

Exploiting Hopsets: Improved Distance Oracles for Graphs of Constant Highway Dimension and Beyond

Siddharth Gupta

Ben-Gurion University of the Negev, Israel
siddhart@post.bgu.ac.il

Adrian Kosowski

Inria, Paris, France
adrian.kosowski@inria.fr

Laurent Viennot

Inria, Paris, France
Laurent.Viennot@inria.fr

Abstract

For fixed $h \geq 2$, we consider the task of adding to a graph G a set of weighted shortcut edges on the same vertex set, such that the length of a shortest h -hop path between any pair of vertices in the augmented graph is exactly the same as the original distance between these vertices in G . A set of shortcut edges with this property is called an *exact h -hopset* and may be applied in processing distance queries on graph G . In particular, a 2-hopset directly corresponds to a distributed distance oracle known as a *hub labeling*. In this work, we explore centralized distance oracles based on 3-hopsets and display their advantages in several practical scenarios. In particular, for graphs of constant highway dimension, and more generally for graphs of constant skeleton dimension, we show that 3-hopsets require *exponentially* fewer shortcuts per node than any previously described distance oracle, and also offer a speedup in query time when compared to simple oracles based on a direct application of 2-hopsets. Finally, we consider the problem of computing minimum-size h -hopset (for any $h \geq 2$) for a given graph G , showing a polylogarithmic-factor approximation for the case of unique shortest path graphs. When $h = 3$, for a given bound on the space used by the distance oracle, we provide a construction of hopset achieving polylog approximation both for space and query time compared to the optimal 3-hopset oracle given the space bound.

2012 ACM Subject Classification Theory of computation → Shortest paths; Theory of computation → Graph algorithms analysis; Theory of computation → Data structures design and analysis

Keywords and phrases Hopsets, Distance Oracles, Graph Algorithms, Data Structures

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.143

Category Track C: Foundations of Networks and Multi-Agent Systems: Models, Algorithms and Information Management

Related Version A full version of the paper is available at <https://arxiv.org/abs/1803.06977>.

Funding *Siddharth Gupta*: Supported in part by the Zuckerman STEM Leadership Program.

Adrian Kosowski: Supported by ANR project DESCARTES (ANR-16-CE40-0023).

Laurent Viennot: Supported by ANR project DISTANCIA (ANR-17-CE40-0015) and ANR project MULTIMOD (ANR-17-CE22-0016).

1 Introduction

An exact h -hopset for a weighted graph G is a weighted edge set, whose addition to the graph guarantees that every pair of vertices has a path between them with at most h edges (*hops*) and whose length is exactly the length of shortest path between the vertices.



© Siddharth Gupta, Adrian Kosowski, and Laurent Viennot;
licensed under Creative Commons License CC-BY
46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 143; pp. 143:1–143:15



Leibniz International Proceedings in Informatics
LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



The concept of a hopset was first explicitly described by Cohen [19] in its *approximate* setting, in which the length of h -hop path between a pair of vertices in the hopset should approximate the length of the shortest path in G . Hopsets were introduced in the context of parallel computation of approximate shortest paths. In this paper, we study hopsets in their exact version, with the general objective of optimizing exact shortest path queries.

Data structures which allow for querying distance between any pair of vertices of a graph have been intensively studied under the name of *distance oracles*. The efficiency of an exact distance oracle is typically measured by the interplay between the *space* requirement of the representation of the data structure and its *decoding time*. It is a well-established empirical fact that many real-world networks admit efficient (i.e., low-space and fast) distance oracles [6, 22]. A key example here concerns transportation networks, and specifically road networks, which are empirically known [34, 32, 5] to be augmentable by carefully tailored sets of shortcut edges, allowing for shortest-path computation. These sets of shortcuts may be hopsets (as is the case for the hub-labeling approach which effectively implements a 2-hopset), but may also be considered in some related (and frequently more involved) framework, such as contraction hierarchies [31] or transit-node routing [11].

An interesting theoretical insight due to Abraham et al. [3, 4, 5] provides theoretical bounds on the number of shortcuts required in all of the above-mentioned frameworks. They introduce a parameter describing the structure of shortest paths within ball neighborhoods of a graph, called *highway dimension* \tilde{h} . They also express the number of shortcuts that need to be added for each node so as to achieve shortest-path queries in a graph of n nodes with weighted diameter D as a polynomial of \tilde{h} , $\log n$, and $\log D$; this approach has been extended in subsequent work [2, 37]. The value of \tilde{h} is known to be small in practice (e.g., typically $\tilde{h} < 100$ for continental-sized road networks [4]), and does indeed appear to be inherently linked to the size of the required shortcut sets. In fact, empirical tests have suggested that the (average) number of necessary shortcuts per node is in fact very close to \tilde{h} , laying open the question of whether the additional dependence of the number of shortcuts on logarithmic factors in n and D may be an artifact of the theoretical analysis of the oracles, which for each node require a separate shortcut for every “scale” of distance.

1.1 Results and Organization of the Paper

Our main result is to provide strong evidence that the dependence of the number of shortcuts on such logarithmic factors in n and D is indeed not essential, and we design a simple distance oracle based on a 3-hopset in which the number of shortcuts per node depends only on \tilde{h} , $\log \log n$, and the logarithm of the average edge length. This result is in fact shown in the framework of a strictly broader class of graphs, namely, graphs with a bounded value of a parameter known as *skeleton dimension* k ($k \leq \tilde{h}$), describing the width of the shortest-path tree of a node after pruning all branches at a constant fraction α of their depth. Considering various ranges of fraction α for increasing distance ranges was a novel key step for improving over [37, 36] from a 2-hopset construction to a 3-hopset construction.

From a general perspective, our connection between h -hopsets and distance oracles is original and offers new perspectives for studying the trade-off between size and query time of distance oracles. To exemplify this, we provide a construction of h -hopsets for graphs of treewidth t following a classical approach in pre-processing product queries on trees [7, 16]. For 3-hopsets, we obtain a distance oracle with quadratic dependency in t which improves over the construction of [15] (which has cubic dependency) for $t = \omega(\log^2 \log n)$. The space and time-bounds of oracles based on 3-hopsets are presented in Table 1, and compared with the corresponding parameters of oracles based on 2-hopsets. For the case of constant skeleton dimension or constant treewidth, we remark that using a 3-hopset instead of a 2-hopset reduces the number of shortcuts per node from $O(\log n)$ to $O(\log \log n)$ while achieving a query time of $O(\log^2 \log n)$.

■ **Table 1** Comparison of distance oracles based on 2-hopsets (hub labeling [19, 28, 37]) and 3-hopsets (this paper). Size represents the number of shortcut edges in the hopset, i.e., the number of $O(\log n)$ -bitsize words when measuring oracle size. The main results concern skeleton dimension and are stated in simplified form, assuming average edge length at most $O(\text{poly } \log n)$, with expected query times given for both types of oracles.

