start at v^* and end at u^* (i.e., the interval containing all vertices of P_β except those in $P(u^*, v^*)$). By definition, the interval contains all vertices of $P_\beta(a)$, so property (i) is satisfied. To show property (ii), consider a vertex $w \in P_\beta$ in this interval. By choice of the interval, z is enclosed by $C_{u^*v^*}$ and w is not strictly enclosed by $C_{u^*v^*}$. Hence, the shortest path from z to w must intersect the a-to-u^* path or the a-to-v^* path. Suppose without loss of generality that the intersection is with the a-to-u^* path (the other case is identical with v^* taking the role of u^*), and let x be an intersection vertex. If $w \in P_\beta(a)$ then property (ii) is satisfied by definition of $P_\beta(a)$. Otherwise, suppose for the sake of contradiction that property (ii) is not satisfied. That is, $\mathsf{dist}_F(a, w) > (1 + \varepsilon)\alpha$. In particular, the sum of lengths of a-to-x prefix of the a-to-u^* path and the x-to-w suffix of the z-to-w path is at least $(1 + \varepsilon)\alpha$. But since the sum of lengths of the a-to-u^* path and of the z-to-w path is less than $\beta + \alpha$, we get that the sum of the length of the z-to-x prefix of the z-to-w path and the x-to-u^* suffix of the a-to-u^* path must be less than $\beta - \varepsilon\alpha$, a contradiction to $u \in P_\beta$. ☐

We are now ready to prove Lemma 5.9.

Lemma 5.9. *There exists a labeling scheme* $\mathcal{L}^{\mathsf{P'\to P\backslash f}} = \mathcal{L}^{\mathsf{P'\to P\backslash f}}_{H,P,P',\alpha,\varepsilon}$ *where H is a planar graphs H with a 0-length path P' and a path P without outgoing edges which lies on a single face, such that $P \cap P' = \emptyset$, and $\alpha, \varepsilon \in \mathbb{R}^+$. For $f \in P$ let P_1 and P_2 be the prefix and suffix of P before and after f (without f), respectively. Given the labels of two vertices $a \in P'$ and $f \in P$, one can retrieve two vertices b_1 and b_2 which are $\varepsilon\alpha$-$\mathsf{first}^{\alpha}_{\mathsf{H\backslash\{f\}}}(a, P_1)$ and $\varepsilon\alpha$-$\mathsf{first}^{\alpha}_{\mathsf{H\backslash\{f\}}}(a, P_2)$, respectively. The size of each label is $\tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$.*

Proof. For every $a \in P'$ the label of a stores: $\mathsf{first}^{\alpha}_{\mathsf{G}}(a, P)$ and for every $\beta \in \Gamma$, the set of (at most two) intervals obtained by Claim 8.1. For every $f \in P$ the label of f stores: for every $\beta \in \Gamma$, the successor of f in P_β. (Every vertex of P is stored with its index on P.)

Size. It is clear that the size of each label is $O(|\Gamma|) = \tilde{O}(\mathsf{poly}(\frac{1}{\varepsilon}))$.

Decoding. Given the labels of $a \in P'$ and $f \in P$, we obtain b_1 and b_2 as follows. First, if $\mathsf{first}^{\alpha}_{\mathsf{G}}(a, P) <_P f$ then $b_1 = \mathsf{first}^{\alpha}_{\mathsf{G}}(a, P)$, otherwise $b_1 = \textit{null}$. To compute b_2, we iterate over all $\beta \in \Gamma$. If f is in one of the intervals obtained by Claim 8.1, then b_2^β is the succesor of f in P_β. Otherwise, b_2^β is the first vertex on P_2 which is an endpoint of an interval, or *null* if there is no such endpoint. Finally, we set $b_2 = \min_{\leq_P}\{b_2^\beta \mid \beta \in \Gamma\}$.

