# Approximation Algorithms for Optimal Hopsets

**Michael Dinitz**
Johns Hopkins University, Baltimore, MD, USA

**Ama Koranteng**
Johns Hopkins University, Baltimore, MD, USA

**Yasamin Nazari**
Vrije Universiteit Amsterdam, The Netherlands

## Abstract

For a given graph $G$, a *hopset* $H$ with hopbound $\beta$ and stretch $\alpha$ is a set of edges such that between every pair of vertices $u$ and $v$, there is a path with at most $\beta$ hops in $G \cup H$ that approximates the distance between $u$ and $v$ up to a multiplicative stretch of $\alpha$. Hopsets have found a wide range of applications for distance-based problems in various computational models since the 90s. More recently, there has been significant interest in understanding these fundamental objects from an existential and structural perspective. But all of this work takes a worst-case (or existential) point of view: How many edges do we need to add to satisfy a given hopbound and stretch requirement for *any input graph*?

We initiate the study of the natural optimization variant of this problem: given a *specific graph instance*, what is the minimum number of edges that satisfy the hopbound and stretch requirements? We give approximation algorithms for a generalized hopset problem which, when combined with known existential bounds, lead to different approximation guarantees for various regimes depending on hopbound, stretch, and directed vs. undirected inputs. We complement our upper bounds with a lower bound that implies Label Cover hardness for directed hopsets and shortcut sets with hopbound at least 3.

**2012 ACM Subject Classification** Theory of computation → Routing and network design problems

**Keywords and phrases** Hopsets, Approximation Algorithms

**Digital Object Identifier** 10.4230/LIPIcs.ICALP.2025.69

**Category** Track A: Algorithms, Complexity and Games

**Related Version** *Full Version*: `https://arxiv.org/abs/2502.06522` [29]

**Funding** *Michael Dinitz*: Supported in part by NSF award 2228995.
*Ama Koranteng*: Supported in part by an NSF Graduate Research Fellowship.
*Yasamin Nazari*: This publication is part of the project VI.Veni.232.038 of the research programme "Dynamic graph algorithms: distances and clustering" which is financed by the Dutch Research Council (NWO).

## 1 Introduction

A hopset $H$ with hopbound $\beta$ and stretch $\alpha$ for a given (directed or undirected) graph $G$ is a set of (possibly weighted) edges such that between every pair of vertices $u$ and $v$ in $G$ there is a path with at most $\beta$ hops in $G \cup H$ that approximates the distance between $u$ and $v$ up to multiplicative stretch $\alpha$ (our results hold for a more generalized version of the problem formally defined in Definition 3). A related object is shortcut sets, which preserve *reachability* via low-hop paths, rather than distances. Hopsets were formally introduced 25 years ago by Cohen [25], and they were used to compute approximate single-source shortest paths in undirected graphs in parallel settings. More recently, they have been shown to be useful for a variety of different problems and settings, and have been studied extensively. Examples

of these applications include low parallel depth single-source shortest paths algorithms [55, 25, 65, 37], distributed shortest-paths computation [36, 35, 16], and dynamic shortest-paths [8, 48, 46, 18, 62] with implications for fast static flow-based algorithms [64, 7, 20] and dynamic clustering [27], faster construction of related objects such as distributed or massively parallel distance sketches [35, 32, 34], parallel reachability [68], and work-efficient algorithms for reachability and directed shortest paths [14, 15, 13, 41, 54].

In addition to their wide range applications, hopsets and shortcut sets are also studied as fundamental objects in their own right. There has been a surge of recent work that, rather than focusing on running time, focuses on finding the best *existential* (extremal) upper or lower bounds for hopsets and shortcut sets [57, 9, 49, 53, 58, 70]. Namely, the main goal is to find the smallest value $\gamma(n, \beta, \alpha)$ such that *every* graph $G$ on $n$ nodes admits a hopset with hopbound $\beta$ and stretch $\alpha$. Another line of work has explored the connections between hopsets and many other fundamental objects such as spanners [5, 2], emulators [38, 52], and distance preservers [56, 50, 12].

The two main existing lines of work in this direction are on finding efficient algorithms for constructing hopsets in various computational models for specific applications, or on existential bounds and structural properties. Such results are very useful, since they give us a *worst-case bound* on how many edges we must add to an arbitrary graph if we want a hopset of a certain quality and how fast can this be done for our particular application. The focus in these results has been on improving these existential bounds and developing fast algorithms for a variety of different settings and parameters. see Section 1.2 for a discussion of known results of this form.

A complementary type of problem with a very different flavor is *optimization*: *given* a graph $G$, hopbound $\beta$, and stretch bound $\alpha$, can we design an algorithm to efficiently find the smallest hopset *for* $G$ (rather than in the worst case over all graphs)? If the minimization problem is NP-hard, then can we *approximate* the smallest hopset for $G$? Note that the existential and optimization versions of these problems are complementary: good existential bounds guarantee that no matter the graph $G$, there will be a reasonably small hopset; conversely, good optimization results guarantee that we can find an approximately minimum hopset, even if the best hopset for $G$ is significantly smaller than for the worst-case graph. An existential bound might convince someone to use a hopset for some application–since they know they will never need to add too many edges–but once they commit to using a hopset for that application, they might naturally want to find the best hopset for their particular input. Additionally, in many distributed and parallel settings, adding hopset edges can be seen as a preprocessing step (that can take considerable time), after which (approximate) distance queries can be performed in fewer distributed/parallel rounds (often corresponding to the hopbound). Thus in such cases, we may be willing to spend more preprocessing time in order to add the fewest number of edges.

This natural complementarity between the existential and optimization versions has led to significant study of *both* versions for a number of related objects, most notably graph spanners (subgraphs that approximately preserve distances). For spanners, while the vast majority of work has been on the existential questions (see, e.g., [3] for the fundamental tradeoff between stretch and size), there has also been significant work on the optimization versions (see, e.g., [60, 61, 59, 39, 31, 6, 30, 33, 42, 21, 22, 45, 23, 24]). Similarly, there has also been work on optimization versions of other related objects such as reachability preservers[1], 2-hop covers [26], and diameter reduction[28].