Distance oracle	Treewidth t		Skeleton dimension k	
	Size	Time	Size	Time
2-hopset (hubs):	$n \cdot O(t \log n)$	$O(t + \log \log n)$	$n \cdot O(k \log n)$	$O(k \log n)$
3-hopset:	$n \cdot O(t \log \log n)$	$O(t^2 \log^2 \log n)$	$n \cdot O(k \log k \log \log n)$	$O(k^2 \log^2 k \log^2 \log n)$

A classical assumption (applied, e.g., in almost all literature on transportation networks) resides in the uniqueness of shortest paths. It can be made without loss of generality by slightly perturbing the weights of the edges or by using appropriate tie break rules. In this context of *unique shortest path graph* (USP) graphs where there is a unique shortest path P_{uv} between any two nodes u and v , we propose an LP-based approximation algorithm for constructing h -hopsets with size within a polylog factor from optimal. Our construction can be seen as a non-trivial generalization of the prehub labeling introduced in [9] from 2 to more hops. In the case $h = 3$, we further extend our approach to provide an algorithm which constructs distance oracles in USP graphs based on 3-hopsets, with (approximate) optimality guarantees on size and query time. The form of guarantees we obtain is again novel: for a given size bound \mathcal{S} of 3-hopset based oracle, we construct an oracle with size larger than \mathcal{S} by at most a polylog factor which has average query time within a polylog factor of the performance achieved by the best oracle with size \mathcal{S} .

The rest of the paper is organized as follows. In Section 2, we introduce the necessary notions related to h -hopset and give a general approach for how a h -hopset can be used as a distance oracle, focusing on the special case of $h = 3$. In Section 3, we provide our first main result, using 3-hopsets to obtain improved (smaller and faster) distance oracles in graphs with bounded skeleton dimension. In Section 4, we present our second main result about approximating h -hopsets and constructing 3-hopset based oracles in USP graphs. Finally, Section 5, we show how to construct efficient h -hopsets and 3-hopset based oracles for bounded treewidth graphs. We provide full details of omitted and sketched proofs in the full version [33].

Our work is presented in the context of weighted undirected graphs, but all results can easily be extended to weighted directed graphs.

1.2 Other Related Work

Hopsets. Exact hopsets were implicitly constructed in the context of single-source shortest paths parallel computation [43, 35, 18, 40]. Such works study the work versus time trade-offs of such computation. Cohen [19] explicitly introduced the notion of (h, ϵ) -hopset of G as set H of weighted edges such that paths of at most h hops in $G \cup H$ have length within $(1 + \epsilon)$ of the corresponding shortest path in G . The parameter h is called the *hopbound*. For any graph G and $\epsilon, \epsilon' > 0$, she proposed a construction of $(O(\text{poly } \log n), \epsilon)$ -hopset of G with size $O(n^{1+\epsilon'})$. More recently, Elkin et al. [24] proposed the construction of $(O(\epsilon^{-1} \log \kappa)^{\log \kappa}, \epsilon)$ -hopset with $O(n^{1+1/\kappa} \log n \log \kappa)$ edges for any $\epsilon > 0$ and integral $\kappa \geq 1$. Abboud et al. [1] recently showed the optimality of the Elkin et al. [24] result. In particular, they showed that for any $\delta > 0$ and integer k , any hopset of size less than $n^{1+\frac{1}{2^{k+1}-1}-\delta}$ must have hop bound $h = \Omega(c_k/\epsilon^{k+1})$, where c_k is a constant depending only on k . The linear size case was then improved in [25]. As far as we know, exact hopsets (with $\epsilon = 0$) have not been explicitly studied. However, they are related to the following well studied notion.

Hopsets vs. TC-spanners. In directed graphs, a hopset can be seen as a special case of an *h-transitive-closure spanner* (*h-TC-spanner*). Hopsets and TC-spanners are fundamental graph-theoretic objects and are widely used in various settings from distance oracles to pre-processing for range queries in sequential or parallel setting or even in property testing. The concept of adding transitive arcs to a digraph in order to reduce its diameter was introduced by Thorup [41] in the context of parallel processing. Bhattacharyya et al. [12] defined an *h-TC-spanner* of an unweighted digraph G as a digraph H with same transitive closure as G and diameter at most h . They note that this is a central concept in a long line of work around pre-processing a tree for range queries [7, 16, 42]. A TC-spanner can also be defined as a spanner (for the classical spanner definition [38]) of the transitive closure of a graph that has bounded diameter. We will see that an exact h -hopset defines a h -TC-spanner but that the converse is not necessarily true. Bhattacharyya et al. [12] proposed a construction of h -TC-spanner of size $O(n \log n \lambda_h(n))$ for H -minor-free graphs (where λ_h denotes the h th-row inverse Ackermann function, cf. Section 5).

Exact Distance Oracles. A long line of research studies the interplay between data structure space and query decoding time. A lot of attention has been given to distance oracles for planar graphs [23, 10, 17, 14, 26, 21, 30], and it has recently been shown that a distance oracle with $O(n^{1.5})$ space and $O(\log n)$ query-time is possible [30]. In the context of weighted directed graphs with treewidth t , Chaudhuri and Zaroliagis [15] propose a distance oracle using $O(t^2 n \lambda_h(n))$ space and $O(t^3 h + \lambda_h(n))$ query time for integral $h > 1$ where λ_h is the h th-row inverse Ackermann function (as defined in Subsection 5). In the context of unweighted graphs with treewidth t , Farzan and Kamali [27] obtain distance oracles with $O(t^3 \log^3 t)$ query time using optimal space (within low order terms). This construction heavily relies on the unweighted setting as exhaustive look-up tables are constructed for handling graphs with polylogarithmic size.

Distance Labelings and 2-Hopsets. The distance labeling problem is a special case of a distributed distance oracle, and consists of assigning labels to the nodes of a graph such that the distance between two nodes s and t can be computed from the labels of s and t (see, e.g., [28]).

The notion of 2-hopset studied in this work coincides with the special case of two-hop distance labeling (also called *hub-labeling*), where labels are constructed from hub sets: in hub-labeling, a small hub set $S(u) \subseteq V(G)$ is assigned to each node of a graph G such that for any pair u, v of nodes, the intersection of hub sets $S(u) \cap S(v)$ contains a node on a shortest $u - v$ path. Such a construction is formally proposed in [20] and is implicitly introduced by Gavaille et al. [28] and applied to graphs of treewidth t with labels of $O(t \log n)$ size and allows to answer distance queries in $O(t \log n)$ time; the hub sets have a hierarchical structure, which allows for an improvement of query time to $O(t \log \log n)$ time by a binary search over levels. Hub labelings are the currently best known distance labelings for sparse graphs, achieving sublinear node label size [8, 29], and may also be used to provide a 2-additive-approximation for distance labeling in general graphs using sublinear-space labels [29].

In graphs of bounded highway dimension, hub labels were among the first identified distance oracles to provide label size and query time polynomial in the highway dimension and polylogarithmic in other graph parameters [5]. This result was then extended to the more general class of graphs with bounded skeleton dimension [37, 36].

Hub sets with near to optimal size can be constructed in polynomial time. A greedy set cover-type $O(\log n)$ -approximation algorithm (with respect to average size of a hub set) was proposed by Cohen et al. [20]. For the case of USP graphs, this approximation ratio was improved by Angelidakis et al. [9] to the logarithm of the graph hop-diameter D_H , i.e., the maximum number of hops of a shortest path in G , showing an approximation gap between USP and non-USP graphs.

2 Preliminaries

2.1 Definitions

We are given a weighted undirected graph $G = (V, E, \omega)$ where $\omega : E \rightarrow \mathbb{R}^+$ associates a weight with each edge of G . For a positive integer parameter h and a pair $u, v \in V$, the h -limited distance between u and v , denoted $d_G^h(u, v)$, is defined as the length of the shortest path from u to v that contains at most h edges (aka *hops*). The usual shortest path distance can be defined as $d_G(u, v) = d_G^{h-1}(u, v)$. For the sake of brevity, we often let uv denote the pair $\{u, v\}$ representing an edge from u to v .