Correctness. It is straightforward that if $\mathsf{first}^{\alpha}_{\mathsf{G}}(a, P) \geq_P f$ then $\mathsf{first}^{\alpha}_{\mathsf{G\backslash\{f\}}}(a, P_1) = \textit{null}$ and therefore $b_1 = \textit{null}$ is a valid answer. If $\mathsf{first}^{\alpha}_{\mathsf{G}}(a, P) <_P f$ then clearly since P does not have outgoing edges $\mathsf{first}^{\alpha}_{\mathsf{G}}(a, P) = \mathsf{first}^{\alpha}_{\mathsf{G\backslash\{f\}}}(a, P_1)$ is an $\varepsilon\alpha$-$\mathsf{first}^{\alpha}_{\mathsf{G\backslash\{f\}}}(a, P_1)$.

For every $\beta \in \Gamma$, in all cases b_2^β is in one of the intervals obtained by Claim 8.1 and so $b_2^\beta \in P_\beta$. Thus, by Claim 8.1 (ii), $\mathsf{dist}_G(a, b_2^\beta) \leq \mathsf{dist}_F(a, b_2^\beta) \leq (1 + \varepsilon)\alpha$.

Let $b^* = \mathsf{first}^{\alpha}_{\mathsf{G\backslash\{f\}}}(a, P_2)$ and let $\beta = q \in \Gamma \mid q \geq \mathsf{dist}_F(z, b^*)\}$. It remains to prove $b_2 \leq_P b^*$. Notice that $b^* \in P_\beta$. Moreover $b^* \in P_\beta(a) \subseteq P_\beta$. Let $I = P[u, v]$ be the endpoints of the interval obtained by Claim 8.1 containing b^* (such an interval exists by (i)). If $f \in P[u, v]$ then b_2^β is the successor w of f in P_β which implies $w \leq_P b^*$. If $f <_P u$, then $b_2^\beta \leq_P u \leq_P b^*$. To conclude, $b_2^\beta \leq_P b^*$ and by the minimality of b_2 among all values of β we have $b_2 \leq_P b^*$. ☐

References

[ACC$^+$96] Srinivasa Rao Arikati, Danny Z. Chen, L. Paul Chew, Gautam Das, Michiel H. M. Smid, and Christos D. Zaroliagis. Planar spanners and approximate shortest path queries among obstacles in the plane. In *4th ESA*, volume 1136, pages 514–528, 1996.

[ACG12] Ittai Abraham, Shiri Chechik, and Cyril Gavoille. Fully dynamic approximate distance oracles for planar graphs via forbidden-set distance labels. In *44th STOC*, pages 1199–1218, 2012.

[ACGP16] Ittai Abraham, Shiri Chechik, Cyril Gavoille, and David Peleg. Forbidden-set distance labels for graphs of bounded doubling dimension. *ACM Trans. Algorithms*, 12(2):22:1–22:17, 2016.

[ADK17] Stephen Alstrup, Søren Dahlgaard, and Mathias Bæk Tejs Knudsen. Optimal induced universal graphs and adjacency labeling for trees. *J. ACM*, 64(4):27:1–27:22, 2017.

[AKTZ19] Stephen Alstrup, Haim Kaplan, Mikkel Thorup, and Uri Zwick. Adjacency labeling schemes and induced-universal graphs. *SIAM J. Discret. Math.*, 33(1):116–137, 2019.

[AN17] Noga Alon and Rajko Nenadov. Optimal induced universal graphs for bounded-degree graphs. In *28th SODA*, pages 1149–1157, 2017.

[BCG$^+$22] Aviv Bar-Natan, Panagiotis Charalampopoulos, Pawel Gawrychowski, Shay Mozes, and Oren Weimann. Fault-tolerant distance labeling for planar graphs. *Theor. Comput. Sci.*, 918:48–59, 2022.

[BCR18] Surender Baswana, Keerti Choudhary, and Liam Roditty. Fault-tolerant subgraph for single-source reachability: General and optimal. *SIAM J. Comput.*, 47(1):80–95, 2018.