Despite both the recent importance of hopsets and the extensive study of optimization for related objects, the optimization version of hopsets has not yet been considered. In this paper we initiate this line of research, introducing these optimization variants and proving both upper and lower bounds on their approximability.

## 1.1 Our Results and Techniques

There are many variants of this problem, divided along three main axes: whether the graph is directed or undirected, what the desired hopbound $\beta$ is, and what the required stretch is (and in particular whether it is arbitrary or whether it is in some particularly nice regime like $1+\epsilon$ for small $\epsilon$ or $2k-1$ for integer $k$). A summary of our results for these variants can be found in Table 1.

**Table 1** Comparison of hopset results, where $\epsilon' > 0$ is an arbitrarily small constant, $\epsilon \in (0,1)$, and $\eta > \beta^{1/(\ln\ln\beta - \frac{1}{2}\ln\ln\ln\beta)}$. For $\beta \geq 3$, $\eta \geq 6$. "Individual" stretch means arbitrary distance bounds, possibly different for each individual demand pair. All results are for pairwise demands, with edge lengths being nonnegative integer and upper bounded by $\texttt{poly}(n)$.

| Un/Directed | Hopbound $\beta$ | Stretch | Approximation | Theorem |
|---|---|---|---|---|
| Directed | $\widetilde{O}(n^{2/5})$ | Individual | $\widetilde{O}(\beta^{1/3} \cdot n^{2/3+\epsilon'})$ | 21 |
| Directed | $\widetilde{\Omega}(n^{2/5})$ | Individual | $\widetilde{O}(n^{1+\epsilon'}/\sqrt{\beta})$ | 22 |
| Directed | $\geq 20 \log n$ | $1+\epsilon$ | $\widetilde{O}(n^{3/4+\epsilon'} \cdot \epsilon^{-\frac{1}{4}})$ | 23 |
| Directed | 2 | Individual | $O(\ln n)$ | Full Version [29] |
| Undirected | $\widetilde{O}(n^{\frac{1}{2}-\frac{1}{2\eta}})$ | $1+\epsilon$ | $\widetilde{O}(\sqrt{\beta} \cdot n^{\frac{1}{2}+\frac{1}{2\eta}+\epsilon'})$ | 24 |
| Undirected | $\widetilde{O}(k^{-1/2} \cdot n^{\frac{1}{2}-\frac{1}{2k}})$ | $2k-1$ | $\widetilde{O}(\sqrt{k\beta} \cdot n^{\frac{1}{2}+\frac{1}{2k}+\epsilon'})$ | 25 |

### 1.1.1 Upper Bounds

While this list may appear a bit complex, it turns out that almost all of the results are generated by trading off several approximation algorithms that we design and analyze with the known existential bounds for the given regime. In particular, we consider three main approximation algorithms: one based on rounding an LP relaxation, one based on randomly sampling stars, and one based on defining and analyzing an appropriate variant of junction trees [42, 44, 22]. Instead of just analyzing the approximation ratio of each algorithm as a function of $n$ (the traditional point of view), we analyze the approximation ratios as functions of $n$, $OPT$, $\beta$ and the *local neighborhood size* [31, 6]. Importantly, none of these algorithms have performance that depends on whether the graph is undirected or directed, or what the allowed stretch is.

To get the results in Table 1, we trade these three algorithms off not only with each other based on the described parameters, but also with the known existential bounds. Unsurprisingly, there are different existential bounds known for each setting. These different existential bounds are what lead to each of the different rows of Table 1.

We also note that there is a simple reduction from hopsets to the multicriteria spanner problem, introduced by [44]. For multicriteria spanners, instead of edges being associated with just edge lengths–as with traditional spanners–edges in the multicriteria spanner problem each have a resource consumption vector, where each entry corresponds to the consumption of the respective resource on that edge. The simple reduction to multicriteria spanners implies a $\widetilde{O}(|D|^{1/2+\epsilon} \cdot \beta)$-approximation for hopsets, where $D$ is the set of vertex demand pairs given in the input.

#### 1.1.1.1 Technical Overview

Technical components of our main approximation algorithms have appeared before, in particular, in the literature on approximation algorithms for spanners. The LP rounding algorithm we use is a variant of the one introduced for spanners by [6], our star sampling is a variant of the arborescence sampling used by [31], and our junction tree algorithm is a variant of the junction trees used by [22] and [42]. But using them for hopsets involves overcoming a number of technical difficulties.

First, the performance bound on arborescence sampling used by [31, 6] fundamentally depends on the fact that for traditional graph spanners, the required connectivity implies that $\mathsf{OPT} \geq n - 1$, and hence polynomial size bounds of the form $n^\gamma$ can be interpreted as $n^{\gamma-1} \cdot \mathsf{OPT}$, i.e., as $n^{\gamma-1}$-approximations. But for hopsets, since we are adding edges rather than removing them, this is not the case: the only bound we have is the trivial bound of $\mathsf{OPT} \geq 1$ (or else we are already done). Hence bounds that are sublinear for spanners (e.g., the $O(n^{1/2})$-approximation of [6]) turn into *superlinear* approximation ratios if translated to hopsets in the obvious way. These are still nontrivial bounds, in the sense that the trivial bound for hopsets is an $O(n^2)$-approximation ($\mathsf{OPT} \geq 1$ and we can just add all pairwise edges to get a solution of size $O(n^2)$). But superlinear approximation ratios are not particularly satisfying. This is why we need different algorithms for various regimes of $\mathsf{OPT}$. By combining the star sampling and randomized LP rounding we get an approximation bound that is comparatively better for cases where $\beta$ is small and $\mathsf{OPT}$ is large. On the other hand, our junction tree algorithm performs better when $\mathsf{OPT}$ is smaller.