► **Definition 1.** An (exact) h -hopset for a weighted graph G is a set of edges H such that $d_{G \cup H}^h(u, v) = d_G(u, v)$ for all u, v in $V(G)$ where $G \cup H = (V, E \cup H, w')$ is the graph augmented with edges of the hopset with weights $w'(u, v) = d_G(u, v)$ for $uv \in H$ and $w'(u, v) = w(u, v)$ for $uv \in E \setminus H$. The parameter h is called the hopbound of the hopset. Edges from set H are called shortcuts in G .

By convention, we will assume that all self-loops at nodes of V are included in H . Thus, $G \cup H$ is a graph whose h -th power in the $(\min, +)$ algebra on $n \times n$ matrices of edge weights corresponds to the transitive closure of the weight matrix of graph G .

Equivalently, a h -hopset can be defined as a set H of edges such that for any pair s, t , there exists a path P of at most h edges from s to t in $G \cup H$ and a shortest path Q from s to t in G such that all nodes of P belong to Q and appear in the same order. Note that a h -hopset is completely specified by its set H of edges as the associated weights are deduced from distances in the graph.

2.2 Using a Hopset as a Distance Oracle

Hopsets may be used to answer shortest-path queries in a graph $G = (V, E)$. In general, given a hopset H , the naïve way to approach a query for $d_G(u, v)$ for a given node pair u, v is to perform a bidirectional Dijkstra search in graph $G \cup H$ from this node pair, limited to a maximum of $\lceil h/2 \rceil$ hops distance from each of these nodes. We have, in particular for any pair $u, v \in V$:

$$d_G(u, v) = \min_{w \in V} (d_{G \cup H}^{\lceil h/2 \rceil}(u, w) + d_{G \cup H}^{\lfloor h/2 \rfloor}(v, w)).$$

Different optimizations of this technique are possible.

In this paper, we focus only on the time complexity of the case of $h = 3$, where we perform the following optimization of query execution. We represent set H as the union of two (not necessarily disjoint) sets of shortcuts, $H = H_1 \cup H_2$, where an edge belongs to H_1 if it is used as the first or third (last) hop on a shortest path in $G \cup H$, and it belongs to H_2 if it is used as the second hop on such a path. By convention, we assume that self-loops at nodes are added to H_1 , thus e.g. a 3-hop path between a pair of adjacent nodes in G is constructed by taking a self-loop from H_1 , the correct edge from $G \subseteq G \cup H_2$, and another self-loop from H_1 . (Note that we never directly use edges of G as first or last hops in the hopset; if such an edge is required for correctness of construction, it should be explicitly added to set H_1 .) We further apply an orientation to the shortcuts in H_1 , constructing a corresponding set of arcs \vec{H}_1 , such that, for any node pair $u, v \in V$, there exist $x, y \in V$ such that $(u, x) \in \vec{H}_1$, $\{x, y\} \in H_2$, $(v, y) \in \vec{H}_1$, and:

$$d_G(u, v) = d_G(u, x) + d_G(x, y) + d_G(y, v).$$

The orientation (w, z) of an arc in \vec{H}_1 indicates that edge $\{w, z\}$ can be used as the first edge of a 3-hop path from w or as the third edge of a 3-hop path to w . We note that $|\vec{H}_1| \leq |\vec{H}_1| \leq 2|H_1|$, since each shortcut from H_1 corresponds to at most a pair of symmetric

arcs in \vec{H}_1 . For a node $w \in V$, let $N_1(w) = |\{x \in V : (w, x) \in \vec{H}_1\}|$ represent the out-neighborhood of w in the graph (V, \vec{H}_1) . To perform shortest path queries on G , for each node w , we now store the list $\{(x, d_G(w, x)) : x \in N_1(w)\}$. We also store a hash map, mapping all node pairs $\{x, y\} \in H_2$ to the length of the respective link, $d_G(x, y)$. Now, we answer the distance query for a node pair $u, v \in G$ as follows:

$$d_G(u, v) = \min_{x \in N_1(u), y \in N_1(v) : \{x, y\} \in H_2} (d_G(u, x) + d_G(x, y) + d_G(y, v)).$$

Using the given data structures, the query is then processed using $|N_1(u)| \cdot |N_1(v)|$ hashmap look-ups, one for each pair $(x, y) \in N_1(u) \times N_1(v)$, i.e., in time $\mathcal{T}_{uv} = O(|N_1(u)| \cdot |N_1(v)|)$. Time \mathcal{T}_{uv} is simply referred to as the *query time* for the considered node pair in the 3-hopset oracle H . Assuming uniform query density over all node pairs, the *uniform-average query time* $\mathcal{T}(H)$ is given as: $\mathcal{T}(H) \equiv \mathbf{E}_{uv} \mathcal{T}_{uv} = O\left(\frac{1}{n^2} \left(\sum_{u \in V} |N_1(u)|\right)^2\right) = O(|H_1|^2/n^2)$. Thus, in the uniform density setting (which we refer to only in Section 4), the average time of processing a query is proportional to the square of the average degree of a node with respect to edge set H_1 .

The size of set H_2 affects only the size of the data structure required by the distance oracle, which is given as at most $\mathcal{S} = O(|E| + |H_1| + |H_2|)$ edges, with each edge represented using $O(\log n)$ bits.

In the 3-hopset distance oracles described in the following sections, we will confine ourselves to describing shortcut sets H_1 and H_2 , noting that the correct orientation \vec{H}_1 of H_1 will follow naturally from the details of the provided constructions.

3 Bounded Skeleton Dimension

A formal definition of the notion of skeleton dimension relies on the concept of the geometric realization of a graph, cf. [37]. The *geometric realization* \tilde{G} of G can be seen as the “continuous” graph where each edge is seen as infinitely many vertices of degree two with infinitely small edges, such that for any $uv \in E(G)$ and $t \in [0, 1]$, there is a node in \tilde{G} at distance $td_G(u, v)$ from u on edge uv . Given a shortest-path tree T_u of node u with length function $\ell : E(T_u) \rightarrow \mathbb{R}^+$, obtained as the union of shortest paths $\bigcup\{P_{uv} : v \in V(G)\}$, we treat it as directed from root to leaves and consider the geometric realization \tilde{T}_u of this directed graph. We define the *reach* of $v \in V(\tilde{T}_u)$ as the distance from v to the furthest leaf in its subtree of the directed tree \tilde{T}_u , i.e., $Reach_{\tilde{T}_u}(v) := \max_{x: v \in P_{ux}} d_{\tilde{T}_u}(v, x)$. For a given value $\alpha > 0$, we then define the *skeleton* T_u^* of T_u as the subtree of \tilde{T}_u induced by nodes with reach at least α times their distance from the root. More precisely, \tilde{T}_u^* is the subtree of \tilde{T}_u induced by $\{v \in V(\tilde{T}_u) \mid Reach_{\tilde{T}_u}(v) \geq \alpha d_{\tilde{T}_u}(u, v)\}$.

The α -*skeleton dimension* k_α of a graph G is now defined as the maximum width of the skeleton of a shortest path tree, taken over cuts at all possible distances from the root of the tree: $k = \max_{u \in V(G)} \max_{r > 0} |Cut_r(\tilde{T}_u^*)|$, where $Cut_r(\tilde{T}_u^*)$ is the set of nodes $v \in V(\tilde{T}_u^*)$ with $d_{\tilde{T}_u^*}(u, v) = r$. When $\alpha = \frac{1}{2}$, $k_{1/2}$ is simply called the *skeleton dimension* of G and we let $k = k_{1/2}$ denote it.