[BGL07] Nicolas Bonichon, Cyril Gavoille, and Arnaud Labourel. Short labels by traversal and jumping. *Electronic Notes in Discrete Mathematics*, 28:153–160, 2007.

[BLM12] Surender Baswana, Utkarsh Lath, and Anuradha S. Mehta. Single source distance oracle for planar digraphs avoiding a failed node or link. In *23rd SODA*, pages 223–232, 2012.

[Cab12] Sergio Cabello. Many distances in planar graphs. *Algorithmica*, 62(1-2):361–381, 2012.

[CDW17] Vincent Cohen-Addad, Søren Dahlgaard, and Christian Wulff-Nilsen. Fast and compact exact distance oracle for planar graphs. In *58th FOCS*, pages 962–973, 2017.

[CGK09] Bruno Courcelle, Cyril Gavoille, and Mamadou M. Kanté. Compact labelings for efficient first-order model-checking. *Journal of Combinatorial Optimization*, 21(1):19–46, 2009.

[CGL$^+$23] Panagiotis Charalampopoulos, Pawel Gawrychowski, Yaowei Long, Shay Mozes, Seth Pettie, Oren Weimann, and Christian Wulff-Nilsen. Almost optimal exact distance oracles for planar graphs. *J. ACM*, 70(2):12:1–12:50, 2023.

[CGMW19] Panagiotis Charalampopoulos, Pawel Gawrychowski, Shay Mozes, and Oren Weimann. Almost optimal distance oracles for planar graphs. In *51st STOC*, pages 138–151, 2019.

[Cho16] Keerti Choudhary. An optimal dual fault tolerant reachability oracle. In *43rd ICALP*, pages 130:1–130:13, 2016.

[CMT19] Panagiotis Charalampopoulos, Shay Mozes, and Benjamin Tebeka. Exact distance oracles for planar graphs with failing vertices. In *30th SODA*, pages 2110–2123, 2019.

[CT10] Bruno Courcelle and Andrew Twigg. Constrained-path labellings on graphs of bounded clique-width. *Theory of Computing Systems*, 47(2):531–567, 2010.

[CX00] Danny Z. Chen and Jinhui Xu. Shortest path queries in planar graphs. In *32nd STOC*, pages 469–478, 2000.

[Dji96] Hristo Djidjev. Efficient algorithms for shortest path queries in planar digraphs. In *22nd WG*, volume 1197, pages 151–165, 1996.

[ES35] P. Erdös and G. Szekeres. A combinatorial problem in geometry. *Compositio Mathematica*, 2:463–470, 1935.

[FKMS07] Joan Feigenbaum, David R. Karger, Vahab S. Mirrokni, and Rahul Sami. Subjective-cost policy routing. *Theor. Comput. Sci.*, 378(2):175–189, 2007.

[FR06] Jittat Fakcharoenphol and Satish Rao. Planar graphs, negative weight edges, shortest paths, and near linear time. *J. Comput. Syst. Sci.*, 72(5):868–889, 2006.

[GGI$^+$17] Loukas Georgiadis, Daniel Graf, Giuseppe F. Italiano, Nikos Parotsidis, and Przemyslaw Uznanski. All-pairs 2-reachability in $O(n^\omega \log n)$ time. In *44th ICALP*, pages 74:1–74:14, 2017.

[GIP17] Loukas Georgiadis, Giuseppe F. Italiano, and Nikos Parotsidis. Strong connectivity in directed graphs under failures, with applications. In *28th SODA*, pages 1880–1899, 2017.

[GKK$^+$01] Cyril Gavoille, Michal Katz, Nir A. Katz, Christophe Paul, and David Peleg. Approximate distance labeling schemes. In *9th ESA*, pages 476–487, 2001.

[GKR01] Anupam Gupta, Amit Kumar, and Rajeev Rastogi. Traveling with a pez dispenser (or, routing issues in MPLS). In *42nd FOCS*, pages 148–157, 2001.

[GMWWN18] Pawel Gawrychowski, Shay Mozes, Oren Weimann, and Christian Wulff-Nilsen. Better tradeoffs for exact distance oracles in planar graphs. In *SODA*, 2018.