Second, while the LP relaxation that we use for our rounding algorithm is a natural variant of the standard spanner LP relaxation (see [31, 6, 33]), the hopset variant turns out to be significantly more difficult to solve. Most notably, solving the equivalent LP relaxations for spanners (either the flow version of [31] or the fractional cut version of [33]) requires using the ellipsoid method with a separation oracle for a problem known as Restricted Shortest Path, in which we are given two distance metrics (think of one metric as being our original edge lengths and one metric as being the fractional values given by the LP to edges) and are asked to find the shortest path in the second metric subject to a distance constraint in the first metric. Unfortunately, Restricted Shortest Path is NP-hard, but it turns out that one can use approximation algorithms for Restricted Shortest Path to solve the LP up to any desired accuracy (see [31]). But for our hopset LP, the equivalent separation oracle has *three* metrics (the original edge lengths, the number of hops, and the LP values), and we need to find the shortest path in the third metric subject to upper bounds in the first two metrics. So we design a PTAS for this more complex problem in order to solve our LP.

Third, for similar reasons our junction tree argument is more complex. Junction trees were originally introduced for network design problems where the only constraints are on connectivity, not distances; see, e.g., [19, 40]. By reducing to a "layered" graph, Chlamtáč et al. [22] showed that junction tree-based schemes are also useful in distance-constrained settings, and this layering technique was pushed significantly further by [42]. When trying to use these ideas for hopsets, though, we have the same difficulty as when solving the LP: we have essentially *two* distance constraints for each demand, corresponding to the number of hops and the allowed distance. To overcome this, we give a "two-stage" layering reduction, where we first construct a layered graph that allows us to get rid of the hop requirement without changing the distance requirements. Then we can use the further layered graph of [42] to get rid of the distance requirement, transforming it into a pure connectivity problem. In fact, we can use [42] as a black-box for this second step.

The black-box nature of our use of [42] has an interesting corollary: even without caring about junction trees, the layered graph reduction we design will immediately let us use the known results for pairwise spanners [42] (by considering them on the *weighted transitive*

*closure*) to get a non-trivial approximation guarantee for our full problem. However this reduction introduces additional factors in $\beta$, which we improve by white-boxing their approach, which will in turn give us a better trade-off in combination with the other algorithms.

### 1.1.2 Lower Bounds

To complement our upper bounds, we show that (at least for directed graphs) we cannot hope to get approximation ratios that are subpolynomial. In particular, we give an approximation-preserving reduction from the famous Label Cover problem (or more precisely, its minimization variant Min-Rep [59]) to the problem of computing the smallest hopset in a directed graph with hopbound at least 3, for any stretch value. This gives the following theorem.

▶ **Theorem 1.** *Assuming that $NP \not\subseteq DTIME(2^{polylog(n)})$, for any constant $\epsilon > 0$, and for any $\beta \geq 3$, there is no polynomial-time algorithm that can approximate directed* Generalized $\beta$-Hopset *(for any stretch value; see Definition 3) or the minimum shortcut set on directed graphs with approximation ratio better than $2^{\log^{1-\epsilon} n}$.*

Our reduction is quite similar to known hardness reductions used for spanners [59, 39, 30, 21]. The main difficulty is that these previous reductions, since they are for spanners, only have to reason about *subgraphs* of the graph created by the reduction. That is, given an instance of Min-Rep, they create some graph $G$ and argue that if the Min-REP optimum is large then any subgraph of $G$ that is a spanner must be large. But for hopsets, not only are the edges of $G$ "free," we also need to argue about hopset edges that *are not* part of $G$. This requires some changes in the reductions and analysis. The full reduction and details can be found in the full version of the paper [29].

## 1.2 Related Work

In earlier applications, sparse $(1+\epsilon)$-approximate hopsets for undirected graphs were used to obtain low parallel depth single-source shortest paths algorithms [55, 25, 65, 37]. Similarly, fast hopset constructions for undirected graphs were studied in many other settings such as distributed [36, 35, 16, 35, 32, 34] and dynamic settings [8, 48, 46, 18, 62].

Another line of work focuses on fast computation of hopsets for directed graphs and shortcut sets that preserve pairwise reachability, rather than distances, while reducing the diameter. These have gained attention particularly due to their application in parallel reachability and directed shortest path computation [68, 14, 15, 13, 41, 54]. More recently, hopsets and shortcut sets have been studied from an extremal (or existential) point of view. In particular, for $(1+\epsilon)$-hopsets, there are almost (up to $1/\epsilon$ factors) matching upper bounds [36, 52] and lower bounds [2] in *undirected graphs*. One trade-off that is widely used in $(1+\epsilon)$-approximate single-source shortest paths applications is that for any graph, there exists a $(1+\epsilon)$-hopset of size $n^{1+o(1)}$ with hopbound $n^{o(1)}$. On the other hand, [2] showed that there are instances in which this trade-off is tight. For larger stretch, better size/hopbound trade-offs are known.

In directed graphs there are polynomial gaps between existential lower bounds and upper bounds, both for approximate hopsets and shortcut sets. A widely used folklore sampling approach implies an $\widetilde{O}(n)$ size for exact hopset and shortcut sets, with hopbound $O(\sqrt{n})$ [1]. A breakthrough result of [57] went beyond this long-standing bound by constructing shortcut

[1] The upperbounds for directed hopsets and shortcut sets give a smooth trade-off between size and hopbound. For simplicity, we discuss the important linear-size regime.

sets of size $\widetilde{O}(n)$ with hopbound $\widetilde{O}(n^{1/3})$. Later, [9] showed that the same upper bound also holds for directed $(1+\epsilon)$-hopsets (up to log factors in weights and aspect ratio). More on these existential bounds can be found in Section 4.3, where we trade them off with our approximation algorithms, and in the full version [29]. Another line of work focused on obtaining subsequently better existential lower bounds for directed graphs [49, 53, 58, 70]. The current best known lower bound for a linear size shortcut set is $\widetilde{\Omega}(n^{1/4})$ [11, 70].