The definition was originally proposed with $\alpha = \frac{1}{2}$ (for comparison with highway dimension) in the context of USP graphs [37]. In the long version [36], the definition is extended to other choices of α with $0 < \alpha < 1$ and applies to any choice of shortest paths trees that pairwise agree on their paths (the path from u to v in T_u must be the reverse of the path from v to u in T_v). In the non-USP case, the skeleton dimension should be measured with the best choice of agreeing trees. In particular, if a small perturbation of the edge weights of G provides unique shortest path trees whose skeletons have width at most k_α , then the skeleton dimension of G is at most k_α . The α -skeleton dimension (with parameter α) was

introduced in [36] for the sake of a general definition with fixed α value in mind. We use it here in a novel manner with α tending towards 0 as we consider larger distances, enabling analysis of our new construction.

For the definition of the related concept of *highway dimension*, we refer the readers to [4]. We note that if a graph G has highway dimension h , then G has skeleton dimension $k = k_{1/2} \leq h$; hence, in all subsequent asymptotic analyses, upper bounds expressed in terms of skeleton dimension can be replaced by analogous bounds in terms of highway dimension.

3.1 Construction of the 3-Hopset

We denote by L_{\max} the maximum length of an edge in graph G . The construction of the 3-hopset H is obtained by taking a union of sets of shortcuts, each of which covers sets of node pairs within a given distance range. The first shortcut set H' covers all node pairs $u, v \in V$ with $d_G(u, v) \leq D'$, for some choice of distance bound D' , whereas each of the subsequent shortcut sets $H^{(D)}$ covers nodes at a distance in an exponentially increasing distance range, $d_G(u, v) \in [D, D^{1+\epsilon}]$, where $\epsilon := \frac{1}{2 \log_2 k}$ is suitably chosen. We then put:

$$H = H' \cup \bigcup_{i=1,2,\dots} H^{(D'^{i(1+\epsilon)})}.$$

Construction of set H' . We note that a construction of 2-hopsets for graphs of skeleton dimension k was performed in [37]. As a direct corollary of [37][Lem. 2, Cor. 1,2], given a distance bound D' , there exists a randomized polynomial-time construction of a set of shortcuts H' for graph G with the property that for any pair of nodes $u, v \in V$ with $d_G(u, v) \leq D'$, we have $d_{G \cup H'}^2 = d_G(u, v)$, such that $|H'| = O(nk \log D')$, and moreover for all $u \in V$, we have $\mathbf{E} \deg_{H'}(u) = O(k \log D')$ and $\deg_{H'}(u) = O(k \log D' \log \log n + \log n)$. We directly use set H' for the value $D' := L_{\max}^4 k^6 \log^{12} n$, considering H' as a 3-hopset for node pairs $u, v \in V$ with $d_G(u, v) \leq D'$. So we have:

$$|H'| = O(nk(\log \log n + \log L_{\max} + \log k)),$$

and for all $u \in V$:

$$\begin{aligned} \mathbf{E} \deg_{H'}(u) &= O(k(\log \log n + \log L_{\max} + \log k)), \\ \deg_{H'}(u) &= O(k \log \log n (\log \log n + \log L_{\max} + \log k) + \log n). \end{aligned}$$

We remark that, without loss of generality, in asymptotic analysis one may assume that $L_{\max} \leq kL$, where L is the *average* edge length in G , noting that edges longer than kL can be subdivided into edges of length at most kL by inserting additional vertices, increasing the number of nodes of the graph only by a multiplicative constant. Thus, in the above bounds, we can replace $(\log \log n + \log L_{\max} + \log k)$ by $(\log \log n + \log L + \log k)$.

Construction of set $H^{(D)}$. We now proceed to construct a 3-hopset for node pairs u, v with $d_G(u, v) \in [D, D^{1+\epsilon}]$. The construction of set $H^{(D)}$ is randomized and completely determined by assignment of real values $\rho(u) \in [0, 1]$ to each node $u \in V$, uniformly and independently at random. We condition all subsequent considerations on the event that all values ρ are distinct, i.e., $|\rho(V)| = |V|$, which holds with probability 1. ($\rho(V) = \{\rho(v) | v \in V\}$)

Now, hopset $H^{(D)}$ is defined as $H^{(D)} := H_1^{(D)} \cup H_2^{(D)}$, where following our usual notation, $H_1^{(D)}$ is the set of first and last hops, and $H_2^{(D)}$ is the set of middle hops.

Set of first and last hops. For $u \in V$, let $R^{(D)}(u)$ be the set of nodes which lie on a shortest path of length at least D which has one of its endpoints at u , and which have minimum value of ρ among all vertices on this path at distance in $[D/4, D/2]$ from u :

$$R^{(D)}(u) = \bigcup_{v \in V: d_G(u,v) \geq D} \left\{ \operatorname{argmin}_{r \in P_{uv}, d_G(u,r) \in [D/4, D/2]} \rho(r) \right\}.$$

We now put: $H_1^{(D)} := \{ur : u \in V, r \in R(u)\}$.

Set of middle hops. We put in $H_2^{(D)}$ links between all pairs of nodes which have a small value of ρ , satisfy the natural upper bound of $D^{1+\epsilon}$ on distance between them, and have sufficiently large reach, i.e., the shortest path between them can be extended by at least $D/4$:

$$H_2^{(D)} := \left\{ qr : q, r \in \bigcup_{u \in V} R^{(D)}(u) \wedge d_G(q,r) \leq D^{1+\epsilon} - D/2 \wedge (\exists v \in V r \in P_{qv} \wedge d_G(r,v) \geq D/4) \right\}.$$

The validity of H as a 3-hopset is immediate to verify from the construction: consider u, v and $i \geq 0$ such that $d_G(u, v) \in [D, D^{1+\epsilon}]$ with $D = D^{i(1+\epsilon)}$.

For $q = \operatorname{argmin}_{w \in P_{uv}, d_G(u,w) \in [D/4, D/2]} \rho(w)$ and $r = \operatorname{argmin}_{w \in P_{uv}, d_G(u,w) \in [D/2, 3D/4]} \rho(w)$, we then have $uq \in H_1^{(D)}$, $qr \in H_2^{(D)}$ and $vr \in H_1^{(D)}$, yielding a 3-hop shortest path from u to v . For $d_G(u, v) \leq D'$, H' contains a 2-hop shortest path from u to v .

3.2 Bound on 3-Hopset Size and Oracle Time

► **Lemma 2.** Fix $u \in V$ and $D > 0$. We have: $|R^{(D)}(u)| \leq k$.

From the above Lemma, it follows that for any $u \in V$, we have $\deg_{H_1^{(D)}}(u) \leq k$. Thus summing over all the $O(\log \log(nL_{\max})/\log(1+\epsilon)) = O(\log \log(nL_{\max}) \log k)$ levels of the construction, we successively obtain:

$$\deg_{H_1}(u) \leq \deg_{H'}(u) + k \cdot O(\log \log(nL_{\max}) \log k) = O(k \log \log n \log k (\log \log n + \log L) + \log n), \quad (1)$$

$$\mathbf{E} \deg_{H_1}(u) \leq \mathbf{E} \deg_{H'}(u) + k \cdot O(\log \log(nL_{\max}) \log k) = O(k \log k (\log \log n + \log L)), \quad (2)$$

$$|H_1| \leq |H'| + nk \cdot O(\log \log(nL_{\max}) \log k) = O(nk \log k (\log \log n + \log L)). \quad (3)$$

We now proceed to bound the size of the set H_2 of middle hopsets.

