# On Dynamic Approximate Shortest Paths for Planar Graphs with Worst-Case Costs

Ittai Abraham * Shiri Chechik † Daniel Delling‡ Andrew V. Goldberg§
Renato F. Werneck¶

**Abstract**

Given a base weighted planar graph $G_{input}$ on $n$ nodes and parameters $M, \epsilon$ we present a dynamic distance oracle with $1+\epsilon$ stretch and worst case update and query costs of $\epsilon^{-3}M^4 \cdot \text{poly-log}(n)$. We allow arbitrary edge weight updates as long as the shortest path metric induced by the updated graph has stretch of at most $M$ relative to the shortest path metric of the base graph $G_{input}$.

For example, on a planar road network, we can support fast queries and dynamic traffic updates as long as the shortest path from any source to any target (including using arbitrary detours) is between, say, 80 and 3 miles-per-hour.

As a warm-up we also prove that graphs of bounded treewidth have exact distance oracles in the dynamic edge model.

To the best of our knowledge, this is the first dynamic distance oracle for a non-trivial family of dynamic changes to planar graphs with worst case costs of $o(n^{1/2})$ both for query and for update operations.

## 1 Introduction

Computing shortest paths is a fundamental optimization problem that received a lot of attention recently due to proliferation of GPS navigation and location-based services. The ubiquity of sensors and mobile devices and emergent crowdsourcing applications motivate dynamic algorithms that can efficiently handle edge weight changes. In recent years the algorithm engineering community has been developing an ever improving stream of speedup techniques for computing shortest paths in road networks (see [15] for a survey). Currently, the fastest exact distance oracles implementations run in microseconds or less on road networks with tens of millions of vertices [2, 3, 4, 5]. This is achieved by preprocessing the network for a few minutes (or hours) to generate a moderate amount of auxiliary data that speeds up queries. Moreover, several of these techniques can also handle dynamic changes of edge weights or vertex addition/removal [7, 14, 16, 19, 26, 32]. Some of the fastest techniques for dynamic changes [7, 14] are based on the multi-level partition approach [21, 27, 28] and can process an edge weight update in a road network with tens of millions of vertices in milliseconds.

The theory community has also studied dynamic distance oracles aiming to derive bounds for general graphs. Demetrescu and Italiano [17] devised a fully dynamic exact distance oracle with amortized update time $\tilde{O}(n^2)$. Thorup [30] extended the result to negative edge weights and slightly improved the update time. Thorup [31] also presented an algorithm with worst-case update time $\tilde{O}(n^{2.75})$. Baswana et al. [6] considered the decremental case for unweighted directed graphs and obtained amortized update time $\tilde{O}(n^3/m)$.

For approximate queries, Roditty and Zwick [25] devised distance oracles for the incremental only and for the decremental only cases, both with $(1+\epsilon)$ stretch, constant query time and amortized update time $\tilde{O}(n)$. Bernstein [8] considered the fully dynamic case and presented an algorithm for weighted, undirected graphs with $O(\log\log\log n)$ query time and update time close to

*VMware Research, iabraham@vmware.com
†Tel Aviv University, shiri.chechik@gmail.com. This research was supported by the ISRAEL SCIENCE FOUNDATION (grant No. 1528/15).
‡daniel.delling@gmail.com
§andvgol@gmail.com
¶rwerneck@acm.org

$\tilde{O}(m)$ and $(2+\epsilon)$ stretch.

Better results are known for special graph classes, in particular for planar graphs. Klein and Subramanian [23] construct a $(1+\epsilon)$-distance oracle for planar graphs with worst-case query time of $\tilde{O}(n^{2/3})$ and amortized update time of $\tilde{O}(n^{2/3})$. This has been recently improved [1] to worst-case query and update time of $\tilde{O}(n^{1/2})$.

In recent years there has been much progress in obtaining linear and near-linear time algorithms for planar graphs, for example see [18, 22, 11, 10]. However, for dynamic distance oracles on planar graphs there are no schemes that obtain $o(n^{1/2})$ for both update and query. In this paper we identify a restricted family of dynamic changes which allow us to break this $O(n^{1/2})$ barrier.

**Our results.** As a warm-up, out first result is for graphs of bounded treewidth with dynamic arbitrary non-negative edge weight changes.

THEOREM 1.1. *For an undirected graph of treewidth $k$ and positive weights, there exists a data structure of size $O(k^2 n)$ that supports edge weight changes and exact distance queries with worst-case cost $O(k^3 \log n)$ and $O(k^2 \log n \log(k \log n))$ respectively.*

All of our sizes are measured in words, where a word is a space unit big enough to contain any distance or the identifier of a vertex. Empirical analogs of this result were suggested by [7, 20] for graphs with a certain small cut property. Their algorithms are similar, and could implicitly lead to comparable bounds. We provide the first proof in the edge dynamic setting for the family of graphs of bounded treewidth. To the best of our knowledge, the best fully dynamic distance oracles for constant treewidth have query and update cost of $\tilde{O}(n^{1/2})$ (using the results of [12, 13] combined with [1]). Our result is exponentially better, but allows only edge weight updates. In many practical scenarios (such as handling traffic changes) edge weight updates are sufficient.

Our main result is for planar graphs with a parameter $M$ which allows dynamic edge weights as long as the shortest path metric induced by the new graph has stretch of at most $M$ relative to some fixed base weighted graph. Formally, given a base graph $G_{input}$ we require that after each edge weight update, the new weights for graph $G$ obey:

$$\forall u, v \in V(G): \quad d_{G_{input}}(u,v) \leq d_G(u,v) \leq M \cdot d_{G_{input}}(u,v)$$

We believe that the assumption that edge weights maintain a global bound of $M$ is quite reasonable in many practical scenarios. For example, a reasonable way to model changes in traffic for road networks is to assume that for any road segment, there is always a way to bypass it and obtain a weight that varies between say 100 mps to 1 mps (bounded range). In such a case we can set $M = 100$. Our scheme also has a parameter $\epsilon > 0$ and each query provides a $1+\epsilon$ stretch approximation to the shortest path.

THEOREM 1.2. *Given a base weighted planar graph $G_{input}$ on $n$ vertices and parameters $M, \epsilon$, there exists a data structure of size $\tilde{O}(Mn/\epsilon)$ that supports edge weight changes as long as the shortest path metric of the new graph has a stretch of at most $M$ relative to $G_{input}$. The scheme supports $1+\epsilon$ stretch distance queries and edge weight changes with worst-case operation cost of $\tilde{O}(\epsilon^{-3} M^4)$.*

In particular, setting $M = \tilde{O}(1)$ and $\epsilon = 1/\tilde{O}(1)$ gives a scheme where the worst case costs per operation are bounded by $\tilde{O}(1)$. To the best of our knowledge this is the first $o(n^{1/2})$ cost scheme for any non-trivial family of planar graph dynamic updates. Our scheme is useful when the graph weight dynamism always maintains bounded stretch relative to the initial graph. We believe this restriction captures interesting real-world workloads. For arbitrary edge weight updates the bound of $\tilde{O}(n^{1/2})$ of [1] is better.

Theorem 1.2 relies on a variant of the shortest path separator tree of [1] detailed in Section 3. We then proceed with a non-trivial extension of the approach we use for Theorem 1.1. In particular, obtaining a stretch of $1+\epsilon$ requires to delicately control the loss of stretch that is accumulated over multiple phases. This is captured in Property 4.1 (in particular part (b) requires stretch that depends on parameters in a not obvious manner). The update algorithm and the proof that Property 4.1 is maintained in dynamic changes requires a subtle double inductive argument (see Lemma 4.6).

**Notation.** Let $G = (V, E, \omega)$ be a connected weighted undirected graph where $\omega : E \to \mathbb{R}^+$, $|V| = n$ and $|E| = m$. Let $\mathbf{dist}(s,t,G)$ be the length of a shortest path from $s$ to $t$ in the graph $G$. When $G$ is clear from the context we simply write $\mathbf{dist}(s,t)$. Given a path $P$, we denote by $|P|$ the number of edges on the path and by $\omega(P)$ be the sum of the weights of its edges. Given two paths $P_1$ and $P_2$ that share a common endpoint, let $P_1 \circ P_2$ be the concatenation of the two paths.

## 2 Edge Dynamic Distance Oracle for Bounded Treewidth Graphs

In this section we present a distance oracle for graphs of bounded treewidth that can handle dynamic edge weight changes and prove Theorem 1.1. Roughly speaking,

we maintain a binary tree decomposition of logarithmic depth. For any two vertices in a bag $x$, we store the distance $d_x$ between them in a subgraph of $G$ induced by some subset of the nodes. The query operation consists of building a concise upward search graph from $s$ and from $t$. The update operation of an edge works by updating the data that is stored in each bag in a bottom-up fashion.

A tree decomposition of a graph $G = (V, E)$ is a pair $(X, T)$ where $T$ is a rooted tree on $X$ with root $r \in X$. Each $x \in X$ (called a *bag*) is a subset of $V$ with the following properties:

1. for each $(u, v) \in E$ there exists $x \in X$ such that $u, v \in x$.
2. for each $v \in V$ the set $\{x \in X \mid v \in x\}$ is a subtree of $T$.

The width of a tree decomposition is $\max_{x \in X} |x| - 1$. Let $c(x)$ be the set of children bags of $x$ in $T$. We say that $y \in X$ is a descendant of $x \in X \setminus \{y\}$ if $x$ belongs to the path from $y$ to the root of $T$. Let $d(x)$ be the depth of $x$ in $T$. The depth of a decomposition is $\max_{x \in X} d(x)$. A graph $G$ has treewidth $k$ if $k$ is the least integer such that $G$ admits a tree decomposition of width $k$.

We use the following well-know result ("rake and compress"):

LEMMA 2.1. *[9] If $G$ has treewidth $k$ then $G$ has a binary tree decomposition with width $3k + 2$ and depth $O(\log n)$.*

Given $x \in X$, let $V(x) = \{v \in V \mid \exists y \in X, v \in y, y$ is a descendant of $x$ or $x$ itself$\}$, and let $T(x) = \{y \in X \mid y$ is a descendant of $x$ or $x$ itself$\}$. Note that $V(x) = \cup_{y \in T(x)} y$.

For any $U \subseteq V$ let $G[U]$ be the subgraph of $G$ induced by $U$. The following is a crucial definition. Let $G^-[V(x)]$ be the induced subgraph of $V(x)$, where we include only edges $(u, v)$ such that $u, v \in V(x)$ and there is no bag $x'$ such that $u, v \in x'$ and $x$ is a descendant of $x'$. This definition maps each edge $e$ to at most $O(\log n)$ bags $x$ for which $e \in G^-[V(x)]$ (essentially $e = (u, v) \notin G^-[V(x)]$ if $u, v \in y$ and $y$ is an ancestor of $x$). This is what allows for the fast update operation. We will often consider the metric $\mathbf{dist}(\star, \star, G^-[V(x)])$.

Given a vertex $v$, let $x_v \in X$ be the closest bag to the root such that $v \in x_v$. Given an edge $e = (u, v)$ let $x_e \in X$ be the closest bag to the root such that both $u, v \in x_e$.