[GPPR04] Cyril Gavoille, David Peleg, Stéphane Pérennes, and Ran Raz. Distance labeling in graphs. *Journal of Algorithms*, 53(1):85–112, 2004.

[GU23] Pawel Gawrychowski and Przemyslaw Uznanski. Better distance labeling for unweighted planar graphs. *Algorithmica*, 85(6):1805–1823, 2023.

[HL09] Tai-Hsin Hsu and Hsueh-I Lu. An optimal labeling for node connectivity. In *20th ISAAC*, 2009.

[HLNW17] Monika Henzinger, Andrea Lincoln, Stefan Neumann, and Virginia Vassilevska Williams. Conditional hardness for sensitivity problems. In *8th ITCS*, pages 26:1–26:31, 2017.

[HRT15] Jacob Holm, Eva Rotenberg, and Mikkel Thorup. Planar reachability in linear space and constant time. In *56th FOCS*, pages 370–389, 2015.

[IKP21] Giuseppe F. Italiano, Adam Karczmarz, and Nikos Parotsidis. Planar reachability under single vertex or edge failures. In *32nd SODA*, pages 2739–2758, 2021.

[KKKP04] Michal Katz, Nir A. Katz, Amos Korman, and David Peleg. Labeling schemes for flow and connectivity. *SIAM J. Comput.*, 34(1):23–40, 2004.

[Kle05] Philip N Klein. Multiple-source shortest paths in planar graphs. In *SODA*, pages 146–155, 2005.

[KM] P.N. Klein and S. Mozes. Optimization algorithms for planar graphs. http://planarity.org. Book draft.

[KNR92] Sampath Kannan, Moni Naor, and Steven Rudich. Implicit representation of graphs. *SIAM Journal on Discrete Mathematics*, 5(4):596–603, 1992.

[Kor10] Amos Korman. Labeling schemes for vertex connectivity. *ACM Trans. Algorithms*, 6(2):39:1–39:10, 2010.

[KS99] Valerie King and Garry Sagert. A fully dynamic algorithm for maintaining the transitive closure. In *31st STOC*, pages 492–498, 1999.

[LT79] Richard J. Lipton and Robert Endre Tarjan. A separator theorem for planar graphs. *SIAM J. Appl. Math.*, 36(2):177–189, 1979.

[MS12] Shay Mozes and Christian Sommer. Exact distance oracles for planar graphs. In Yuval Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 209–222. SIAM, 2012.

[Nus11] Yahav Nussbaum. Improved distance queries in planar graphs. In Frank Dehne, John Iacono, and Jörg-Rüdiger Sack, editors, *Algorithms and Data Structures - 12th International Symposium, WADS*, volume 6844 of *Lecture Notes in Computer Science*, pages 642–653. Springer, 2011.

[Pel05] David Peleg. Informative labeling schemes for graphs. *Theor. Comput. Sci.*, 340(3):577–593, 2005.

[PRSWN16] Casper Petersen, Noy Rotbart, Jakob Grue Simonsen, and Christian Wulff-Nilsen. Near-optimal adjacency labeling scheme for power-law graphs. In *43rd ICALP*, pages 133:1–133:15, 2016.

[Rot16] Noy Galil Rotbart. *New Ideas on Labeling Schemes.* PhD thesis, University of Copenhagen, 2016.

[Tho04] Mikkel Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *J. ACM*, 51(6):993–1024, 2004.

[Twi06] Andrew D. Twigg. Compact forbidden-set routing. Technical Report UCAM-CL-TR-678, University of Cambridge, Computer Laboratory, December 2006.

[vdBS19] Jan van den Brand and Thatchaphol Saranurak. Sensitive distance and reachability oracles for large batch updates. In *60th FOCS*, pages 424–435, 2019.

[WN10] Christian Wulff-Nilsen. *Algorithms for planar graphs and graphs in metric spaces*. PhD thesis, University of Copenhagen, 2010.