Another recent result [11] showed that for *exact hopsets* both in directed and undirected *weighted* graphs, the folklore sampling (see Section 4.3) is existentially optimal, implying that the separation between approximate hopsets for directed and undirected graphs, does not hold for exact hopsets.

On the approximation algorithms side, most relevant to hopsets are the weighted pairwise spanner algorithms [42], 2-hop covers [26], and multicriteria spanners [44]. In particular, weighted pairwise spanners (like hopsets and unlike $k$-spanners) do not have $n$ as a trivial lower bound on $\mathsf{OPT}$, and hence some of the difficulties encountered when designing algorithms are similar. Additionally, when we do not have distance bounds–that is, when stretch is $\infty$ for all demand pairs–weighted pairwise spanners captures the hopset problem.

Approximation algorithms for 2-hop covers [26], hub labelings [26, 4], and 2-spanners [60, 31] in particular are closely related to our hopbound 2 regime (see full version [29]). In all of these problems, the requirement of covering using 2-paths leads to algorithms that are essentially solving some variant of Set Cover. Our situation is similar, so we can use similar techniques, but our Set Cover variant is slightly different from these other problems, most notably because we do not "pay" for edges that already exist in the graph.

Multicriteria spanners were recently introduced by [44]; they also rely on a modification of the junction tree framework to give an approximation algorithm. While their junction tree framework works for our setting, we present a different framework tailored to hopsets that yields better approximations for our setting than their framework gives. Other related problems are optimizations for other variants of spanners, including traditional directed $k$-spanner (unit edge costs, all-pairs demands) [31, 6], buy-at-bulk spanners [43], and transitive closure spanners [10, 17]. In addition to transitive closure spanners, [17] shows hardness of bicriteria approximation for shortcut sets (where they allow the hopbound to be violated).

## 2 Preliminaries

Going forward, we will generally operate on what we call the *weighted transitive closure* of $G$, defined as follows. Let $d_G(s,t)$ denote the distance in $G$ from $s$ to $t$.

▶ **Definition 2** (Weighted Transitive Closure). *The weighted transitive closure of a graph $G$, denoted by $G_M = (V, E_M)$, is the weighted graph obtained by first taking the transitive closure of $G$, then assigning weight $d_G(u,v)$ to each edge $(u,v)$ in the transitive closure. We use $\widetilde{E} = E_M \setminus E$ to denote the set of edges in the weighted transitive closure that are not provided in the input $G$.*

Formally, we consider the following problem:

▶ **Definition 3** (Generalized $\beta$-Hopset). *Given a directed graph $G = (V, E)$, edge weights (or "lengths") $\ell : E \to \{1, 2, 3, ..., \mathtt{poly}(n)\}$, a set of vertex pairs $\mathcal{D} \subseteq V \times V$, and a distance bound function $Dist : \mathcal{D} \to \mathbb{N}_{\geq 0}$, find the smallest set of edges $H \subseteq \widetilde{E}$ such that for each vertex pair $(s,t) \in \mathcal{D}$, it is the case that $d_G(s,t) \leq d^{(\beta)}_{H \cup G}(s,t) \leq Dist(s,t)$. In other words, there must be an $s-t$ path $P$ in $H \cup G$ such that $|P| \leq \beta$ and $\sum_{e \in P} \ell(e) \leq Dist(s,t)$.*

If $G$ is directed, we say that the problem of interest is *directed* Generalized $\beta$-Hopset; otherwise it is *undirected* Generalized $\beta$-Hopset. Additionally, if for all $(s,t) \in \mathcal{D}$ we have that $Dist(s,t) = k \cdot d_G(s,t)$ for some $k$, then this is the *stretch-$k$* Generalized $\beta$-Hopset problem. Note that Generalized $\beta$-Hopset is a generalized version of the traditional hopset problem: all of our results hold for *any* set of vertex demand pairs, and many of our results hold for arbitrary distance bound functions.

We will use OPT to refer to the cost of the optimal solution to the Generalized $\beta$-Hopset problem instance. That is, OPT will refer to the number of edges in the optimal hopset. A path is an *$i$-hop path* if there are exactly $i$ edges on the path. We also say that a demand $(s,t) \in \mathcal{D}$ is *settled* or *satisfied* by a graph $G$ (or an edge set $E$) if there is a path from $s$ to $t$ in $G$ (in $E$) with at most $\beta$ hops and distance at most $Dist(s,t)$. Otherwise, demand $(s,t)$ is considered *unsettled* or *unsatisfied*.

Note that the weighted transitive closure of a graph can be found in polynomial-time. When working in the weighted transitive closure, we will generally assign costs to the edges in $G_M$: edges in $E$ will have cost 0, while edges in $\widetilde{E}$ will have cost 1. It is easy to see that Generalized $\beta$-Hopset on $G$ is equivalent to finding a min-cost subgraph of $G_M$ that settles all demand pairs. We will use $c(F)$ to denote the *cost* of an edge set $F$; namely, $c(F) = |F \cap \widetilde{E}|$ is the number of edges in $F$ not provided in the input. Finally, we will assume each edge $(u,v) \in E$ in the input is a shortest path from $u$ to $v$ in $G$.

## 3 LP Relaxation

In this section, we state and solve a cut-covering linear program (LP) for Generalized $\beta$-Hopset. A few of our approximation algorithms will randomly round the solution to this LP as a subroutine.

Let $\mathcal{P}_{s,t}$ be the set of paths from $s$ to $t$ that have at most $\beta$ hops and path length at most $Dist(s,t)$. We will refer to these paths as "allowed" or "valid" paths. The natural flow LP for our problem requires building enough capacity to send one unit of (non-interfering) flow along valid paths for each demand; this is the basic LP used in essentially all network design problems, and was introduced for spanners by [31]. An equivalent LP, which is what we will use for Generalized $\beta$-Hopset, is obtained through the duality between flows and fractional cuts (of valid paths). This version of the LP for spanners was studied by [33], and strengthens the anti-spanner LP of [6].