► **Lemma 3.** Fix $D \geq D'$. With probability $1 - O(1/n^2)$, it holds that for all $u \in V$ and for all $r \in R^{(D)}(u)$, we have $\rho(r) \leq L_{\max}/D$.

We now proceed under the assumption that the event from the claim of the Lemma holds. We now consider an arbitrary node $q \in R^{(D)}(u)$ for some $u \in V$, and look at $\deg_{H_2^{(D)}}(q)$. We now have that if $qr \in H_2^{(D)}$, then by the definition of $H_2^{(D)}$ and the above Lemma, the following conditions jointly hold:

■ $\rho(r) \leq L_{\max}/D$

■ $r \in \{w \in V : \exists v \in V D^{1+\epsilon} \geq d_G(q, v) \geq d_G(q, w) + D/4 \wedge P_{qv} \subseteq P_{qw}\} =: W(q)$.

We note that $W(q)$ is the subset of the vertex set of the shortest path tree of node q , pruned to contain only those paths which have reach at least $D/4$ at depth less than $D^{1+\epsilon}$. This tree has depth bounded by $D^{1+\epsilon}$, and width bounded by an α -skeleton dimension k_α (following [36]), with parameter $\alpha = \frac{D/4}{D^{1+\epsilon}} = D^{-\epsilon}/4$. Following [36][Section 6], k_α can be easily expressed using skeleton dimension $k = k_{1/2}$ as:

$$k_\alpha \leq k^{\lceil \log_2(1+1/\alpha) \rceil} < k^{1+\log_2(4D^\epsilon)} = k^3 D^{\epsilon \log_2 k}.$$

We then have $|W(q)| \leq D^{1+\epsilon} k_\alpha < k^3 D^{1+\epsilon(1+\log_2 k)}$. Moreover, by an easy concentration bound, we have that for all $q \in V$, $|\{r \in W(q) : \rho(r) \leq L_{\max}/D\}| = O(\log n) + \frac{2L_{\max}}{D}|W(q)|$, with probability $1 - O(1/n^2)$. It follows that with probability $1 - O(1/n^2)$, we have for all $q \in \bigcup_{u \in V} R^{(D)}(u)$:

$$\deg_{H_2^{(D)}}(q) \leq O(\log n) + \frac{2L_{\max}}{D}|W(q)| \leq O(\log n + L_{\max}k^3 D^{\epsilon \log_2 k}).$$

Noting that with probability $1 - O(1/n^2)$:

$$\left| \bigcup_{u \in V} R^{(D)}(u) \right| \leq |\{w \in V : \rho(w) \leq L_{\max}/D\}| \leq O(\log n + nL_{\max}/D)$$

we finally obtain that with probability $1 - O(1/n^2)$:

$$\begin{aligned} |H_2^{(D)}| &\leq O(\log n + nL_{\max}/D)O(\log n + L_{\max}k^3 D^{\epsilon \log_2 k}) = O(\log^2 n + nL_{\max}^2 k^3 D^{\epsilon \log_2 k - 1}) \\ &\leq O(nL_{\max}^2 k^3 D^{-1/2}) \leq O(nD'^{-1/4}) \leq O(n/\log^3 n), \end{aligned}$$

where in the last two transformations we use the fact that $\epsilon = \frac{1}{2 \log_2 k}$ and that $D \geq D' \geq L_{\max}^4 k^6 \log^{12} n$. Using a union bound and summing over all levels of the construction, we eventually obtain that with probability $1 - O(1/n)$:

$$|H_2| \leq O(n/\log_2 n). \quad (4)$$

Thus, the set of middle links is sparse and does not contribute to the asymptotic size of the overall representation of the 3-hopset.

Overall, considering a randomized construction which rejects random choices of ρ for which any of the considered w.h.p. events fail, by combining Eq. (1)–(4) with the hopset-based distance oracle framework described in the Preliminaries, we obtain the following Theorem.

► **Theorem 4.** *For a unique shortest path graph with skeleton dimension k and average link length $L \geq 1$, there exists a randomized construction of a 3-hopset distance oracle of size $|H| = O(nk \log k (\log \log n + \log L))$, which for an arbitrary queried node pair performs distance queries in expected time $O(k^2 \log^2 k (\log^2 \log n + \log^2 L))$ (where the expectation is taken over the randomized construction of the oracle), and in time $O(k^2 \log^2 k \log^2 \log n (\log^2 \log n + \log^2 L) + \log^2 n)$ with certainty.*

In particular, for graphs with constant-length edges and small skeleton dimension ($k = O(\log n)$), the 3-hopset has size $|H| = O(nk \log k \log \log n)$, with expected time of any query given as $O(k^2 \log^2 k \log^2 \log n)$.

4 LP-based Approximation Algorithm

In this section, we propose an Integer Linear Programming (ILP) formulation for h -hopsets with a minimum number of edges, which we then relax to a LP formulation. Whereas both formulations are applicable to the general case, we prove relations between them only for USP graphs.

4.1 ILP and LP Formulations

A necessary and sufficient condition for H to be a h -hopset for G is that for every pair of vertices s, t there exists a path $P_{st} = (s = v_0, v_1, \dots, v_{l_{st}} = t)$ in $G \cup H$ such that $l_{st} \leq h$ and in graph G there exists some shortest $s-t$ path passing through all of the vertices $v_0, \dots, v_{l_{st}}$, in the given order. For a fixed pair s, t , we consider the directed graph H^{st} with vertex set

$V \times \{0, \dots, h\} \equiv V_h$ (by convention, elements of V_h will be denoted compactly as v_i , where $v \in V$, $i \in \{0, \dots, h\}$) and with an arc set defined as follows. For $i \in \{0, \dots, h-1\}$, we add arc (u_i, v_{i+1}) to H^{st} if and only if $\{u, v\} \in G \cup H$ and u, v lie on some shortest $s-t$ path in the given order, i.e., if $d_G(s, u) + d_G(u, v) + d_G(v, t) = d_G(s, t)$. In particular, all arcs of the form (u_i, u_{i+1}) , for $u \in V$ on a $s-t$ shortest path, belong to H^{st} . Now, we have that H is a h -hopset for G if and only if there exists a path from s_0 to t_h in H^{st} . This is equivalent to saying that for all $s, t \in V$, the flow value from s_0 to t_h is at least 1 in H^{st} . Given graph G , we thus have the following ILP formulation for the minimum h -hopset problem, using indicator variables x_{uv} for $G \cup H$ (given as 1 if $\{u, v\} \in G \cup H$ and 0 otherwise) and variables $f_{u_i v_j}^{st}$, representing the flow value along arc (u_i, v_j) in H^{st} :

$$\text{Minimize:} \quad \sum_{u \neq v, \{u, v\} \notin E} x_{uv} \quad (5)$$

Subject to:

$$x_{uv} \in \{0, 1\} \quad (6)$$

$$0 \leq f_{u_i v_j}^{st} \leq \begin{cases} x_{uv}, & \text{if } j = i + 1 \text{ and } d_G(s, u) + d_G(u, v) + d_G(v, t) = d_G(s, t), \\ 0, & \text{otherwise.} \end{cases} \quad (7)$$

$$\sum_{u_i} f_{v_j u_i}^{st} - \sum_{u_i} f_{u_i v_j}^{st} = \begin{cases} 0, & \text{for } v_j \in V_h \setminus \{s_0, t_h\} \\ +1, & \text{for } v_j = s_0 \\ -1, & \text{for } v_j = t_h \end{cases}, \quad (8)$$

where indices s, t, u, v traverse V and indices i, j traverse $\{0, \dots, h\}$.