**2.1 Data Structure** Using Lemma 2.1, let $(X, T)$ be a tree decomposition with width $3k+2$ and depth $O(\log n)$. For each $x \in X$ and $u, v \in x$ store $d_x(u, v) := \mathbf{dist}(u, v, G^-[V(x)])$. In addition, for every vertex $v \in V$ store a pointer to the bag $x_v$. Similarly, for every edge $e \in E$ store a pointer to the bag $x_e$.

LEMMA 2.2. *The size of the data structure is $O(k^2 n)$.*

*Proof.* There are $n$ vertices and $O(kn)$ edges in $G$. Thus, storing $x_v$ and $x_e$ for every $v \in V$ and $e \in E$ requires $O(kn)$ space. In addition, there are $O(n)$ bags, and each bag $x$ needs $O(|x|^2)$ space. Since $|x| \leq 3k+2$, the lemma follows.

Just as in the separator-based multi-level graph overlay approach [20], we store for every bag/region $x \in X$ a clique of distances between the boundary vertices. Observe that this clique is a concise way to store all the shortest path that are fully contained in $G^-[V(x)]$. Therefore, the query simply joins all the relevant cliques and runs a shortest path algorithm in the induced graph.

**2.2 Query** Let $X(s, t)$ be the set of all $x \in X$ such that either $x_s$ or $x_t$ is a descendant of $x$. Let $G(s, t)$ be the weighted graph whose vertices are all $v \in V$ such that there exists $x \in X(s, t)$ and $v \in x$. The edges of $G(s, t)$ are all pairs $(u, v)$ such that there exists $x \in X(s, t)$ with $u, v \in x$, and the weight of $(u, v)$ in $G(s, t)$ is set to the minimum $d_x(u, v)$ over all $x \in X(s, t)$ with $u, v \in x$.

Observe that $G(s, t)$ has $O(k \log n)$ vertices and $O(k^2 \log n)$ edges, so the query time is dominated by the time to run Djikstra from $s$ to $t$ in $G(s, t)$ which is $O(k^2 \log n \log(k \log n))$ (this can be slightly improved using Fibonacci heaps). Towards proving the correctness of the query operation we prove the following auxiliary claim.

CLAIM 1. *For any edge $e = (u, v) \in E$, if $d(x_u) \geq d(x_v)$ then $x_e = x_u$.*

**Proof:** Recall that $x_e$ is the closest bag to the root in $X$ such that both $u, v \in x$, namely, $x_e$ is the bag with minimal $d(x_e)$ among all bags that contain both $u$ and $v$. Note that as $u \in x_e$, $d(x_u) \leq d(x_e)$. Seeking a contradiction, assume that $d(x_u) < d(x_e)$. By property (2) of the tree decomposition it follows that the parent $x'$ of $x_e$ contains $u$. Note that $x'$ must also contain $v$ as $d(x_u) \geq d(x_v)$. It follows that $x'$ contains both $u$ and $v$ and $d(x') < d(x_e)$, contradiction to the definition of $x_e$. ∎

The following lemma proves the correctness of the query operation.

LEMMA 2.3. *For every two nodes $s$ and $t$, $\mathbf{dist}(s, t, G(s, t)) = \mathbf{dist}(s, t)$.*

*Proof.* The proof is by induction on the number of edges on the shortest path from $s$ to $t$ (assume unique shortest path by breaking ties consistently, say with virtual node weights). Let $P(s,t)$ be the shortest path from $s$ to $t$ in $G$. Consider the base case $|P(s,t)| = 1$, i.e., shortest paths consisting of a single edge. Assume without loss of generality that $d(x_t) \leq d(x_s)$. Let $e = (s,t)$. By Claim 1 we have $x_s = x_e$ and thus $x_e \in X(s,t)$. Observe that $d_{x_e}(s,t) = \omega(s,t)$ and thus $\mathbf{dist}(s,t,G(s,t)) = \omega(s,t)$.

Assume the induction hypothesis holds for every two nodes $s',t'$ such that $|P(s',t')| < \ell$ and consider two nodes $s,t$ such that $|P(s,t)| = \ell$. Assume without loss of generality that $d(x_t) \leq d(x_s)$.

If $P(s,t) \subseteq G^-[V(x_s)]$, then in particular $t \in V(x_s)$. Note that if $t \notin x_s$ then by property 2 of the tree decomposition since $t$ belongs to a descendant of $x_s$ then $t$ belongs only to bags in the subtree of $x_s$. It follows that $d(x_s) < d(x_t)$ with a contradiction to the assumption that $d(x_t) \leq d(x_s)$. Hence, $t \in x_s$ and $P(s,t) \subseteq G^-[V(x_s)]$ so $G(s,t)$ will contain the clique edge $(s,t)$ with $d_{x_s}(s,t) = \mathbf{dist}(s,t,G^-[V(x_s)]) = \mathbf{dist}(s,t)$.

Otherwise ($P(s,t) \not\subseteq G^-[V(x_s)]$), let $(v_0 = s, v_1, \ldots, v_{z-1}, v_z = t)$ be the path $P(s,t)$ and let $(v_i, v_{i+1})$ be the minimal index edge such that $(v_i, v_{i+1}) \notin G^-[V(x_s)]$.

Note that $v_1 \in V(x_s)$. To see this notice that all nodes that contain $s$ are in $T(x_s)$. Since $s$ and $v_1$ are connected by an edge, then there must a bag in $T(x_s)$ containing both $s$ and $v_1$. Moreover, note that $x_e$ for $e = (s,v_1)$ must be in $T(x_s)$ as all nodes that contain $s$ are in $T(x_s)$. It follows that $(v_0 = s, v_1) \in G^-[V(x_s)]$. Hence $i \geq 1$.

Since $(v_{i-1}, v_i) \in G^-[V(x_s)]$ then by the tree decomposition properties $v_i \in x_s$. To see this, note that we assume $v_i \in T(x_s)$. In addition, from the assumption that $(v_i, v_{i+1}) \notin G^-[V(x_s)]$ it follows that there must be a bag $y \notin T(x)$ such $v_{i+1} \in y$. Seeking a contradiction, assume $v_i \notin x_s$. By Property 2 it follows that $v_i$ cannot appear in any bag not in $T(x_s)$. However, $v_{i+1}$ does not appear in $T(x)$ and thus there is no bag containing both $v_i$ and $v_{i+1}$, with a contradiction to Property 1.

Due to the minimality of $i$, we have that $P(s,v_i) \subseteq G^-[V(x_s)]$ (as we assume that $(v_i, v_{i+1})$ is the minimal index edge in $v_0 = s, v_1, \ldots, v_{z-1}, v_z = t$ that is not in $G^-[V(x_s)]$). So $G(s,t)$ contains the edge $(s,v_i)$ with $d_{x_s}(s,v_i) = \mathbf{dist}(s,v_i,G^-[V(x_s)]) = \mathbf{dist}(s,v_i)$.

By the induction hypothesis $\mathbf{dist}(v_i,t,G(v_i,t)) = \mathbf{dist}(v_i,t)$. Note that $G(v_i,t) \subseteq G(s,t)$. To see this we show that $x_{v_i}$ is an ancestor of $x_s$ or $x_s$ itself and then it follows by definition that $G(v_i,t) \subseteq G(s,t)$. Recall that $x_{v_i}$ is the closest bag to the root that contains $v_i$, namely, the bag $x_{v_i}$ of minimal $d(x_{v_i})$ among all bags that contain $v_i$. We already showed above that $v_i \in x_s$. It follows that $x_{v_i}$ must be an ancestor of $x_s$ or $x_s$ itself. Hence $G(v_i,t) \subseteq G(s,t)$, It follows that $\mathbf{dist}(v_i,t) \leq \mathbf{dist}(v_i,t,G(s,t)) \leq \mathbf{dist}(v_i,t,G(v_i,t)) = \mathbf{dist}(v_i,t)$. We get that $\mathbf{dist}(s,t,G(s,t)) = \mathbf{dist}(s,v_i) + \mathbf{dist}(v_i,t) = \mathbf{dist}(s,t)$.

**2.3 Update Operation** For a bag $x \in X$ let $H(x)$ be the complete graph whose vertex set is $x$. For every two nodes $u,v \in x$, set $\omega(u,v,x)$ as follows. If $e = (u,v) \in E$ and $x = x_e$, set $\omega(u,v,x) = \omega(u,v)$, otherwise set $\omega(u,v,x) = \infty$. Set the weight of an edge $(u,v)$ in the graph $H(x)$ as $\min\{d_y(u,v), \omega(u,v,x) \mid y \in c(x), u,v \in y\}$.

Given a change to $\omega(u,v)$, the data structure can be updated as follows:

1. update the weight edge table for edge $(u,v)$.
2. let $x_0 \in X$ be the closest bag to the root such that $u,v \in x_0$. Let $x_i$ be the parent of $x_{i-1}$.
3. for $i = 0$ till $x_i$ is the root:
   (a) construct the graph $H(x_i)$
   (b) for all $u',v' \in x_i$, set $d_{x_i}(u',v') := \mathbf{dist}(u',v',H(x_i))$

The following lemmas show the correctness and the time analysis of the update operation.

LEMMA 2.4. *After the update of an edge $(u,v)$, for every bag $x$ and nodes $u',v'$ such that $u',v' \in x$, $d_x(u',v') = \mathbf{dist}(u',v',G^-[V(x)])$.*

*Proof.* Let $x_0 \in X$ be the closest bag to the root such that $u,v \in x_0$. Let $x_i$ be the parent of $x_{i-1}$. By construction, $(u,v) \in G^-[V(x')]$ only for bags $x' \in \{x_i\}$. We thus need to show that the claim holds only for the bags $\{x_i\}$.

The algorithm iterates the bags $\{x_i\}$ from $x_0$ until $x_i$ is the root. Consider iteration $i$ of the algorithm. We next show that assuming that $d_y(u',v') = \mathbf{dist}(u',v',G^-[V(y)])$ for all $y \in c(x_i)$ and $u',v' \in y$ then after the iteration $i$ $d_{x_i}(u',v') = \mathbf{dist}(u',v',G^-[V(x_i)])$ for all $u',v' \in x_i$. Notice that the claim is already satisfied for all $y \in c(x_0)$ as $(u,v) \notin G^-[V(y)]$. We thus don't need to handle the base case separately.

Assume the claim holds for every $x_j$ where $j < i$ and consider $x_i$.

We prove by induction on the number of hops in the shortest path in $G^-[V(x_i)]$ that for every $u',v' \in x_i$, $\mathbf{dist}(u',v',H(x_i)) = \mathbf{dist}(u',v',G^-[V(x_i)])$. Let $P(u',v')$ be the shortest path between $u'$ and $v'$ in

$G^-[V(x_i)]$. If $|P(u',v')| = 1$ then the path consists of a single edge $(u',v')$. Since $(u',v') \in G^-[V(x_i)]$, we have $\mathbf{dist}(u',v',H(x_i)) = \omega(u',v',x_i) = \omega(u',v')$.