In more detail, for a graph $G = (V,E)$, we say that an "$s-t$ cut against valid paths" is a set of edges $F$ such that in the graph $G \setminus F$, there are no valid paths from $s$ to $t$. In the cut covering LP we will use, the constraints will ensure that any feasible solution must "cover" every cut against valid paths; that is, any feasible solution must buy an edge in each of these cuts. This leads to our LP relaxation, which we call LP(Hopset). The LP has a variable $x_e$ for every edge $e \in E_M$. Note that because edges in $E$–edges from the input graph–do not contribute to the cost of the solution, we can assume without loss of generality that $x_e = 1$ for all $e \in E$. Let $\mathcal{Z}_{s,t} = \{\mathbf{z} \in [0,1]^{|E|} : \forall P \in \mathcal{P}_{s,t}\ ,\ \sum_{e \in P} z_e \geq 1\}$ be the set of vectors representing all fractional cuts against valid $s-t$ paths. For each demand, our LP requires any feasible integral solution to buy at least one edge from every cut against valid paths.

$$\begin{aligned} \min \quad & \sum_{e \in \widetilde{E}} x_e & & \\ \text{s.t.} \quad & \sum_{e \in E_M} z_e x_e \geq 1 & & \forall (s,t) \in \mathcal{D},\, \forall \mathbf{z} \in \mathcal{Z}_{s,t} \\ & x_e \geq 0 & & \forall e \in E_M \end{aligned} \tag{LP(Hopset)}$$

As written, LP(Hopset) has an infinite number of constraints. Consider the polytope $\mathcal{Z}_{s,t}$ for some demand $(s,t)$. Due to convexity, we only need to keep the constraints that correspond to the vectors $\mathbf{z}$ that form the vertices of of $\mathcal{Z}_{s,t}$. Since there are only an exponential number of these constraints, we can assume the LP has exponential size. It is easy to see that LP(Hopset) is a valid LP relaxation of Generalized $\beta$-Hopset; see the full version for a proof [29].

## 3.1 Solving the LP

Some of our algorithms for Generalized $\beta$-Hopset involve some flavor of random LP rounding, so in this section we will (approximately) solve LP(Hopset), our LP of interest. The LP has an exponential number of constraints, so we must solve it using the ellipsoid algorithm with a separation oracle. To do so, we start by defining another LP, LP(Oracle 1), that captures the problem of finding a violated constraint of LP(Hopset). To solve LP(Oracle 1), we find yet another (approximate) separation oracle. This second oracle boils down to solving a hopbounded version of the Restricted Shortest Path problem. We show that this problem admits an FPTAS, and that this ultimately translates to a $(1+\epsilon)$-approximation for LP(Hopset). More formally, we will prove the following:

▶ **Theorem 4.** *There is a $(1+\epsilon)$-approximation to the optimal solution of LP(Hopset) for any arbitrarily small constant $\epsilon > 0$.*

### 3.1.1 Oracle 1

As noted, LP(Hopset) has exponential size, so we find a separation oracle in order to run ellipsoid. To do so, we must determine if there is a fractional cut $\mathbf{z}$ that is not covered by the LP solution $\mathbf{x}$–that is, given $\mathbf{x}$ (satisfying the non-negativity constraints), we want to find $\mathbf{z} \in \mathcal{Z}_{s,t}$, for some demand $(s,t) \in \mathcal{D}$, such that $\sum_{e \in E_M} z_e x_e < 1$. We can find such a violated cut-covering constraint by solving the following LP for each demand $(s,t) \in \mathcal{D}$.

$$\begin{aligned} \min \quad & \sum_{e \in E_M} z_e x_e & & \\ \text{s.t.} \quad & \sum_{e \in P} z_e \geq 1 & & \forall P \in \mathcal{P}_{s,t} \\ & z_e \geq 0 & & \forall e \in E_M \end{aligned} \tag{LP(Oracle 1)}$$

### 3.1.2 Oracle 2: Hopbounded Restricted Shortest Path Problem

LP(Oracle 1) is also exponential in the number of constraints, so we use the ellipsoid algorithm again, with yet another separation oracle. Given an LP solution $\mathbf{z}$, we must determine whether there is some valid $s-t$ path that is not fractionally cut (or, covered) by $\mathbf{z}$. Specifically, we

need to find a violated constraint of the form $\sum_{e \in P} z_e < 1$ for some path $P \in \mathcal{P}_{s,t}$. Observe that we now have three separate metrics: the $\mathbf{z}$ metric (given by a solution to LP(Oracle 1)), our original distance metric from the input, and a hop metric (where each edge has hop "length" 1). We only care about valid $s-t$ paths; namely, paths with length at most $D(s,t)$ in our original distance metric and with length at most $\beta$ in our hop metric. Thus, our goal is to find the shortest path in the $\mathbf{z}$ metric that respects these upper bounds in the distance and hop metrics.

When there are only two metrics–that is, when the goal is to find the shortest path in one metric subject to an upper bound in the other–this is the Restricted Shortest Path problem, defined as follows.