To obtain an LP relaxation of the above problem, we replace the integral condition $x_{uv} \in \{0, 1\}$ by the fractional one $x_{uv} \in [0, 1]$. We look at the connection between the integral and fractional forms for the special case of unique shortest path graphs.

We remark that the above formulation can be seen as a generalization of the LP and ILP statement of Angelidakis et al. [9] proposed for the special case of 2-hop labeling. In the case of 2-hop labeling, Angelidakis et al. do not rely on an explicit flow formulation but use a single constraint of the simpler form $\sum_{w \in P^{st}} \min\{x_{sw}, x_{wt}\} \geq 1$, where P^{st} represents the set of nodes on some shortest $s-t$ path in G . However, the analysis of the integrality gap does not carry over from the case of $h = 2$ to $h > 2$, i.e., as soon as there exist internal shortcuts which have neither s nor t as one of their endpoints.

4.2 Bounding Integrality Gap for Unique Shortest Path Graphs

We analyze the integrality gap of the above LP formulation for the case of *unique shortest path (USP) graphs*, i.e., graphs in which each pair of nodes $s, t \in V$ is connected by a unique shortest path P^{st} in G . We will occasionally identify P^{st} with its set of nodes, and we will introduce a linear order on its vertices, writing for $u, v \in P^{st}$ that $u <^{st} v$ if $d_G(s, u) < d_G(s, v)$; we will denote the order simply as “ $<$ ” when the path P^{st} is clear from the context. Observe that in the LP formulation, we may have $f_{u_i v_j}^{st} \neq 0$ only if $u <^{st} v$ and $j = i + 1$. Thus, fixing $s, t \in V$, the flow $f^{st} = (f_{u_i v_j}^{st} : u_i, v_j \in V_h)$ is non-zero between vertices of $\{P^{st}\} \times \{0, 1, \dots, h\}$ only, and the flow is oriented towards t on this path.

Let $(x_{uv}, f_{u_i v_j}^{st})$ be a fixed solution to the LP problem in a USP graph, with cost $COST_{LP} = \sum_{u \neq v, \{u, v\} \notin E} x_{uv}$. We will show how to use this set to construct a valid hopset H'' for G (thus, equivalently, also solving the ILP formulation). We first apply a randomized rounding procedure following the classical scheme of Raghavan and Thomson [39]. We define the family

of independent random variables $(x'_{u_i v_{i+1}} : u, v \in V, i \in \{0, \dots, h\})$, with $x'_{u_i v_{i+1}} \in \{0, 1\}$. For $u \neq v, \{u, v\} \notin E$ we put $\Pr[x'_{u_i v_{i+1}} = 1] = \min\{Cx_{uv}, 1\}$, where $C \geq 1$ is a suitably chosen probability amplification parameter (we put $C = 8h \ln n$). We will assume, without affecting the validity or cost of the solution, that $x_{uv} = x'_{u_i v_{i+1}} = 1$, when $u = v$ or $\{u, v\} \in E$.

We denote $H' = \{\{u, v\} : u, v \in V \wedge u \neq v \wedge \{u, v\} \notin E \wedge \exists_{i \in \{0, \dots, h-1\}} x'_{u_i v_{i+1}} = 1\}$. Let $\pi : V \rightarrow \{1, \dots, n\}$ be a bijection picked uniformly at random (it is a random permutation when $V = \{1, \dots, n\}$). We define the set of shortcuts $S(\{u, v\})$ associated with each pair $\{u, v\} \in H'$ as the set of all pairs of nodes on path P^{uv} , one of which is a prefix minimum on this path with respect to π , and the other of which is a suffix minimum with respect to π :

$$S(\{u, v\}) := \left\{ \{u^*, v^*\} : u^*, v^* \in P^{uv} \wedge \pi(u^*) = \min_{z \in P^{uv}, z \leq u v u^*} \pi(z) \wedge \pi(v^*) = \min_{z \in P^{uv}, z \geq u v v^*} \pi(z) \right\}.$$

The obtained solution is given as the set of all such shortcuts: $H'' := \bigcup_{\{u, v\} \in H'} S(\{u, v\})$.

► **Proposition 5.** *With probability $1 - O(1/n)$, set H'' is a hopset for G of size $O(h^2 \log^3 n \cdot \text{COST}_{\text{LP}})$.*

We remark that the above Proposition implies that the h -hopset problem can be efficiently approximated by finding an optimal fractional LP solution and constructing set H'' .

► **Theorem 6.** *There exists a randomized polynomial-time $O(\text{poly log } n)$ -approximation algorithm for the h -hopset problem in unique shortest path graphs, for any $h \leq O(\text{poly log } n)$.*

4.3 Approximating Average Query Time for 3-Hopsets

In order to design an efficient distance oracle based on 3-hopsets, we follow the framework described in the preliminaries and use an LP-rounding technique to obtain sets $H_1 \cup H_2 =: H$. The obtained claim relies on the notion of uniform-average query time introduced in the Preliminaries.

► **Theorem 7.** *For any feasible bound \mathcal{S} , let $H_{\text{OPT}, \mathcal{S}}$ be a 3-hopset for a unique shortest path graph, which satisfies the given bound on the number of edges $|H_{\text{OPT}, \mathcal{S}}| \leq \mathcal{S}$ and such that the uniform-average query time $\mathcal{T}(H_{\text{OPT}, \mathcal{S}})$ is minimized. Then, there exists a randomized polynomial-time algorithm which finds a 3-hopset H with $|H''| \leq O(\log^3 n) \mathcal{S}$ and $\mathcal{T}(H'') \leq O(\log^4 n) \mathcal{T}(H_{\text{OPT}, \mathcal{S}})$.*

We remark that the above Theorem can be directly generalized to a notion of average query time for non-uniform query densities, in which the goal is to minimize expected query time in a model in which each node $v \in V$ is assigned its relative frequency $f_v \in [0, 1]$, and a node pair uv is queried with frequency $f_u f_v$.

5 Bounded Treewidth Graphs

We now show how to obtain h -hopsets for graphs with bounded treewidth by following a classical construction for trees. We first begin with preliminaries recalling the definitions of treewidth and inverse Ackermann function.

Treewidth definition. Recall that a graph G has treewidth t if there exists a tree T whose nodes are subsets of $V(G)$ called *bags* such that: $|X| \leq t + 1$ for all $X \in V(T)$; for all edges $uv \in E(G)$, there exists a bag $X \in V(T)$ containing both u and v ($u, v \in X$); and for all nodes $u \in V(G)$, the bags containing u form a sub-tree of T . Without loss of generality, we assume that each bag contains exactly $t + 1$ nodes, and that two neighboring bags share exactly t nodes (the decomposition is standard). This implies $|V(T)| \leq n$ as each bag brings

143:12 Exact Distance Oracles Using Hopsets

one new node. Note that removing a non-leaf bag separates the graph into several connected components. We consider that all edges of T have weight 1. For convenience, we assume that T is rooted at some bag R and define for each node $u \in V(G)$ the root bag of u as the bag $R_u \in V(T)$ containing u which is closest to the root.