Assume the claim holds for any $u'',v''$ such that $|P(u'',v'')| < \ell$ and consider $u',v'$ such that $|P(u',v')| = \ell$. Let $P(u',v') = (v_0 = u', v_1, \ldots, v_{\ell-1}, v_\ell = v')$. Let $w$ be the first vertex after $u'$ on the path $P(u',v')$ such that $w \in x_i$. If $w = v'$, then note that none of the internal vertices on the path $P(u',v')$ belongs to $x_i$. However as we assume that $P(u',v')$ is in $G^-[V(x_i)]$, it follows that $T(x_i)$ contains all internal vertices of $P(u',v')$. Note that every internal vertex $v_j$ for $1 \leq j \leq \ell - 1$ belongs to $T(y_j)$ of exactly one child $y_j \in c(x_i)$. This follows from the fact that $v_j \notin x_i$ and from Property 2. In additional, we claim that $y_{j'} = y_{j'+1}$ for every $1 \leq j' \leq \ell - 1$. To see this, recall that by Property 1 there must be a bag that contains both $v_{j'}$ and $v_{j'+1}$. It follows that $y_{j'} = y_{j'+1}$. Hence $y_1 = y_2 = ... = y_{\ell-1}$. Note that $v_0$ is also contained in $V(y_1)$ by Property 1. Similarly, $v_\ell$ is contained in $V(y_{\ell-1})$. Hence there is a node $y \in c(x_i)$ such that the entire path $P(u',v')$ is contained in $G^-[V(y)]$. Thus the correctness follows by the assumption that the claim holds for the children of $x_i$.

Otherwise, by the induction hypothesis on the number of hops we have $\mathbf{dist}(u',w,H(x_i)) = \mathbf{dist}(u',w,G^-[V(x_i)])$ and $\mathbf{dist}(w,v',H(x_i)) = \mathbf{dist}(w,v',G^-[V(x_i)])$ and thus $\mathbf{dist}(u',v',H(x_i)) = \mathbf{dist}(u',v',G^-[V(x_i)])$ as required.

LEMMA 2.5. *The update operation takes $O(k^3 \log n)$ time.*

*Proof.* Constructing $H(x)$ for some $x \in X$ takes $O(k^2)$ time. Finding all pairs shortest paths in $H(x)$ takes $O(k^3)$ time. Since $O(\log n)$ bags $x \in X$ are updated, the entire update operation takes $O(k^3 \log n)$ time.

Just as in separator-based multi-level graph overlay approach [14] we proceed to update in a bottom up fashion, and use the fact that we can quickly compute distances for level $i$ based on distances that we computed for level $i-1$.

## 3 Tree of Separators for Planar Graphs

The following section presents a variant of the planar separator of [1] that is tailored to this problem (see definition of $\alpha$).

Consider a planar graph $G_{input}$ with edge weights $\omega$. Denote by $V(G_{input})$ and $E(G_{input})$ respectively the sets of vertices and edges of $G_{input}$. Let $G''$ be the planar graph obtained by triangulating a plane embedding of $G_{input}$ (we set the weight of new added edges to be infinity).

We will construct a shortest path decomposition to split the graph into smaller subgraphs recursively. For reasons that will become clearer later on, it would be useful to separate also the edges of the planar graph and not just the vertices. We add a dummy node $w$ in the middle of each edge $(u,v)$ and set the weights $\omega(u,w) = \omega(v,w) = \omega(u,v)/2$. Let $G'$ be the graph obtained by adding these dummy nodes to $G''$.

To triangulate $G'$ we can add two parallel edges to each original edge on each side and set its weight to infinity. The new graph is triangulated.

Let $T$ be a shortest path tree in $G''$ rooted at some vertex $\mathrm{root}(T)$.

The root path of some vertex $v$ of $G''$, denoted by $T_v$, is the path between $\mathrm{root}(T)$ and $v$ in the tree $T$. Two nodes of a tree are *relatives* if one is the ancestor of the other one.

We will make use of the two-path separator for planar graphs [24].

LEMMA 3.1. *Given a subgraph $C$ of $G$, one can find in linear time an edge $(u,v) \in E(G') \setminus E(T)$ such that $C \setminus (T_u \cup T_v)$ is divided into two subgraphs $A, B$ of at most $2|C|/3$ vertices such that no edge of $G$ links a vertex of $A$ to a vertex of $B$. Moreover, $A$ and $B$ lie on different faces of the plane graph $T_u \cup T_v \cup \{(u,v)\}$.*

We recursively apply Lemma 3.1 and we define a separator hierarchy tree $\mathcal{T}$ as follows. The tree $\mathcal{T}$ is binary, each node $\mu$ of $\mathcal{T}$ corresponds to an edge $(u,v)$ of $G'$ induced by Lemma 3.1 applied on some subgraph $C$ of $G$. The root of $\mathcal{T}$ is the edge $(u,v)$ of Lemma 3.1 in the case $C = G$. The two children of $(u,v)$ are then the two edges corresponding to the two path separators when applying Lemma 3.1 to the subgraphs $A$ and $B$. The decomposition stops whenever we find $(u,v)$ for $C$ such that $C \subseteq T_u \cup T_v$, that is there is no more subgraphs $A$ and $B$.

Such a tree $\mathcal{T}$ has depth $O(\log n)$ and can be constructed in $O(n \log n)$ time. We use properties and definitions introduced in [1] and bring them here for completeness.

**Cluster.** Consider a node $\mu = (u,v)$ of $\mathcal{T}$. The *cluster* of a node $\mu$ (denoted by $\textsc{Cluster}(\mu)$) is the subgraph $C$ of $G$ on which we invoked Lemma 3.1 to produce $(u,v)$. The *level* of $\mu$ (denoted by $\textsc{Level}(\mu)$) is the number of edges in $\mathcal{T}$ from $\mu$ to the root of $\mathcal{T}$.

**Cycle-separator.** The *cycle-separator* of node $\mu$ is a subgraph of $G'$ defined recursively as follows. If $\mu$ is the root of $\mathcal{T}$, then $\textsc{Cycle-Sep}(\mu) = T_u \cup T_v \cup \{(u,v)\}$,

otherwise $\text{CYCLE-SEP}(\mu) = \text{CYCLE-SEP}(\mu') \cup T_u \cup T_v \cup \{(u,v)\}$, where $\mu'$ is the parent of $\mu$. The *separator* of $\mu$ (denoted by $\text{SEP}(\mu)$) is simply the subgraph of CYCLE-SEP induced by the edges of $T$, i.e., $\text{SEP}(\mu) = \text{CYCLE-SEP}(\mu) \cap E(T)$. Note that $\text{SEP}(\mu)$ is a subtree of $T$ containing $\text{root}(T)$.

Consider a vertex $x$ of $G$. The node $\mu$ of $\mathcal{T}$ of smallest level such that $x$ belongs to $\text{SEP}(\mu)$ is called the *home* of $x$, and is denoted by $\text{HOME}(x)$. This definition is justified by the fact that the set of nodes of $\mathcal{T}$ containing in its separator any given vertex of $G$ forms a non-empty subtree of $\mathcal{T}$.

**Apices.** Note that if $\mu'$ is the parent of $\mu$ then vertices of $\text{CYCLE-SEP}(\mu')$ separate $\text{CLUSTER}(\mu)$ from the rest of the graph. We define a subset of $\text{CYCLE-SEP}(\mu')$ which we call $\text{FRAME}(\mu)$ and a set of special vertices in $\text{FRAME}(\mu)$ that we call $\text{RELAPICES}(\mu)$. Roughly speaking we will show that (1) vertices in $\text{FRAME}(\mu)$ separate $\text{CLUSTER}(\mu)$ from the rest of the graph; (2) the size of $\text{RELAPICES}(\mu)$ is $\tilde{O}(1)$;

The *apices* of $\mu$ (denoted by $\text{APICES}(\mu)$) are the vertices of $\text{CYCLE-SEP}(\mu)$ with degree $\geq 3$ in $\text{CYCLE-SEP}(\mu)$. The following key property was proved as Property 1 in [1]:

PROPERTY 3.1. *[1] For every node $\mu$ of $\mathcal{T}$, $|\text{APICES}(\mu)| \leq 2 \cdot \text{LEVEL}(\mu) + 1$.*

**Tails.** The *tails* of $\mu$, denoted by $\text{TAILS}(\mu)$, is the subgraph induced by the all the edges of $\text{SEP}(\mu)$ not in $\text{SEP}(\mu')$, if $\mu$ has a parent $\mu'$, or simply induced by the edges of $\text{SEP}(\mu)$ if $\mu$ is the root. In other words, $\text{SEP}(\mu) = \text{TAILS}(\mu) \cup \text{SEP}(\mu')$. Each one of the subpaths $\text{TAILS}(\mu) \cap T_u$ and $\text{TAILS}(\mu) \cap T_v$ is called *tail*. If $\mu$ is a leaf in $\mathcal{T}$ then we set $\text{TAILS}(\mu) = \emptyset$.

**Frame.** The subgraph *frame* of $\mu$, is a minimal part of $\text{CYCLE-SEP}(\mu)$ needed for separating $\text{CLUSTER}(\mu)$ from the rest of the graph. If $\mu$ is the root of $\mathcal{T}$, then $\text{FRAME}(\mu)$ is the empty graph. Otherwise, $\text{FRAME}(\mu)$ is the subgraph of all edges $e$ such that there exists a face $F$ and $e \in F$ and $F \subseteq \text{CLUSTER}(\mu)$. Observe that, any path from a vertex of $\text{CLUSTER}(\mu)$ to a vertex outside the cluster has to intersect $\text{FRAME}(\mu)$.

Since an edge in a planar graph can belong to at most two faces we have:

PROPERTY 3.2. *For every level $\ell$, every edge $e$ belongs to the frame of at most two regions in that level.*

**Paths and the parameter $\alpha$** The paths $P(\mu) = \text{FRAME}(\mu) \cap \text{SEP}(\mu)$ are the set of shortest paths that separate $\mu$ and belong to the frame. Let $\alpha$ be the maximum over all $\mu$ of the number of shortest path $|P(\mu)|$. We can trivially bound $\alpha$ by the depth of the tree $O(\log n)$. In the full version we will show that by interleaving stages of $|P(\mu)|$ reduction with stages of reducing the number of vertices in $\mu$ we can guarantee that $\alpha \leq 10$ and still have a depth $O(\log n)$ tree.

**Region.** The *region* of $\mu$ (denoted by $\text{REG}(\mu)$) is the subgraph of $G$ induced by all the vertices of $\text{CLUSTER}(\mu) \cup \text{FRAME}(\mu)$.

Let $\text{NEWAPICES}(\mu) = \text{APICES}(\mu) \setminus \text{APICES}(\mu')$ be the new apices of node $\mu$ w.r.t. its parent $\mu'$. The *relevant apices* of $\mu$ (denoted by $\text{RELAPICES}(\mu)$) is the set of vertices in $(\text{APICES}(\mu) \cap \text{FRAME}(\mu)) \setminus \text{NEWAPICES}(\mu)$. Intuitively, the relevant apices for a node $\mu$ are the apices that are the apices that a relevant for seperating $\mu$ from the rest of the graph (so apices not on the frame or new apices used for $\mu$'s children are not relevant). Note that by Property 3.1, $|\text{NEWAPICES}(\mu)| \leq 2$.