▶ **Definition 5** (Restricted Shortest Path Problem). *Let $G = (V, E)$ be a graph such that each edge $e \in E$ is associated with a cost $z_e$ and a delay $\ell_e$. Let $T$ be a positive integer, and $s, t \in V$ be the source and target nodes, respectively. The Restricted Shortest Path problem is to find a path $P$ from $s$ to $t$ such that the delay along this path is at most $T$, and the cost of $P$ is minimal.*

The Restricted Shortest Path problem is well studied [47, 69, 66, 71, 51]. The problem admits an FPTAS, meaning there exists a polynomial-time $(1+\epsilon)$-approximation for the problem [63]. Since our problem of interest has a third metric (the hop metric), we refer to it as the *Hopbounded* Restricted Shortest Path problem. We modify the Restricted Shortest Path FPTAS algorithm of [63] to give an FPTAS for the Hopbounded Restricted Shortest Path problem, thus giving a $(1+\epsilon)$-approximate separation oracle for LP(Oracle 1). We formally define the Hopbounded Restricted Shortest Path problem with respect to LP(Oracle 1):

▶ **Definition 6** (Hopbounded Restricted Shortest Path problem). *Let $G_M = (V, E_M)$ be a graph such that each edge $e \in E$ is associated with a cost $z_e$ and a delay, or length, $\ell_e$. Let $T = D(s,t)$, and $s, t \in V$ be the source and target nodes, respectively. The Hopbounded Restricted Shortest Path problem is to find a path $P$ from $s$ to $t$ with at most $\beta$ hops, such that the length along this path is at most $T$, and the cost of $P$ is minimal.*

### 3.1.3 Hopbounded Restricted Shortest Paths Algorithm

At a high level, the Restricted Shortest Path algorithm works as follows. The algorithm runs multiple binary searches to find good upper and lower bounds on the cost of the optimal solution, then uses these bounds to scale and discretize (or "bucket") the costs of the edges. They then give a pseudo-polynomial time dynamic programming algorithm on the problem with bucketed edge costs, which they show is a $(1+\epsilon)$-approximation for the original problem.

To get this algorithm to work for the Hopbounded Restricted Shortest Path problem, we simply modify the DP algorithm to take our hop metric into account, adding a factor $\beta$ to the runtime (see the full version for the algorithm and analysis [29]). The arguments of [63] generally also hold for our algorithm, so we have that our algorithm is a $(1+\epsilon)$-approximation of the Hopbounded Restricted Shortest Path problem.

▶ **Lemma 7.** *The Hopbounded Restricted Shortest Path problem admits an FPTAS.*

### 3.1.4 Proof of Theorem 4

With an FPTAS for the Hopbounded Restricted Shortest Path problem (by Lemma 7), we have an approximate separation oracle for LP(Oracle 1). Using the ellipsoid algorithm with this oracle, we find a solution $\mathbf{z}$ for LP(Oracle 1). While $\mathbf{z}$ may not be feasible, it only violates

each constraint by a factor of at most $(1+\epsilon)$. That is, $\mathbf{z}$ satisfies $(1+\epsilon)\sum_{e\in P} z_e \geq 1$ for all $P \in \mathcal{P}_{s,t}$. Thus if we scale $\mathbf{z}$ by $(1+\epsilon)$, we get a feasible solution. Let $\mathbf{z'}$ be this scaled solution, where $z'_e = (1+\epsilon)z_e$ for all $e \in E_M$. We then also have that $\mathbf{z'}$ is a $(1+\epsilon)$-approximation for LP(Oracle 1). This is implied by the fact that for any feasible solution $\mathbf{z''}$ of LP(Oracle 1), the value of the objective on $\mathbf{z}$ is at most the value of the objective on $\mathbf{z''}$ (the entire feasible region satisfies the $(1+\epsilon)$ "approximate" constraints, and therefore the feasible region is in the search space of the ellipsoid algorithm). That is, $\sum_{e\in E_M} z_e x_e \leq \sum_{e\in E_M} z''_e x_e$ for all feasible solutions $\mathbf{z''}$. Thus we have a $(1+\epsilon)$-approximation for LP(Oracle 1). By similar arguments, which we describe in the full version [29], this also implies a $(1+\epsilon)$-approximation for the hopset LP.

## 4 Approximation Algorithms for General Hopbounds

Continuing the connection to spanners, there is a reduction from Generalized $\beta$-Hopset to the directed pairwise weighted spanner problem (where "weighted" refers to edge costs). The most general version of this problem, studied by [42], allows for any demand set, any positive rational edge costs, integer edge weights in $\texttt{poly}(n)$, and arbitrary distance bound functions. The reduction starts with the transitive closure $G_M$, and builds a layered graph with $\beta+1$ copies of each node in $G_M$, and $\beta$ copies of each edge (see Section 4.1.2 for a more detailed description of the reduction). Since [42] achieved an $\widetilde{O}(n^{4/5+\epsilon})$-approximation for directed pairwise weighted spanners, this reduction immediately gives an $\widetilde{O}((\beta n)^{4/5+\epsilon})$-approximation for Generalized $\beta$-Hopset.

In this section, we improve upon this result and get an $\widetilde{O}(n^{4/5+\epsilon})$-approximation for Generalized $\beta$-Hopset, removing the dependence on $\beta$. We will give approximation algorithms in terms of $n, \beta, \mathsf{OPT}$, and "local neighborhood size." All of our algorithms are based on spanner algorithms, and we must modify them (and provide different analyses) to accommodate the hop constraint. We will then trade these algorithms off with known existential hopset results to get approximations (in terms of $n$ and $\beta$) in many regimes.

Our first algorithm, the junction tree algorithm of Section 4.1, will perform best when $\beta$ and $\mathsf{OPT}$ are relatively small. Our second and third algorithms, the star sampling and randomized LP rounding algorithms of Section 4.2, will together give better approximations as $\mathsf{OPT}$ gets larger. We also give an $O(\log n)$-approximation for hopbound 2, which we defer to the full version [29].

### 4.1 Junction Tree Algorithm

In this section, we prove the following theorem for directed Generalized $\beta$-Hopset.

▶ **Theorem 8.** *There is a polynomial-time $\widetilde{O}(\beta n^{\epsilon} \cdot \mathsf{OPT})$-approximation for directed* Generalized $\beta$-Hopset.