Inverse Ackermann notation. The k th-row inverse Ackermann function $\lambda_k(\cdot)$ can be defined by $\lambda_0(n) = \frac{n}{2}$, $\lambda_1(n) = \sqrt{n}$, $\lambda_2(n) = \log n$, $\lambda_3(n) = \log \log n$, $\lambda_4(n) = \log^* n$, and more generally for $k \geq 2$ by the recurrence $\lambda_k(n) = \lambda_{k-2}^*(n)$ where we define for any function f : $f^{(0)}(n) = n$, $f^{(i)}(n) = f(f^{(i-1)}(n))$ for $i > 0$, and $f^*(n) = \min\{j \mid f^{(j)}(n) \leq 1\}$. The inverse Ackermann function can be defined as $\alpha(n) = \min\{j \mid \lambda_{2j}(n) \leq j\}$. See [33] for a more formal definition based on Ackermann function.

We first consider the case of (weighted) trees for which the construction of h -hopsets is classical (even though the connection with hopsets was not made). It is implicit in [7, 16], explicit for unweighted trees in [13] and directed trees in [42]. We provide a short construction which fine-grains the dependence of the hopset size on h (e.g., replacing $2h$ by h with respect to the asymptotic analysis in [7]). The construction is based on the following folklore lemma for splitting a tree into smaller sub-trees (it can be seen as a generalization of the existence of a centroid).

► **Lemma 8.** *Given a rooted tree T with n nodes and a value $p > 1$, there exists a set P of at most $2p$ nodes such that each connected component of $T \setminus P$ contains less than n/p nodes and is connected to at most two nodes in P . Set P can be computed in linear time through a bottom-up traversal of the tree.*

h -hopset construction for trees. A 1-hopset in a tree T is obtained by adding all pairs as edges with appropriate weight. For $h > 1$, we recursively define a h -hopset of T as follows. Select a set P of $2p$ nodes at most with $p = \frac{n}{\lambda_{h-2}(n)}$ according to Lemma 8. When $h = 2$, we add an edge from each node u of T to each node in P . When $h > 2$, we consider the forest T' induced by nodes in P : it has node set P and edges xy such that y is the closest ancestor of x in T that belongs to P . The weight of such an edge is defined as $w'(x, y) = d_T(x, y)$. We then add a $(h - 2)$ -hopset of T' to the construction. Additionally, we add one or two edges per node not in P : for each connected component C of $T \setminus P$, add an edge ux for each node $u \in C$ and each $x \in P$ connected to C . Note that Lemma 8 ensures that there are at most two such nodes x for a given component C . In both cases ($h \geq 2$), we construct recursively a h -hopset of each sub-tree induced by a connected component C of $T \setminus P$. In the special case of $h = 3$, the $(h - 2)$ -hopsets contribute to H_2 while all edges connecting to a node in some selected set P contribute to H_1 according to the $H = H_1 \cup H_2$ convention introduced in the Preliminaries.

Following a similar approach on the tree decomposition of a graph with treewidth t , we obtain the following result (detailed construction is given in the full paper [33]).

► **Theorem 9.** *For all $h > 1$, any graph with treewidth t has a h -hopset with $O(tn\lambda_h(n))$ edges and a $2(\alpha(n) + 1)$ -hopset with $O(t^2n)$ edges.*

For the special case of $h = 3$, we have $\lambda_3(n) = \log \log n$, and the size required to represent the 3-hop data structure is $\mathcal{S} = O(tn \log \log n)$ edges. Following the convention $H = H_1 \cup H_2$, we note that we have $\deg_{H_1}(v) = O(t \log \log n)$ for any $v \in V$. The following bound on the query time follows.

► **Theorem 10.** *Any graph with treewidth t admits a 3-hopset distance oracle represented on $O(tn \log \log n)$ edges of $O(\log n)$ bits, with a query time of $O(t^2 \log^2 \log n)$.*

References

- 1 Amir Abboud, Greg Bodwin, and Seth Pettie. A Hierarchy of Lower Bounds for Sublinear Additive Spanners. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 568–576. SIAM, 2017. doi:10.1137/1.9781611974782.36.
- 2 Ittai Abraham, Daniel Delling, Amos Fiat, Andrew V. Goldberg, and Renato F. Werneck. VC-dimension and shortest path algorithms. In *ICALP*, volume 6755 of *Lecture Notes in Computer Science*, pages 690–699. Springer, 2011.
- 3 Ittai Abraham, Daniel Delling, Amos Fiat, Andrew V. Goldberg, and Renato F. Werneck. Highway Dimension and Provably Efficient Shortest Path Algorithms. *J. ACM*, 63(5):41:1–41:26, 2016. doi:10.1145/2985473.
- 4 Ittai Abraham, Daniel Delling, Amos Fiat, Andrew V. Goldberg, and Renato F. Werneck. Highway Dimension and Provably Efficient Shortest Path Algorithms. *J. ACM*, 63(5):41:1–41:26, 2016. doi:10.1145/2985473.
- 5 Ittai Abraham, Amos Fiat, Andrew V. Goldberg, and Renato F. Werneck. Highway Dimension, Shortest Paths, and Provably Efficient Algorithms. In Moses Charikar, editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 782–793. SIAM, 2010. doi:10.1137/1.9781611973075.64.
- 6 Takuya Akiba, Yoichi Iwata, and Yuichi Yoshida. Dynamic and historical shortest-path distance queries on large evolving networks by pruned landmark labeling. In Chin-Wan Chung, Andrei Z. Broder, Kyuseok Shim, and Torsten Suel, editors, *23rd International World Wide Web Conference, WWW '14, Seoul, Republic of Korea, April 7-11, 2014*, pages 237–248. ACM, 2014. doi:10.1145/2566486.2568007.
- 7 Noga Alon and Baruch Schieber. Optimal preprocessing for answering on-line product queries. Technical Report 71/87, Tel Aviv University, 1987.
- 8 Stephen Alstrup, Søren Dahlgaard, Mathias Bæk Tejs Knudsen, and Ely Porat. Sublinear Distance Labeling. In Piotr Sankowski and Christos D. Zaroliagis, editors, *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*, volume 57 of *LIPICs*, pages 5:1–5:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPICs.ESA.2016.5.
- 9 Haris Angelidakis, Yury Makarychev, and Vsevolod Oparin. Algorithmic and Hardness Results for the Hub Labeling Problem. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1442–1461. SIAM, 2017. doi:10.1137/1.9781611974782.94.
- 10 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 Josep Díaz and Maria J. Serna, editors, *Algorithms - ESA '96, Fourth Annual European Symposium, Barcelona, Spain, September 25-27, 1996, Proceedings*, volume 1136 of *Lecture Notes in Computer Science*, pages 514–528. Springer, 1996. doi:10.1007/3-540-61680-2_79.
- 11 H. Bast, Stefan Funke, and Domagoj Matijevec. Ultrafast Shortest-Path Queries via Transit Nodes. In Camil Demetrescu, Andrew V. Goldberg, and David S. Johnson, editors, *The Shortest Path Problem, Proceedings of a DIMACS Workshop, Piscataway, New Jersey, USA, November 13-14, 2006*, volume 74 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 175–192. DIMACS/AMS, 2006.
- 12 Arnab Bhattacharyya, Elena Grigorescu, Kyomin Jung, Sofya Raskhodnikova, and David P. Woodruff. Transitive-Closure Spanners. *SIAM J. Comput.*, 41(6):1380–1425, 2012. doi:10.1137/110826655.
- 13 Hans L. Bodlaender, Gerard Tel, and Nicola Santoro. Trade-Offs in Non-Reversing Diameter. *Nord. J. Comput.*, 1(1):111–134, 1994.
- 14 Sergio Cabello. Many Distances in Planar Graphs. *Algorithmica*, 62(1-2):361–381, 2012. doi:10.1007/s00453-010-9459-0.