The last useful property is the following.

PROPERTY 3.3. $\sum_{\mu \in \mathcal{T}} |V(\text{REG}(\mu))| = O(n \log n)$.

Denote by $\text{ANCESTORS}(\mu)$ be the set of proper ancestor nodes of $\mu$ in $\mathcal{T}$, and let $\text{ANCESTORS}[\mu] = \text{ANCESTORS}(\mu) \cup \{\mu\}$.

In the remaining, we will mainly focus on the regions of the nodes of $\mathcal{T}$, a notion which is more convenient to use. We extend all the definitions we saw so far about a node $\mu$ of $\mathcal{T}$ to its region $R = \text{REG}(\mu)$. This mainly concerns TAILS, RELAPICES, ANCESTORS, LEVEL. For instance, $\text{RELAPICES}(R)$ is just a short for $\text{RELAPICES}(\mu)$ such that $R = \text{REG}(\mu)$.

For a region $R$, let $R_{input}$ be the region $R$ with the edge weight of the original graph $G_{input}$. Similarly, let $T_{input}$ be the tree $T$ with the original edge weight of $G_{input}$.

## 4 Edge Dynamic Distance Oracle for Planar Graphs

In this section we show a construction for dynamic distance oracle for planar graphs with dynamic edge weight changes. The algorithm is given an initial weighted graph $G_{input}$ and the assumption is that during each edge weight update the shortest path metric of the new graph $G$ has the property that the stretch is at most $M'$ relative to the initial graph $G_{input}$. Formally, after every updated the new weights for graph $G$ obey:

$$\forall u, v \in V(G): \quad d_{G_{input}}(u,v) \leq d_G(u,v) \leq M' \cdot d_{G_{input}}(u,v) \tag{4.1}$$

In the analysis, we use a simpler assumption, that the weight of every edge can change by at most $M$ factor:

Table 1: Examples for definitions CYCLE-SEP, APICES, SEPof Regions $\mu_0, \ldots, \mu_3$ for Figure 1.

| Region | CYCLE-SEP | APICES | SEP |
|---|---|---|---|
| $\mu_0 = (u_0, v_0)$ | $T_{u_0} \cup T_{v_0} \cup \{(u_0, v_0)\}$ | $\emptyset$ | $T_{u_0} \cup T_{v_0}$ |
| $\mu_1 = (u_1, v_1)$ | $T_{u_0} \cup T_{v_0} \cup T_{u_1} \cup T_{v_1} \cup \{(u_0, v_0), (u_1, v_1)\}$ | $a_1, a_2$ | $\bigcup_{0 \le j \le 1} T_{u_j} \cup T_{v_j}$ |
| $\mu_2 = (u_2, v_2)$ | $\bigcup_{0 \le j \le 2} T_{u_j} \cup T_{v_j} \cup \bigcup_{0 \le j \le 2} (u_j, v_j)$ | $a_1, a_2, a_3, a_4$ | $\bigcup_{0 \le j \le 2} T_{u_j} \cup T_{v_j}$ |
| $\mu_3 = (u_3, v_3)$ | $\bigcup_{0 \le j \le 3} T_{u_j} \cup T_{v_j} \cup \bigcup_{0 \le j \le 3} (u_j, v_j)$ | $a_1, \ldots, a_6$ | $\bigcup_{0 \le j \le 3} T_{u_j} \cup T_{v_j}$ |

Table 2: Examples for definitions TAILS, FRAME, NEWAPICES, RELAPICESof Regions $\mu_0, \ldots, \mu_3$ for Figure 1. $\underset{x,y}{T}$ to denote the unique path in $T$ from $x$ to $y$.

| Region | TAILS | FRAME | NEWAPICES | RELAPICES |
|---|---|---|---|---|
| $\mu_0$ | $T_{u_0} \cup T_{v_0}$ | $\emptyset$ | $\emptyset$ | $\emptyset$ |
| $\mu_1$ | $(x, y) \in E(T_{u_1} \cup T_{v_1}) \setminus E(T_{u_0} \cup T_{v_0})$ | $T_{u_0} \cup T_{v_0} \cup (u_0, v_0)$ | $a_1, a_2$ | $\emptyset$ |
| $\mu_2$ | $(x, y) \in E(T_{u_2} \cup T_{v_2}) \setminus E(T_{u_0})$ | $\underset{a_1,u_0}{T} \cup \underset{a_1,u_1}{T} \cup \underset{a_2,v_0}{T} \cup \underset{a_2,v_1}{T} \cup (u_0, v_0), (u_1, v_1)$ | $a_3, a_4$ | $a_1, a_2$ |
| $\mu_3$ | $(x, y) \in E(T_{u_3} \cup T_{v_3}) \setminus E(T_{u_2} \cup T_{v_2})$ | $\underset{a_4,u_2}{T} \cup \underset{a_4,v_2}{T} \cup (u_2, v_2)$ | $a_5, a_6$ | $a_4$ |

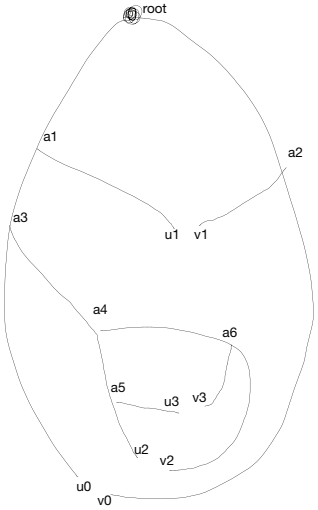


Figure 1: Example of four nested regions: $\mu_0 = (u_0, v_0), \ldots, \mu_3 = (u_3, v_3)$, see Tables 1 and 2 for values of CYCLE-SEP, APICES, SEP, TAILS, FRAME, NEWAPICES, RELAPICES of these regions

$$\forall (u, v) \in E(G) : \quad (4.2)$$
$$\omega(u, v, G_{input}) \le \omega(u, v, G) \le M \cdot \omega(u, v, G_{input}),$$

where $\omega(u, v, H)$ is the weight of the edge $(u, v)$ in the graph $H$.

A dynamic scheme under assumption 4.2 with a parameter $2M$ can easily be transformed into a dynamic scheme under assumption 4.1 with parameter $M$ using the following reduction. In the update algorithm given a weight change for an edge $(u, v)$ check if the new weight satisfies $\omega(u, v, G_{input}) \le \omega(u, v, G) \le M \cdot \omega(u, v, G_{input})$, if so then simply invoke the update algorithm tailored for assumption 4.2. Otherwise, temporarily set the weight of the edge to be $2M \cdot \omega(u, v, G_{input})$ (so assumption 4.2 holds). By assumption 4.1, there must be a shorter path from $u$ to $v$. Use the data structure to compute this shortest path and finally set the edge $(u, v)$ to the distance of the computed path.

**4.1 High-level Overview** We decompose the planar graph using shortest path separators. In a bounded tree width graph each region is separated by $O(k)$ vertex separators, in a planar graph each region is separated by $O(\log n)$ shortest paths. Since each path may contain $O(n)$ vertexes, a common solution is to we maintain only a subset of these vertexes. In particular we use a hierarchy of net-points on each path. In the bounded tree width graph we maintained a clique of all the distances between all separators of each region. Here we only maintain a sparse 'low-stretch' overlay spanner between the net points of each scale. This is done by storing for each net

point of scale $i$ only poly-log edges to other net points of scale $i$. The update algorithm again proceeds in a bottom-up fashion. To update the scale $i$ overlay spanner in a region $R$ we first need to update all child regions of $R$ (as in the tree width case) and also all scale $< i$ overlay spanners for region $R$. As in the bounded tree width case, the query operation performs a bidirectional upward search from $s$ and from $t$. In order to bound the size of the search space we use the fact that only poly-log net points in each region are needed in order to guarantee a $1+\epsilon$ path from $s$ to $t$.

**4.2 Preprocessing** Given a graph $G_{input}$ fix a shortest path tree $T$ rooted at $\mathrm{root}(T)$. Construct the hierarchy tree $\mathcal{T}$ on $G_{input}$ and $T$ as in Section 3. Let $D$ be the maximum of the diameter of the graph $G_{input}$ and $n$ (assuming the minimal edge weight is 1). Let $\epsilon_2 = \epsilon/(4(2\log n + \log D))$ and $\epsilon_1 = \epsilon_2/10$.

Given any base set $B \subseteq V$ we iteratively define hierarchical subset of net points $N(B,i)$ for $1 \leq i \leq \log D$ as follows. Let $N(B,1) = B$ be the set containing all vertices. Given $N(B,i-1)$, the set $N(B,i) \subseteq N(B,i-1)$ is defined by the following process. Consider all vertices $x \in N(B,i-1)$ in an increasing order of their distance from $\mathrm{root}(T)$ (handle ties in any consistent manner). Let $y \in N(B,i-1)$ be the closest ancestor of $x$ (again break ties consistently), that is, the node $y \in N(B,i-1)$ such that $y$ is an ancestor of $x$ and $\mathbf{dist}(x,y)$ is minimal. Add $x$ to $N(B,i)$ if $\mathbf{dist}(x,y) > \epsilon_1/(16M)\cdot 2^i$.

For a Region $R$ and index $i$, let $N_i(R) = N(\textsc{Frame}(R) \cup \textsc{Tails}(R), i) \cup \textsc{RelApices}(R)$. In words $N_i(R)$ is the union of the net points of scale $i$ on $\textsc{Frame}(R) \cup \textsc{Tails}(R)$ with the relevant apices of $R$.

For a vertex $v \in R$, region $R$ and scale $i$, let $\mathcal{RN}_1(v,R,i)$ be the nodes $x \in N_i(R)$ such that $\mathbf{dist}(v,x,R) \leq 2^{i+2}$ and let $\mathcal{RN}_2(v,R,i)$ be the nodes $x \in N_i(R)$ such that $\mathbf{dist}(v,x,R) \leq 2^i$.

Observe that the sets $\mathcal{RN}_1(v,R,i)$ and $\mathcal{RN}_2(v,R,i)$ for a node $v \in R$, and region $R$ are constructed during the preprocessing phase and they are fixed, namely, they rely only on the original edge weights of $G_{input}$.

**4.3 Data Structure** The data structure stores the following fixed components:

1. The hierarchy tree $\mathcal{T}$.
2. For every region $R$ and index $i$ the set $N_i(R)$.
3. For every $v \in \textsc{Reg}(R)$ and index $i$ the sets $\mathcal{RN}_1(v,R,i)$ and $\mathcal{RN}_2(v,R,i)$.
4. For every edge $e$, let $\mathrm{RR}(e)$ be all the regions that $e$ belongs to, note that by Lemma 3.2 and the fact that the depth of $\mathcal{T}$ is $O(\log n)$ we have that $|\mathrm{RR}(e)| = O(\log n)$. Store the set $\mathrm{RR}(e)$ for every $e \in E$.