To prove the theorem, we give an algorithm similar to a subroutine of the directed pairwise weighted spanner algorithm of [42], where "weighted" refers to edge costs. In the pairwise weighted spanner subroutine of [42], they define a variant of the junction tree (the "distance-preserving junction tree"). Junction trees are rooted trees that satisfy demands, and *good* junction trees satisfy many demands at low cost; that is, they have low "density." Junction trees have been used in several spanner approximation algorithms (e.g. [42, 22, 44]). In [42], they give an algorithm that iteratively buys their version of low density junction trees until all demands are satisfied. Our algorithm will follow the same structure. The main technical work in this section is in showing that low-density *hopbounded* junction trees exist in our setting, and that we can use the subroutine of [42] to find these hopbounded junction trees, even though their subroutine does not have any hop guarantees.

We note that the junction tree framework developed by [44] for multicriteria spanners also works for our setting. Their framework implies an $\widetilde{O}(|\mathcal{D}|^{\epsilon} \cdot \beta)$-approximation for finding the hopbounded minimum-density junction tree, and thus a $\widetilde{O}(\beta^2 \cdot |\mathcal{D}|^{\epsilon} \cdot \mathsf{OPT})$-approximation for directed Generalized $\beta$-Hopset. By tailoring our framework to hopsets, we achieve an $O(n^{\epsilon})$-approximation for finding the hopbounded min-density junction tree (note the lack of dependence on $\beta$), implying our main result of this section (Theorem 8), a $\widetilde{O}(\beta \cdot n^{\epsilon} \cdot \mathsf{OPT})$-approximation overall. By using our hopbounded junction tree framework, we lose a factor $\beta$ in our overall approximation compared to what is implied by [44].

We first define a hopbounded variant of the junction tree, which we call an $(i, j)$-distance-preserving hopbounded junction tree. We parameterize by $i, j$, where $i + j \leq \beta$, to ensure that both "sides" of the rooted tree–the in-arborescence and the out-arborescence that make up the tree–are hopbounded by $i$ and $j$, respectively, so that all paths in the tree have at most $\beta$ hops.

▶ **Definition 9** ($(i, j)$-Distance-Preserving Hopbounded Junction Tree)**.** *An* $(i, j)$-distance-preserving hopbounded junction tree*, where* $i + j \leq \beta$*, is a subgraph of* $G_M$ *that is a union of an in-arborescence and an out-arborescence, both rooted at some vertex* $r \in V$*, with the following properties: 1) every leaf of the in-arborescence has a path of at most* $i$ *hops to* $r$*, 2) for every leaf* $w$ *in the out-arborescence, there is a path of at most* $j$ *hops from* $r$ *to* $w$*, and 3) for some node* $s$ *in the in-arborescence and some node* $t$ *in the out-arborescence, there is an* $s - t$ *path through* $r$ *with distance at most* $Dist(s, t)$*. The* density *of an* $(i, j)$*-distance-preserving hopbounded junction tree* $T$ *is the ratio of the cost of* $T$ *to the number demands settled by* $T$*.*

Going forward, we will refer to the $(i, j)$-distance-preserving hopbounded junction tree as simply an "$(i, j)$-junction tree." Our algorithm will find and remove a low-density hopbounded junction tree from $G_M$, add its edges to the current solution, and repeat, until all demand pairs are settled. We will give an $O(n^{\epsilon})$-approximation for finding these low-density junction trees. The algorithm will return a subgraph with total cost of $\widetilde{O}(\beta^2 n^{\epsilon} \cdot \mathsf{OPT}^2)$.

### 4.1.1 Existence of Low-Density Junction Trees

Let $D$ be the set of *unsettled* vertex pairs at some iteration of the algorithm. We first use an averaging argument to show that a hopbounded junction tree with density $O(\beta \cdot \mathsf{OPT}^2 / |D|)$ always exists (at any iteration), where $\mathsf{OPT}$ is the cost of the optimal solution to the problem instance. We defer the proof to the full version [29].

▶ **Lemma 10.** *For any set of unsettled demands* $D$*, there exists an* $(i, j)$*-junction tree with density* $O(\beta \cdot \mathsf{OPT}^2 / |D|)$*.*

### 4.1.2 Layered Graph Reduction

We want to show that we can *find* a junction tree with low enough density at each iteration of the algorithm. To do so, we will use the junction-tree finding subroutine provided in [42]. Their subroutine, however, finds non-hop-constrained junction trees. We therefore transform our input graph in order to use their subroutine to find $(i, j)$-junction trees. We build the following $\beta$-layered graph out of $G_M$.

#### 4.1.2.1 Layered Graph Construction

To construct the layered graph $G_L = (V_L, E_L)$ with costs $c_L : E_L \to \{0,1\}$, weights $\ell_L : E_L \to \{1, 2, \ldots, \texttt{poly}(n)\}$, demand set $\mathcal{D}_L$, and distance bounds $Dist_L : \mathcal{D}_L \to \mathbb{N}_{\geq 0}$, we first create $\beta + 1$ copies of each vertex $u \in V$ so that $u$ corresponds to vertices $u_0, u_1, \ldots, u_\beta$ in $V_L$. For each edge $(u, v) \in E_M$ we add edges $\{(u_i, v_{i+1}) : i \in [0, \beta - 1]\}$ to $E_L$. For each edge $e = (u_i, v_{i+1})$ of this type, we set $\ell_L(e) = \ell(u, v)$. We also set $c_L(e) = 1$ if $(u, v) \in \widetilde{E}$; otherwise we set $c_L(e) = 0$. For each vertex in $V_L$, we also add edges $\{(u_i, u_{i+1}) : i \in [0, \beta - 1]\}$ to $E_L$, and set their costs and weights to 0. Finally, we add a demand $(s_0, t_\beta)$ to $\mathcal{D}_L$ for demand each $(s, t) \in \mathcal{D}$.

By design, $(i, j)$-junction trees in $G_M$ correspond to $(i, j)$-junction trees in $G_L$ (and vice versa) with the same density. The proof of this is straightforward, and is deferred to the full version [29]. We say that Min Density $(i, j)$-Junction Tree is the optimization problem of finding the minimum density $(i, j)$-junction tree in an input graph, over all possible values of $i, j$.