- 15 Shiva Chaudhuri and Christos D. Zaroliagis. Shortest Paths in Digraphs of Small Treewidth. Part I: Sequential Algorithms. *Algorithmica*, 27(3):212–226, 2000. doi:10.1007/s004530010016.
- 16 Bernard Chazelle. Computing on a Free Tree via Complexity-Preserving Mappings. *Algorithmica*, 2:337–361, 1987. doi:10.1007/BF01840366.
- 17 Danny Z. Chen and Jinhui Xu. Shortest path queries in planar graphs. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, pages 469–478, 2000. doi:10.1145/335305.335359.
- 18 Edith Cohen. Using Selective Path-Doubling for Parallel Shortest-Path Computations. *J. Algorithms*, 22(1):30–56, 1997. doi:10.1006/jagm.1996.0813.
- 19 Edith Cohen. Polylog-time and near-linear work approximation scheme for undirected shortest paths. *J. ACM*, 47(1):132–166, 2000. doi:10.1145/331605.331610.
- 20 Edith Cohen, Eran Halperin, Haim Kaplan, and Uri Zwick. Reachability and Distance Queries via 2-Hop Labels. *SIAM J. Comput.*, 32(5):1338–1355, May 2003. doi:10.1137/S0097539702403098.
- 21 Vincent Cohen-Addad, Søren Dahlgaard, and Christian Wulff-Nilsen. Fast and Compact Exact Distance Oracle for Planar Graphs. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 962–973, 2017. doi:10.1109/FOCS.2017.93.
- 22 Daniel Delling, Andrew V. Goldberg, Thomas Pajor, and Renato F. Werneck. Robust Distance Queries on Massive Networks. In Andreas S. Schulz and Dorothea Wagner, editors, *Algorithms - ESA 2014 - 22th Annual European Symposium, Wroclaw, Poland, September 8-10, 2014. Proceedings*, volume 8737 of *Lecture Notes in Computer Science*, pages 321–333. Springer, 2014. doi:10.1007/978-3-662-44777-2_27.
- 23 Hristo Djidjev. On-Line Algorithms for Shortest Path Problems on Planar Digraphs. In Fabrizio d’Amore, Paolo Giulio Franciosa, and Alberto Marchetti-Spaccamela, editors, *Graph-Theoretic Concepts in Computer Science, 22nd International Workshop, WG ’96, Cadenabbia (Como), Italy, June 12-14, 1996, Proceedings*, volume 1197 of *Lecture Notes in Computer Science*, pages 151–165. Springer, 1996. doi:10.1007/3-540-62559-3_14.
- 24 Michael Elkin and Ofer Neiman. Hopsets with Constant Hopbound, and Applications to Approximate Shortest Paths. In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 128–137. IEEE Computer Society, 2016. doi:10.1109/FOCS.2016.22.
- 25 Michael Elkin and Ofer Neiman. Linear-Size Hopsets with Small Hopbound, and Distributed Routing with Low Memory. *CoRR*, abs/1704.08468, 2017. arXiv:1704.08468.
- 26 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. doi:10.1016/j.jcss.2005.05.007.
- 27 Arash Farzan and Shahin Kamali. Compact Navigation and Distance Oracles for Graphs with Small Treewidth. *Algorithmica*, 69(1):92–116, 2014. doi:10.1007/s00453-012-9712-9.
- 28 Cyril Gavoille, David Peleg, Stéphane Pérennes, and Ran Raz. Distance Labeling in Graphs. *J. Algorithms*, 53(1):85–112, October 2004. doi:10.1016/j.jalgor.2004.05.002.
- 29 Pawel Gawrychowski, Adrian Kosowski, and Przemyslaw Uznanski. Sublinear-Space Distance Labeling Using Hubs. In *Distributed Computing - 30th International Symposium, DISC 2016, Paris, France, September 27-29, 2016. Proceedings*, pages 230–242, 2016. doi:10.1007/978-3-662-53426-7_17.
- 30 Pawel Gawrychowski, Shay Mozes, Oren Weimann, and Christian Wulff-Nilsen. Better Tradeoffs for Exact Distance Oracles in Planar Graphs. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 515–529, 2018. doi:10.1137/1.9781611975031.34.
- 31 Robert Geisberger, Peter Sanders, Dominik Schultes, and Daniel Delling. Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks. In Catherine C. McGeoch, editor, *Experimental Algorithms, 7th International Workshop, WEA 2008, Provincetown, MA, USA, May 30-June 1, 2008, Proceedings*, volume 5038 of *Lecture Notes in Computer Science*, pages 319–333. Springer, 2008. doi:10.1007/978-3-540-68552-4_24.

- 32 Andrew V. Goldberg, Haim Kaplan, and Renato F. Werneck. Reach for A*: Efficient Point-to-Point Shortest Path Algorithms. In *ALLENEX*, pages 129–143. SIAM, 2006.
- 33 Siddharth Gupta, Adrian Kosowski, and Laurent Viennot. Exact Distance Oracles Using Hopsets. *CoRR*, abs/1803.06977, 2018. [arXiv:1803.06977](https://arxiv.org/abs/1803.06977).
- 34 Ronald J. Gutman. Reach-Based Routing: A New Approach to Shortest Path Algorithms Optimized for Road Networks. In *ALLENEX/ANALCO*, pages 100–111. SIAM, 2004.
- 35 Philip N. Klein and Sairam Subramanian. A Randomized Parallel Algorithm for Single-Source Shortest Paths. *J. Algorithms*, 25(2):205–220, 1997. doi:10.1006/jagm.1997.0888.
- 36 Adrian Kosowski and Laurent Viennot. Beyond Highway Dimension: Small Distance Labels Using Tree Skeletons. *CoRR*, abs/1609.00512, 2016. [arXiv:1609.00512](https://arxiv.org/abs/1609.00512).
- 37 Adrian Kosowski and Laurent Viennot. Beyond Highway Dimension: Small Distance Labels Using Tree Skeletons. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1462–1478. SIAM, 2017. doi:10.1137/1.9781611974782.95.
- 38 David Peleg and Alejandro A. Schäffer. Graph spanners. *Journal of Graph Theory*, 13(1):99–116, 1989. doi:10.1002/jgt.3190130114.
- 39 Prabhakar Raghavan and Clark D. Thompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374, 1987. doi:10.1007/BF02579324.
- 40 Hanmao Shi and Thomas H. Spencer. Time-Work Tradeoffs of the Single-Source Shortest Paths Problem. *J. Algorithms*, 30(1):19–32, 1999. doi:10.1006/jagm.1998.0968.
- 41 Mikkel Thorup. Shortcutting Planar Digraphs. *Combinatorics, Probability & Computing*, 4:287–315, 1995. doi:10.1017/S0963548300001668.
- 42 Mikkel Thorup. Parallel Shortcutting of Rooted Trees. *J. Algorithms*, 23(1):139–159, 1997. doi:10.1006/jagm.1996.0829.
- 43 Jeffrey D. Ullman and Mihalis Yannakakis. High-Probability Parallel Transitive Closure Algorithms. In *SPAA*, pages 200–209, 1990. doi:10.1145/97444.97686.