The data structure stores the following two dynamic structures, dynamic tree and dynamic region spanner:

**Dynamic tree.** Maintain the distances $\mathbf{dist}(\mathrm{root}(T), v, T)$, namely, the distance from $\mathrm{root}(T)$ to $v$ in the tree $T$ for every $v \in V$. This can be done by dynamic trees [29] in $O(\log n)$ update time for each edge update. This allows to compute $\mathbf{dist}(x_1, x_2, T)$ when $x_1$ is an ancestor of $x_2$. Note that after edge weight updates, $T$ will not necessarily be a shortest path tree from $\mathrm{root}(T)$.

**Dynamic region spanner.** For every region $R$ and index $i$ a graph $G_i(R)$. The vertex set of $G_i(R)$ is $N_i(R)$. The edge set $E(G_i(R))$ of $G_i(R)$ is a subset of $N_i(R) \times N_i(R)$. Let $\omega(u,v,G_i(R))$ denote the weight of the edge $(u,v)$ in $G_i(R)$ for some $u,v \in N_i(R)$ such that $(u,v) \in E(G_i(R))$ or infinity in case $(u,v) \notin E(G_i(R))$.

The graph $G_i(R)$ will always satisfy the following key properties (initially using equality):

PROPERTY 4.1.

1. *Every node in $N_i(R)$ has edges in $G_i(R)$ to all nodes in $N_i(R)$ at distance (in $R$) at most $2^i$ from it.*
2. *For every region $R$ of level $\textsc{Level}(R) = \ell$ and every two net-points $x_1, x_2 \in N_i(R)$ such that $(x_1,x_2) \in E(G_i(R))$ we have:*

$$\begin{aligned}\mathbf{dist}(u,v,R) &\leq \omega(u,v,G_i(R)) \\ &\leq \mathbf{dist}(u,v,R)(1+\epsilon_2(2(\log n - \ell)+i)).\end{aligned}$$

The next lemmas bound the size of the data structure.

LEMMA 4.1. *For every vertex $v$ and region $R$ such that $v \in R$, $|\mathcal{RN}_1(v,R,i)| = O(\epsilon_1^{-1} M\alpha)$.*

**Proof:** Consider a vertex $v$ and region $R$ such that $v \in R$. Recall that the set $\mathcal{RN}_1(v,R,i) = \{z \in N_i(R) \mid \mathbf{dist}(v,z,R_{input}) \leq 2^{i+2}\}$.

Recall also that $\textsc{Frame}(R)$ consists of at most $\alpha$ paths $P(R)$ such that each $P = P(x,y,T) \in P(R)$ is a path in $T$ from some $x$ to some ancestor $y$. We will show that for each such path $P(x,y,T_{input})$ ,$|\mathcal{RN}_1(v,R,i) \cap P(x,y,T_{input})| = O(M/\epsilon_1)$.

Let $c_v \in P(x,y,T_{input})$ be the vertex with minimal $\mathbf{dist}(v,c_v,R_{input})$. Let $P = P(x,y,T_{input})$ and

let $P(c_v, 2^{i+3})$ be the sub-path of $P$ consisting of all vertices at distance (in $R_{input}$) at most $2^{i+3}$ from $c_v$. By the triangle inequality and the fact that $P$ is a shortest path in $R_{input}$ we have that for every vertex $z \notin P(c_v, 2^{i+3})$, $\mathbf{dist}(v, z, R_{input}) > 2^{i+2}$. Hence the set $\mathcal{RN}_1(v, R, i) \cap P(x, y, T_{input}) \subseteq P(c_v, 2^{i+3})$. Note that by construction of $N_i$ there are no two vertices in $z_1, z_2 \in P(x, y, T_{input}) \cap N_i$ such that $\mathbf{dist}(z_1, z_2, R_{input}) < \epsilon_1/(16M) \cdot 2^i$. It follows that $|\mathcal{RN}_1(v, R, i) \cap P(x, y, T_{input})| = O(\epsilon_1^{-1} M)$. Hence the size of $\mathcal{RN}_1(v, R, i)$ is $O(\epsilon_1^{-1} M\alpha)$. ∎

LEMMA 4.2. *The size of the data structure is* $O(\epsilon_1^{-1} nM \log^2 n) = O(\epsilon^{-1} nM \log^2 n \log D)$.

**Proof:** By Property 3.3 the size of $\mathcal{T}$ is $O(n \log n)$.

Storing the hierarchical sets $N_i(R)$ for all scales $i$ and regions $R$ can be done by storing for every vertex $v \in V$ the largest index $j$ such that $v \in N_j$. Hence storing the sets $N_i$ can be done in $O(n)$ words. Consider a vertex $v$ and region $R$ such that $v \in R$. By Lemma 4.1 we have $|\mathcal{RN}_1(v, R, i)| = O(\epsilon_1^{-1} M\alpha)$. As $\mathcal{RN}_2(v, R, i) \subseteq \mathcal{RN}_1(v, R, i)$ we also have $|\mathcal{RN}_2(v, R, i)| = O(\epsilon_1^{-1} M\alpha)$.

By Property 3.3 we have $\sum_{\mu \in \mathcal{T}} |V(\text{REG}(\mu))| = O(n \log n)$. Therefore, storing all sets $\mathcal{RN}_1(v, R, i)$ and $\mathcal{RN}_2(v, R, i)$ for every region $R$, $v \in V$ and $i$ requires $O(\epsilon_1^{-1} Mn \log n)$ words.

Consider a region $R$ and index $i$. Next, we bound the size of $G_i(R)$. Consider a vertex $v \in N_i(R)$. Note that the set of neighbours of $v$ in $G_i(R)$ is a subset of $\mathcal{RN}_2(v, R, i)$. Thus the number of edges in $G_i(R)$ is bounded by $O(\epsilon_1^{-1}|N_i(R)|M\alpha) = O(\epsilon_1^{-1}|R|M\alpha)$. Hence storing the graphs $G_i(R)$ for every $1 \le i \le \log n$ takes $O(\epsilon_1^{-1}|R|M \log n)$ size. Therefore storing all $G_i(R)$ for every $1 \le i \le \log n$ and every region $R$ takes $O(\epsilon_1^{-1} nM \log^2 n)$ size.

Finally, storing $\text{RR}(e)$ for every edge $e$ takes $O(n \log n)$ size. ∎

The next auxiliary lemma shows that the net-points are dense enough.

LEMMA 4.3. *For every region $R$, vertex $x \in$ FRAME$(R) \cup$ TAILS$(R)$, and index $1 \le i \le \log D$, there exists a net-point $x' \in N_i(R)$ that satisfies the following:*

1. *$x'$ is an ancestor of $x$ (or $x$ itself).*
2. *The path $P(x, x', T) \cap$ APICES$(R) \setminus \{x'\} = \emptyset$, namely, there are no apices of $R$ in $P(x, x', T) \setminus \{x'\}$.*
3. $\mathbf{dist}(x, x', R_{input}) \le \epsilon_1 2^i/(16M)$.

**Proof:** Consider a region $R$, vertex $x \in$ FRAME$(R)$, and index $i$. Let $y$ be the closest ancestor of $x$ (with respect to $T$) that is in APICES$(R)$, that is, the ancestor of $x$ that is in APICES$(R)$ such that the distance from $x$ to $y$ in $T$ is minimal. Note that the path $P(x, y, T)$ belongs to FRAME$(R)$ by definition. By construction of $N(\text{FRAME}(R), i)$, there must be an ancestor $x_1 \in N_i$ of $x$ such that $\mathbf{dist}(x, x_1, T_{input}) \le \epsilon_1 2^i/(16M)$. If $x_1 \in P(x, y, T)$ then $x_1 \in$ FRAME$(R)$ and we are done. Otherwise $x_1 \notin P(x, y, T)$ then note that $\mathbf{dist}(x, y, T_{input}) \le \mathbf{dist}(x, x_1, T_{input}) \le \epsilon_1 2^i/(16M)$. In addition, $y \in$ FRAME$(R)$ so $y \in$ RELAPICES$(R)$ hence $y \in N_i(R)$. ∎

**4.4 Query Algorithm** Given two nodes $s$ and $t$, the distance query builds a graph $H = H(s, t)$ and then runs a shortest path algorithm from $s$ to $t$ on $H$ and return this distance as the estimate on the distance between $s$ and $t$.

The graph $H = H(s, t)$ is constructed as follows: Initially set $H$ to be the empty graph. For every $x \in \{s, t\}$, $R \in$ REGIONS(ANCESTORS[HOME$(x)$]) and index $1 \le i \le \log D$ add the set of vertices $\mathcal{RN}_1(x, R, i)$ to $H$ . For every two vertices $x_1, x_2$: if $x_1$ and $x_2$ are connected by an edge in $G_i(R)$ for some $R \in$ REGIONS(ANCESTORS[HOME$(s)$]) $\cup$ REGIONS(ANCESTORS[HOME$(t)$]) and index $i$, add an edge $(x_1, x_2)$ and set its weight to be $\min\{\omega(x_1, x_2, G_i(R)) \mid R \in$ REGIONS(ANCESTORS[HOME$(s)$]) $\cup$ REGIONS(ANCESTORS[HOME$(t)$]), $1 \le i \le \log D\}$.

The next two lemma proves the correctness of the query algorithm and bound its running time (see proofs in Appendix A).

LEMMA 4.4. *The returned distance* $\tilde{\mathbf{dist}}(s, t)$ *satisfies* $\mathbf{dist}(s, t) \le \tilde{\mathbf{dist}}(s, t) \le (1 + \epsilon)\mathbf{dist}(s, t)$.

LEMMA 4.5. *The query time is* $O(\epsilon_1^{-2} M^2 \alpha^2 \log^2 n \log^2 D) = O(\epsilon^{-2} M^2 \alpha^2 \log^2 n \log^3 D)$.

**4.5 Edge Weight Update Algorithm** The algorithm to update the data structure given a weigh update of edge $e = (s, t)$ is as follows. Let $\omega(s, t)$ be the new weight of $(s, t)$ and let $\omega_{old}(s, t)$ be the previous weight. The first step is to update the distances $\mathbf{dist}(\text{root}(T), v, T)$. This can be done easily by [29]. Note that the edge $e$ effects distances in only the regions $\text{RR}(e)$ (see item 4 of the data structure). So we only need to fix distances in these regions. We start from the leaves and work towards to the root.

**Updating a leaf region $R$ such that $e \in R$.** By definition of being a leaf, the cluster of $R$ is empty. In order to update the distances in $R$ do the following.

1. Delete all edges in $G_i(R)$ between nodes $\mathcal{RN}_2(s,R,i) \cap \mathcal{RN}_2(t,R,i)$ for every $1 \leq i \leq \log D$.
2. To recompute the weight of these edges construct a graph $H$ as follows. Add to the graph $H$ all nodes $\mathcal{RN}_1(s,R,i) \cap \mathcal{RN}_1(t,R,i)$ for every $1 \leq i \leq \log D$. For every two nodes $x,y \in V(H)$ that are related (that is either $x$ is ancestor of $y$ or the other way around) add an edge between them of weight $\mathbf{dist}(x,y,T)$. In addition, for every ancestor path separator $(T_v,T_u)$ if either $v$ or $u$ are dummy then add the physical edge the dummy node represents with its weight. Invoke all pairs shortest paths in $H$. For every two nodes $x,y \in N_i(R)$ such that $\mathbf{dist}(x,y,H) \leq 2^i$ add an edge between them to $G_i(R)$ and set $\omega(x,y,G_i(R)) = \mathbf{dist}(x,y,H)$ for every $1 \leq i \leq \log D$.

**Updating a non-leaf region $R$ such that $e \in R$.** Let $R_1$ and $R_2$ be the children regions of $R$ (which we assume we already updated).

1. Delete all edges in $G_i(R)$ between nodes $x_1,x_2 \in \mathcal{RN}_2(s,R,i) \cap \mathcal{RN}_2(t,R,i)$ for every $1 \leq i \leq \log D$ such that $\omega(x_1,x_2,G_i(R)) \geq 2^{i-1}$.
2. To recompute the weight of these edges construct a graph $H$ as follows. The set of nodes of $H$ is $\cup_{1 \leq i \leq \log D} \mathcal{RN}_1(s,R,i) \cap \mathcal{RN}_1(t,R,i)$. For every two nodes $x_1,x_2$ that are related add the edge $(x,y)$ to $H$ and set its weight to be $\mathbf{dist}(x_1,x_2,T)$. In addition if $(x_1,x_2) \in E(G)$ then simply add the edge $(x_1,x_2)$ with weight $\omega(x_1,x_2)$. For every two nodes $x_1,x_2 \in V(H)$ that are connected by an edge in $G_i(R)$ add the edge $(x_1,x_2)$ with weight $\omega(x_1,x_2,G_i(R))$ for some $1 \leq i \leq \log D$.

   For every two nodes $x_1,x_2 \in V(H)$ that are connected by an edge in $G_i(R_j)$: if the edge $(x_1,x_2)$ already exists in $H$ then set is weight $\omega(x_1,x_2,H) = \min\{\omega(x_1,x_2,H), \omega(x_1,x_2,R_j)\}$ otherwise add the edge $(x_1,x_2)$ with weight $\omega(x_1,x_2,G_i(R_j))$ for some $1 \leq i \leq \log D$ and $j \in \{1,2\}$.

   Invoke all pairs shortest paths in $H$. For every two nodes $x,y \in N_i(R)$ such that $\mathbf{dist}(x,y,H) \leq (1+\epsilon)2^i$ add an edge between them to $G_i(R)$ and set $\omega(x,y,G_i(R)) = \min\{\mathbf{dist}(x,y,H), \omega(x,y,G_{i-1}(R))\}$ for every $1 \leq i \leq \log D$.

The next two lemma show that the update algorithm maintains Property 4.1 and bound the run time (see proofs in Appendix A).

LEMMA 4.6. *After the edge weight update operation, for every index $1 \leq i \leq \log n$, region $R$, and two net-points $x_1,x_2 \in N_i(R)$: $\omega(x_1,x_2,G_i(R)) \leq (1+\epsilon_2(2(\log n - \text{LEVEL}(R)) + i))\mathbf{dist}(x_1,x_2,R)$.*

LEMMA 4.7. *The update algorithm time is $O(\epsilon_1^{-3}M^4\alpha^3 \log^4 n \log^3 D) = O(\epsilon^{-3}M^4\alpha^3 \log^4 n \log^6 D)$.*

**Proof:** Consider a region $R$. as shown in the proof of Lemma 4.5, building the graph $H$ for $R$ can be done in time $O(\epsilon_1^{-2}M^2\alpha^2 \log^2 n \log^2 D)$ and it contains $O(\epsilon_1^{-1}M\alpha \log n \log D)$ vertices and $O(\epsilon_1^{-2}M^2\alpha^2 \log^2 n \log^2 D)$ edges. So all pairs shortest paths in $H$ takes $O(\epsilon_1^{-3}M^4\alpha^3 \log^3 n \log^3 D)$ time.

The update phase updates $O(\log n)$ regions. So the total update time is $O(\epsilon_1^{-3}M^4\alpha^3 \log^4 n \log^3 D)$. ∎

## References


[1] Ittai Abraham, Shiri Chechik, and Cyril Gavoille. Fully dynamic approximate distance oracles for planar graphs via forbidden-set distance labels. In *Proceedings of the 44th symposium on Theory of Computing*, STOC '12, pages 1199–1218, New York, NY, USA, 2012. ACM.

[2] Ittai Abraham, Daniel Delling, Andrew V. Goldberg, and Renato F. Werneck. A Hub-Based Labeling Algorithm for Shortest Paths on Road Networks. In Panos M. Pardalos and Steffen Rebennack, editors, *Proceedings of the 10th International Symposium on Experimental Algorithms (SEA'11)*, volume 6630 of *Lecture Notes in Computer Science*, pages 230–241. Springer, 2011.

[3] Ittai Abraham, Daniel Delling, Andrew V. Goldberg, and Renato F. Werneck. Hierarchical Hub Labelings for Shortest Paths. In Leah Epstein and Paolo Ferragina, editors, *Proceedings of the 20th Annual European Symposium on Algorithms (ESA'12)*, volume 7501 of *Lecture Notes in Computer Science*, pages 24–35. Springer, 2012.

[4] Holger Bast, Stefan Funke, and Domagoj Matijevic. Ultrafast Shortest-Path Queries via Transit Nodes. In Camil Demetrescu, Andrew V. Goldberg, and David S. Johnson, editors, *The Shortest Path Problem: Ninth DIMACS Implementation Challenge*, volume 74 of *DIMACS Book*, pages 175–192. American Mathematical Society, 2009.

[5] Holger Bast, Stefan Funke, Peter Sanders, and Dominik Schultes. Fast Routing in Road Networks with Transit Nodes. *Science*, 316(5824):566, 2007.

[6] Surender Baswana, Ramesh Hariharan, and Sandeep Sen. Improved decremental algorithms for maintaining transitive closure and all-pairs shortest paths. *J. Algorithms*, 62(2):74–92, April 2007.

[7] Reinhard Bauer. Dynamic Speed-Up Techniques for Dijkstra's Algorithm, 2006. Master's thesis, Universität Karlsruhe (TH), Fakultät für Informatik.

[8] Aaron Bernstein. Fully dynamic (2 + epsilon) approximate all-pairs shortest paths with fast query and close to linear update time. In *Proceedings of the 2009 50th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '09, pages 693–702, Washington, DC, USA, 2009. IEEE Computer Society.

[9] Hans L. Bodlaender. Nc-algorithms for graphs with small treewidth, 1988.

[10] Glencora Borradaile and Philip N. Klein. An $o(n \log n)$ algorithm for maximum st-flow in a directed planar graph. *Journal of the ACM*, 56(2), 2009.

[11] Glencora Borradaile, Philip N. Klein, Shay Mozes, Yahav Nussbaum, and Christian Wulff-Nilsen. Multiple-source multiple-sink maximum flow in directed planar graphs in near-linear time. In *Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science*, 2011.

[12] Bruno Courcelle and Andrew Twigg. Compact forbidden-set routing. In *Proceedings of the 24th annual conference on Theoretical aspects of computer science*, STACS'07, pages 37–48, Berlin, Heidelberg, 2007. Springer-Verlag.

[13] Bruno Courcelle and Andrew Twigg. Constrained-path labellings on graphs of bounded clique-width. *Theor. Comp. Sys.*, 47(2):531–567, August 2010.

[14] Daniel Delling, Andrew V. Goldberg, Thomas Pajor, and Renato F. Werneck. Customizable Route Planning. In Panos M. Pardalos and Steffen Rebennack, editors, *Proceedings of the 10th International Symposium on Experimental Algorithms (SEA'11)*, volume 6630 of *Lecture Notes in Computer Science*, pages 376–387. Springer, 2011.

[15] Daniel Delling, Peter Sanders, Dominik Schultes, and Dorothea Wagner. Engineering Route Planning Algorithms. In Jürgen Lerner, Dorothea Wagner, and Katharina A. Zweig, editors, *Algorithmics of Large and Complex Networks*, volume 5515 of *Lecture Notes in Computer Science*, pages 117–139. Springer, 2009.

[16] Daniel Delling and Dorothea Wagner. Landmark-Based Routing in Dynamic Graphs. In Camil Demetrescu, editor, *Proceedings of the 6th Workshop on Experimental Algorithms (WEA'07)*, volume 4525 of *Lecture Notes in Computer Science*, pages 52–65. Springer, June 2007.

[17] Camil Demetrescu and Giuseppe F. Italiano. A new approach to dynamic all pairs shortest paths. *J. ACM*, 51(6):968–992, November 2004.

[18] David Eisenstat and Philip N. Klein. Linear-time algorithms for max flow and multiple-source shortest paths in unit-weight planar graphs. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 735–744, New York, NY, USA, 2013. ACM.

[19] Robert Geisberger, Peter Sanders, Dominik Schultes, and Christian Vetter. Exact Routing in Large Road Networks Using Contraction Hierarchies. *Transportation Science*, 46(3):388–404, August 2012.

[20] Martin Holzer, Frank Schulz, and Dorothea Wagner. Engineering Multilevel Overlay Graphs for Shortest-Path Queries. *ACM Journal of Experimental Algorithmics*, 13(2.5):1–26, December 2008.

[21] Sungwon Jung and Sakti Pramanik. An efficient path computation model for hierarchically structured topographical road maps. *IEEE Trans. on Knowl. and Data Eng.*, 14(5):1029–1046, September 2002.

[22] Philip N. Klein, Shay Mozes, and Christian Sommer. Structured recursive separator decompositions for planar graphs in linear time. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 505–514, New York, NY, USA, 2013. ACM.

[23] Philip N Klein and Sairam Subramanian. A fully dynamic approximation scheme for shortest paths in planar graphs. *Algorithmica*, 22(3):235–249, 1998.

[24] Richard J. Lipton and Robert E Tarjan. A separator theorem for planar graphs. 1977.

[25] Liam Roditty and Uri Zwick. Dynamic approximate all-pairs shortest paths in undirected graphs. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '04, pages 499–508, Washington, DC, USA, 2004. IEEE Computer Society.

[26] Dominik Schultes and Peter Sanders. Dynamic Highway-Node Routing. In Camil Demetrescu, editor, *Proceedings of the 6th Workshop on Experimental Algorithms (WEA'07)*, volume 4525 of *Lecture Notes in Computer Science*, pages 66–79. Springer, June 2007.

[27] Frank Schulz, Dorothea Wagner, and Karsten Weihe. Dijkstra's algorithm on-line: an empirical case study from public railroad transport. *J. Exp. Algorithmics*, 5, December 2000.

[28] Frank Schulz, Dorothea Wagner, and Christos D. Zaroliagis. Using multi-level graphs for timetable information in railway systems. In *Revised Papers from the 4th International Workshop on Algorithm Engineering and Experiments*, ALENEX '02, pages 43–59, London, UK, UK, 2002. Springer-Verlag.

[29] Daniel Dominic Sleator and Robert Endre Tarjan. A data structure for dynamic trees. *J. Comput. Syst. Sci.*, 26(3):362–391, 1983.

[30] Mikkel Thorup. Fully-dynamic all-pairs shortest paths: Faster and allowing negative cycles. In *SWAT'04*, pages 384–396, 2004.

[31] Mikkel Thorup. Worst-case update times for fully-dynamic all-pairs shortest paths. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, STOC '05, pages 112–119, New York, NY, USA, 2005. ACM.

[32] Dorothea Wagner, Thomas Willhalm, and Christos Zaroliagis. Geometric Containers for Efficient Shortest-Path Computation. *ACM Journal of Experimental Algorithmics*, 10(1.3):1–30, 2005.

## A Proofs

**Proof of Lemma 4.4:** Let $P(s,t)$ be the shortest path from $s$ to $t$. Let $R$ be the first region, namely, the region of the node in $\mathcal{T}$ that is closest to the root, such that its path separator $\text{SEP}(R)$ intersects with $P(s,t)$. Let $w \in \text{SEP}(R) \cap P(s,t)$ be the first such vertex (closest to $s$).

Let $R_1(s)$ be the first region (closest to the root) such that $s$ belongs to its separator and let $R_i(s)$ be the parent region of $R_{i-1}(s)$ until $R_\ell(s) = R$. Note that the path $P(s,t)$ must intersect with $\text{FRAME}(R_i(s))$ for every $1 \le i \le \ell$.

Let $z_1 = s$ and $z_\ell = w$. Let $z_i$ for $1 < i < \ell$ be the first vertex (closest to $s$) in $\text{FRAME}(R_i(s)) \cap P(s,t)$. Note that as $z_i$ belongs to the frame of $R_i(s)$ then by definition it belongs to $\text{FRAME}(R_{i+1}(s)) \cup \text{TAIL}(R_{i+1}(s))$.

Let $j$ be the index such that $2^{j-2} \le \mathbf{dist}(s,w) \le 2^{j-1}$.

Let $x_i \in N_j(R_i(s))$ be the closest net-point to $z_i$. By Lemma 4.3, $\mathbf{dist}(x_i, z_i, R_{input}) \le \epsilon_1 2^j/(16M)$. Using the assumption of $M$ bounded stretch we have $\mathbf{dist}(x_i, z_i, R) \le \epsilon_1 2^j/16$.

We show that the entire path $P(s, z_{i+1})$ belongs to $R_{i+1}(s)$. Hence, any subpath of $P(s, z_{i+1})$ is also fully contained in $R_{i+1}(s)$. In particular, $P(z_i, z_{i+1})$ is contained in $R_{i+1}(s)$. For $i = \ell - 1$, the entire path $P(s,t)$ belongs to $R_\ell(s) = R$ (since none on the vertices on $P(s,t)$ appeared in previous ancestors). Hence, any subpath of $P(s,t)$ is also fully contained in $R_\ell(s)$. In particular, $P(s, z_\ell)$ is contained in $R_\ell(s)$. Assume now that for every $i' > i$, $P(s, z_{i'+1})$ is contained in $R_{i'+1}(s)$ and consider $i$. By induction hypothesis, the path $P(s, z_{i+2})$ is contained in $R_{i+2}(s)$. Recall that $z_{i+1}$ is the closest vertex to $s$ in $\text{FRAME}(R_{i+1}(s)) \cap P(s,t)$. Namely, the vertices $V(P(s, z_{i+1})) \setminus \{z_{i+1}\}$ do not belong to $\text{SEP}(R_{i+1})$ (or any previous separators). It is not hard to verify now that the path $P(s, z_{i+1})$ belongs to $R_i$, as needed ($s$ belongs to $R_i$ and this entire path (except $z_{i+1}$) did not belong to previous separators, so all $V(P(s, z_{i+1})) \setminus \{z_{i+1}\}$ belong to the cluster of $R_i$ and by definition $z_{i+1}$ belongs to the region $R_i$).

Therefore we have that

$$\begin{aligned}
\mathbf{dist}(x_i, x_{i+1}, R_{i+1}) &\le \\
&\quad \mathbf{dist}(x_i, z_i, R_{i+1}) + \\
&\quad \mathbf{dist}(z_i, z_{i+1}, R_{i+1}) + \\
&\quad \mathbf{dist}(z_{i+1}, x_{i+1}, R_{i+1}) \\
&\le \frac{\epsilon_1 2^j}{16} + \mathbf{dist}(z_i, z_{i+1}) + \frac{\epsilon_1 2^j}{16} \\
&= \mathbf{dist}(z_i, z_{i+1}) + \frac{\epsilon_1 2^j}{8}.
\end{aligned}$$

Hence

$$\begin{aligned}
\mathbf{dist}(x_i, x_{i+1}, R_{i+1}) &\le \mathbf{dist}(z_i, z_{i+1}) + \epsilon_1 2^j/8 \\
&< 2^{j-1} + \epsilon_1 2^j/8 \\
&< 2^j.
\end{aligned}$$

Using the fact that $\epsilon_1 << 1$.

Hence $x_i$ and $x_{i+1}$ are connected by an edge in $G_j(R_{i+1})$. Moreover $\omega(x_i, x_{i+1}, R_{i+1}) \le (1 + \epsilon_2(2(\log n + \log D)))\mathbf{dist}(x_i, x_{i+1})$. In addition, note that $\mathbf{dist}(s, x_1, R_1(s)) \le \epsilon_1 2^j/16$ and thus $s$ and $x_1$ are connected by an edge in $G(R_1)$.

Denote

$$\beta = (1 + \epsilon_2(2(\log n + \log D)))$$

It follows that

$$\begin{aligned}
& \mathbf{dist}(s, x_\ell, H) \\
\le\ & \omega(s, x_1) + \sum \omega(x_i, x_{i+1}, R_{i+1}) \\
\le\ & \omega(s, x_1) + \sum \beta \mathbf{dist}(x_i, x_{i+1}, R_{i+1}) \\
\le\ & \omega(s, x_1) + \sum \beta(\mathbf{dist}(z_i, z_{i+1} + \mathbf{dist}(z_i, x_i) + \mathbf{dist}(z_{i+1}, x_{i+1}))) \\
\le\ & \omega(s, x_1) + \beta \sum (\mathbf{dist}(z_i, z_{i+1}) + \mathbf{dist}(z_i, x_i) + \mathbf{dist}(z_{i+1}, x_{i+1})) \\
=\ & \omega(s, x_1) + \beta(\mathbf{dist}(s, w) + 2 \sum \mathbf{dist}(z_i, x_i)) \\
\le\ & \omega(s, x_1) + \beta(\mathbf{dist}(s, w) + (\log n - 1)\epsilon_1 2^j/8) \\
\le\ & \epsilon_1 2^j/16 + \beta(\mathbf{dist}(s, w) + (\log n - 1)\epsilon_1 2^j/8) \\
\le\ & (\mathbf{dist}(s, w) + \log n \epsilon_1 2^j/8)\beta \\
\le\ & (\mathbf{dist}(s, w) + \log n \mathbf{dist}(s, w)\epsilon_1/4)(1 + \epsilon_2(2(\log n + \log D))) \\
=\ & \mathbf{dist}(s, w)(1 + \log n \epsilon_1/4)(1 + \epsilon_2(2(\log n + \log D))) \\
\le\ & \mathbf{dist}(s, w)(1 + \epsilon/4)(1 + \epsilon/4) \\
<\ & \mathbf{dist}(s, w)(1 + \epsilon),
\end{aligned}$$

where the second last inequality follows from the facts that $\epsilon_1 < \epsilon_2$ and $\epsilon_2 = \epsilon/(4(2\log n + \log D))$. Also recall that: $\epsilon_2 = \epsilon/(4(2\log n + \log D))$ and $\epsilon_1 = \epsilon_2/10$. Since $\epsilon < 1$ then both $\epsilon_1$ and $\epsilon_2 < 1/logn$ (so definitely less than 1).

Similarly, we can show that $\mathbf{dist}(w, t, H) \le (1 + \epsilon)\mathbf{dist}(w, t)$.

Hence $\mathbf{dist}(s, t, H) \le \mathbf{dist}(s, w, H) + \mathbf{dist}(w, t, H) \le (1 + \epsilon)(\mathbf{dist}(s, w) + \mathbf{dist}(w, t)) = (1 + \epsilon)\mathbf{dist}(s, t)$. ∎

**Proof of Lemma 4.5:** Recall that the set of vertices of graph $H$ is the union of $\mathcal{RN}_1(x, R, i)$ for every $x \in \{s, t\}$, $R \in \text{REGIONS}(\text{ANCESTORS}[\text{HOME}(x)])$ and index $1 \le i \le \log D$.

By Lemma 4.1 the size of $\mathcal{RN}_1(x, R, i)$ is $O(\epsilon_1^{-1} M\alpha)$. The number of regions in

REGIONS(ANCESTORS[HOME($x$)]) is $O(\log n)$ and there are $\log D$ indices $i$. Therefore the number of vertices in $H$ is $O(\epsilon_1^{-1} M\alpha \log n \log D)$.

Consider a vertex $x \in N_i(R)$, note that the set of edges of $x$ in $G_i(R)$ is a subset of $\mathcal{RN}_2(x,R,i)$. Thus the number of edges added to $H$ due to $x$ and $G_i(R)$ is $O(\epsilon_1^{-1} M\alpha)$. The number of regions in REGIONS(ANCESTORS[HOME($x$)]) is $O(\log n)$ and there are $\log D$ indices $i$.

We get that the number of edges in $H$ is $O(\epsilon_1^{-2} M^2\alpha^2 \log^2 n \log^2 D)$. Thus constructing and invoking a shortest path algorithm in $H$ is also $O(\epsilon_1^{-2} M^2\alpha^2 \log^2 n \log^2 D)$ time. ∎

**Proof of Lemma 4.6:** First, we show that all edges that were not deleted in the update phase satisfy the claim. Consider an edge $(x_1,x_2) \in G_i(R)$ for some region $R$ and index $1 \le i \le \log D$.

If $R$ does not contain the edge $(s,t)$ then clearly the claim holds by the assumption that the claim held before the update.

So assume $R \in \mathrm{RR}((s,t))$. Note that if either $x_1 \notin \mathcal{RN}_2(s,R,i) \cap \mathcal{RN}_2(t,R,i)$ or $x_2 \notin \mathcal{RN}_2(s,R,i) \cap \mathcal{RN}_2(t,R,i)$ then the path $(x_1,x_2)$ represents cannot contain the edge $(s,t)$. To see this, assume w.l.o.g. that $x_1 \notin \mathcal{RN}_2(s,R,i)$. It follows that $\mathbf{dist}(x_1,s,R_{input}) > 2^i$. Hence $\mathbf{dist}(x_1,s,R) \ge \mathbf{dist}(x_1,s,R_{input}) > 2^i$. As $\mathbf{dist}(x_1,x_2,R) \le 2^i$, it follows that $(s,t) \notin P(x_1,x_2,R)$. So the edge $(x_1,x_2)$ satisfies the claim.

If $x_1,x_2 \in \mathcal{RN}_2(s,R,i) \cap \mathcal{RN}_2(t,R,i)$ but $\omega(x_1,x_2,G_i(R)) < 2^{i-1}$. Note that $\mathbf{dist}(x_1,x_2) < \omega(x_1,x_2,G_i(R)) < 2^{i-1}$. Hence either the edge was deleted for $i-1$ or it is safe.

So we are left to show correctness for the edges that were deleted in the update phase.

We show correctness by induction on the level of the region (from the leafs to the root).

Assume $R$ is a leaf in $\mathcal{T}$. We claim that $\omega(x_1,x_2,G_i(R)) = \mathbf{dist}(x_1,x_2,R)$ for every index $i$ and vertices $x_1,x_2 \in N_i(R)$. To see this, consider $x_1,x_2 \in N_i(R)$. If $x_1$ and $x_2$ are related then note that $\mathbf{dist}(x_1,x_2,R) = \mathbf{dist}(x_1,x_2,T)$ and as the algorithm adds an edge $(x_1,x_2)$ to $H$ of weight $\mathbf{dist}(x_1,x_2,T)$ then we get $\omega(x_1,x_2,G_i(R)) = \mathbf{dist}(x_1,x_2)$. Recall that the region $R$ consists only of the frame FRAME($R$), namely, $O(\log n)$ shortest paths in $T$ connected by apices. Thus the path $P(x_1,x_2,R)$ is of the form $P(x_1 = z_0,z_1,T) \circ (z_1,z_2) \circ P(z_2,z_3,T) ... \circ P(z_r,x_2 = z_{r+1},T)$, where the vertices $z_i$ for $1 \le i \le r$ are apices. For every edge $(z_i,z_{i+1})$ we have $\mathbf{dist}(z_i,z_{i+1},H) = \omega(z_i,z_{i+1})$ and for every path $P(z_i,z_{i+1},T)$ we also have $\mathbf{dist}(z_i,z_{i+1},H) = P(z_i,z_{i+1},T)$. We conclude that $\mathbf{dist}(x_1,x_2,H) = \mathbf{dist}(x_1,x_2,R)$, as required.

Assume correctness for all regions $R'$ such that LEVEL($R'$) $\ge \ell+1$ and consider region $R$ such that LEVEL($R$) $= \ell$. We prove by induction on $i$ that for all pair of vertices $x_1,x_2 \in N_i(R)$: $\mathbf{dist}(x_1,x_2,H) \le (1+\epsilon_2(2(\log n - \mathrm{LEVEL}(R)) + i))\mathbf{dist}(x_1,x_2,R)$

For $i=0$, $\mathbf{dist}(x_1,x_2,R) \le 1$, namely $(x_1,x_2) \in E(G)$. Recall that the edge $(x_1,x_2)$ is added to $H$ and we get $\mathbf{dist}(x_1,x_2,H) = \mathbf{dist}(x_1,x_2)$.

Assume correctness for all $x_1',x_2' \in N_j(R)$ such that $j \le i-1$, and consider net-points $x_1,x_2 \in N_i(R)$ such that $\mathbf{dist}(x_1,x_2,R) \le 2^i$.

Let $R_1$ and $R_2$ be the two children regions of $R$. Let $P(x_1,x_2,R)$ be the shortest path from $x_1$ to $x_2$ in $R$.

If $\mathbf{dist}(x_1,x_2,R) \le 2^{i-1}$, then we show that the claim follows by the induction hypothesis on $i-1$. To see this, note that as $N_i(R) \subseteq N_{i-1}(R)$ we have $x_1,x_2 \in N_{i-1}(R)$.

So assume $2^{i-1} \le \mathbf{dist}(x_1,x_2,R) \le 2^i$.

We now show how to partition the path $P(x_1,x_2,R)$ into a constant number of subpaths such that we can apply one of the induction hypothesis (either the induction hypothesis on the level of the regions or on the level of the net-points). By concatenating these subpaths we derive the correctness on the edge $(x_1,x_2)$.

Let $z_1$ be the last vertex on $P(x_1,x_2,R)$ (closest to $x_2$) such that $\mathbf{dist}(x_1,z_1,R) \le 2^i/3$ that belongs to FRAME($R_1$) $\cup$ FRAME($R_2$) (note that such a vertex exists as $x_1 \in$ FRAME($R_1$) $\cup$ FRAME($R_2$)). Let $z_2$ be the next vertex in $P(x_1,x_2,R)$ after $z_1$ that belongs to FRAME($R_1$) $\cup$ FRAME($R_2$).

Let $z_3$ be the last vertex on $P(z_2,x_2,R)$ such that $\mathbf{dist}(z_2,z_3,R) \le 2^i/3$ that belongs to FRAME($R_1$) $\cup$ FRAME($R_2$). Let $z_4$ be the next vertex in $P(x_1,x_2,R)$ after $z_3$ (or $z_3$ in case $z_3 = x_2$) that belongs to FRAME($R_1$) $\cup$ FRAME($R_2$).

By Lemma 4.3, there exists a net-point $w_1$ in $N_{i+1}(R)$ such that $\mathbf{dist}(z_1,w_1,R_{input}) \le \epsilon_1 2^{i+1}/(16M)$. Recall that the distances in $R$ are at most $M$ times the distances in $R_{input}$. Hence $\mathbf{dist}(z_1,w_1,R) \le \epsilon_1 2^{i+1}/16$.

As $N_{i+1}(R) \subseteq N_i(R) \subseteq N_{i-1}(R)$, we also have $x_1,w_1 \in N_{i-1}(R)$.

Note that $\mathbf{dist}(x_1,w_1,R) \le \mathbf{dist}(x_1,z_1,R) + \mathbf{dist}(z_1,w_1,R) \le 2^i/3 + 2^{i+1}/16 < 2^{i-1}$.

In addition, we claim that $w_1 \in V(H)$. To see this, note that $\mathbf{dist}(s,w_1,R) \le \mathbf{dist}(s,x_1,R) + \mathbf{dist}(x_1,w_1,R) \le \mathbf{dist}(s,x_1,R) + \mathbf{dist}(x_1,w,R) \le 2^i + 2^{i-1} \le 2^{i+1}$. Thus by definition $w_1 \in \mathcal{RN}_1(s,R,i-1)$.

Hence by induction hypothesis on $i-1$ we get that

$$\begin{aligned}
& \mathbf{dist}(x_1, w_1, H) \\
\leq\ & (1+\epsilon_2(2(\log n - \textsc{Level}(R)) + i - 1))\mathbf{dist}(x_1, w_1, R) \\
\leq\ & (1+\epsilon_2(2(\log n - \textsc{Level}(R)) + i - 1))(\mathbf{dist}(x_1, z_1, R) \\
+\ & \mathbf{dist}(z_1, w_1, R)) \\
\leq\ & (1+\epsilon_2(2(\log n - \textsc{Level}(R)) + i - 1))\mathbf{dist}(x_1, z_1, R) \\
+\ & 2\mathbf{dist}(z_1, w_1, R) \\
\leq\ & (1+\epsilon_2(2(\log n - \textsc{Level}(R)) + i - 1))\mathbf{dist}(x_1, z_1, R) \\
+\ & \epsilon_1\mathbf{dist}(x_1, x_2, R).
\end{aligned}$$

Let $w_2 \in N_{i+1}(R)$ be the net-point from Lemma 4.3, such that $\mathbf{dist}(z_2, w_2, R_{input}) \leq \epsilon_1 2^{i+1}/(16M)$.

Note that the path $P(z_1, z_2, R)$ belongs to either $R_1$ or $R_2$. Assume w.l.o.g that $P(z_1, z_2, R) \subseteq R_1$. Note also that $P(z_2, w_2, T) \subseteq R_1$ and that $P(z_1, w_1, T) \subseteq R_1$.

It follows that $\mathbf{dist}(w_1, w_2, R_1) \leq \mathbf{dist}(w_1, z_1, R_1) + \mathbf{dist}(z_1, z_2, R_1) + \mathbf{dist}(z_2, w_2, R_1) \leq \mathbf{dist}(z_1, z_2, R_1) + \epsilon_1 2^{i+1}/8$.

Similarly to the analysis above showing that $w_1 \in V(H)$, one can show that also $w_2 \in V(H)$.

We have $\mathbf{dist}(w_1, w_2, H) \leq \omega(w_1, w_2, R_1) \leq (1+\epsilon_2(2(\log n - \textsc{Level}(R_1) - 1) + i + 1))(\mathbf{dist}(z_1, z_2, R_1) + \epsilon_1 2^{i+1}/8) \leq (1+\epsilon_2(2(\log n - \textsc{Level}(R_1) - 1) + i + 1))(\mathbf{dist}(z_1, z_2, R_1) + \epsilon_1\mathbf{dist}(x_1, x_2, R)/2) \leq (1+\epsilon_2(2(\log n - \textsc{Level}(R_1) - 1) + i + 1))\mathbf{dist}(z_1, z_2, R_1) + \epsilon_1\mathbf{dist}(x_1, x_2, R) = (1+\epsilon_2(2(\log n - \textsc{Level}(R) - 1) + i - 1))\mathbf{dist}(z_1, z_2, R_1) + \epsilon_1\mathbf{dist}(x_1, x_2, R)$.

Let $w_3 \in N_{i+1}(R)$ be the net-point from Lemma 4.3, such that $\mathbf{dist}(z_3, w_3, R_{input}) \leq \epsilon_1 2^{i+1}/(16M)$. And let $w_4 \in N_{i+1}(R)$ be the net-point from Lemma 4.3, such that $\mathbf{dist}(z_4, w_4, R_{input}) \leq \epsilon_1 2^{i+1}/(16M)$.

Similarly, we can show that $w_3, w_4 \in V(H)$ and that $\mathbf{dist}(w_2, w_3, H) \leq (1+\epsilon_2(\log n - \textsc{Level}(R) + i - 1))\mathbf{dist}(z_2, z_3, R) + \epsilon_1\mathbf{dist}(x_1, x_2, R)$ and $\omega(w_3, w_4, R) \leq (1+\epsilon_2(\log n - \textsc{Level}(R) + i - 1))\mathbf{dist}(z_3, z_4, R) + \epsilon_1\mathbf{dist}(x_1, x_2, R)$ and $\omega(w_4, x_2, R) \leq (1+\epsilon_2(\log n - \textsc{Level}(R) + i - 1))\mathbf{dist}(z_4, x_2, R) + \epsilon_1\mathbf{dist}(x_1, x_2, R)$.

It follows that $\mathbf{dist}(x_1, x_2, H) \leq (1+\epsilon_2(\log n - \textsc{Level}(R) + i - 1))(\mathbf{dist}(x_1, x_2, R) + 5\epsilon_1 2^i) \leq (1+\epsilon_2(\log n - \textsc{Level}(R) + i))\mathbf{dist}(x_1, x_2, R)$, where the last inequality holds for every $\epsilon_1 \leq \epsilon_2/10$ and $\epsilon \leq 1$ (recall that $\epsilon_1 = \epsilon_2/10$ and $\epsilon_2 = \epsilon/(4(2\log n + \log D)) << 1$). ∎