▶ **Lemma 11.** *If there is an $\alpha$-approximation algorithm for* Min Density $(i, j)$-Junction Tree *on $G_L$, then there is also an $\alpha$-approximation algorithm for* Min Density $(i, j)$-Junction Tree *on $G_M$.*

### 4.1.3 Junction Tree-Finding Subroutine

We now show that we can find low-density junction trees at each iteration of the algorithm. Although $(i, j)$-junction trees are hopbounded by definition, we can now use the following length-bounded junction tree-finding subroutine of [42] to find hopbounded junction trees, thanks to the reduction to the $\beta$-layered graph $G_L$.

▶ **Lemma 12** (Lemma 16 of [42])**.** *For any constant $\epsilon > 0$, there is a polynomial-time approximation algorithm for finding the minimum density distance-preserving junction tree. That is, there is a polynomial time algorithm which, given a weighted directed $n$-vertex graph $G = (V, E)$ where each edge $e \in E$ has cost $c(e) \in \mathbb{R}_{\geq 0}$ and integral length $\ell(e) \in \{0, 1, \ldots, \texttt{poly}(n)\}$, terminal pairs $\mathcal{D} \subseteq V \times V$, and distance bounds $Dist : \mathcal{D} \to \mathbb{N}$ for each terminal pair $(s, t) \in \mathcal{D}$, approximates the following problem to within an $O(n^\epsilon)$ factor:*

- *Find a non-empty set of edges $F \subseteq E$ minimizing the ratio:*

$$\min_{r \in V} \frac{\sum_{e \in F} c(e)}{|\{(s,t) \in \mathcal{D} : d_{F,r}(s,t) \leq Dist(s,t)\}|}$$

*where $d_{F,r}(s, t)$ is the length of the shortest path using edges in $F$ which connects $s$ to $t$ while going through $r$ (if such a path exists). We call this problem* Min Density Length-Bounded Junction Tree.

This gives an $O(n^\epsilon)$-approximation algorithm for finding the min-density $(i, j)$-junction tree on $G_M$. The proof of the following lemma can be found in the full version [29].

▶ **Lemma 13.** *There is an $O(n^\epsilon)$-approximation for* Min Density $(i, j)$-Junction Tree *on $G_M$.*

▶ **Lemma 14.** *There is a polynomial-time algorithm that finds an $(i, j)$-junction tree with density $O(\beta n^\epsilon \cdot \mathsf{OPT}^2/|D|)$, where $D$ is the set of unsettled demands in $G$.*

**Proof.** By Lemma 10, there exists an $(i, j)$-junction tree with density $O(\beta \cdot \mathsf{OPT}^2/|D|)$. We can run the $O(n^\epsilon)$-approximation algorithm (Lemma 13) on $G_M$, which outputs an $(i, j)$-junction tree with density $O(\beta n^\epsilon \cdot \mathsf{OPT}^2/|D|)$. ◀

### 4.1.4 Proof of Theorem 8

By iteratively buying these low-density $(i,j)$-junction trees, we get an $O(\beta n^{\epsilon} \cdot \mathsf{OPT})$-approximation for Generalized $\beta$-Hopset. We defer the proof to the full version [29].

## 4.2 Star Sampling with Randomized LP Rounding Algorithm

In this section we prove the following theorem.

▶ **Theorem 15.** *There is a randomized algorithm for directed* Generalized $\beta$-Hopset *with expected approximation ratio* $O(n \ln n / \sqrt{\mathsf{OPT}})$.

The pair of algorithms we give closely follow the $\widetilde{O}(n^{2/3})$- and $\widetilde{O}(\sqrt{n})$-approximations for the unweighted $k$-spanner problem, given by [31] and [6], respectively. The $k$-spanner algorithm is a trade-off between two algorithms: an arborescence sampling algorithm for settling a class of edges (or demands) that they call "thick," and a randomized LP rounding algorithm for settling "thin" edges. For hopsets we will settle these thick demands by sampling directed stars instead of arborescences, and for thin demands we will use a similar LP rounding approach. Although our hopset algorithms are similar to the $k$-spanner algorithms, we get a different approximation ($\widetilde{O}(n/\sqrt{\mathsf{OPT}})$ for hopsets versus $\widetilde{O}(\sqrt{n})$ for spanners). This is because [31, 6] take advantage of the fact that for spanners, $\mathsf{OPT} \geq n-1$. This is not the case for hopsets so we get a different approximation out of the algorithms, in terms of $\mathsf{OPT}$. This approach is also similar to that of [22] for pairwise distance preservers, where again, $\Omega(n)$ is not a lower bound for $\mathsf{OPT}$.

We note that our $O(n \ln n / \sqrt{\mathsf{OPT}})$-approximation is achieved by trading off the two aforementioned algorithms (star-sampling and randomized-rounding). To later achieve the optimal trade-off with other algorithms, one should a priori treat each of these two algorithms as separate, with their own individual approximation ratios. It is however equivalent to trade these two algorithms off first and treat them as one combined algorithm, which we do going forward. This is because these are our only algorithms that will depend on the "local neighborhood size" parameter.

To define thick and thin demands, we must first define subgraphs $G^{s,t}$ for all demands $(s,t)$, as in [31, 6]:

▶ **Definition 16.** *For a demand* $(s,t) \in \mathcal{D}$, *let* $G^{s,t} = (V^{s,t}, E^{s,t})$ *be the subgraph of* $G_M$ *induced by the vertices on paths in* $\mathcal{P}_{s,t}$. *We call* $|V^{s,t}|$ *the* local neighborhood size.

▶ **Definition 17** (Thick and Thin Demands). *Let* $b$ *be a parameter in* $[1,n]$. *If* $|V^{s,t}| \geq n/b$ *then the corresponding demand* $(s,t)$ *is thick, otherwise it is thin. We shall always assume that* $b = \sqrt{\mathsf{OPT}}$.

Let $\mathcal{D}_{thick}$ and $\mathcal{D}_{thin}$ be the set of all thick and thin demands, respectively. We will run two algorithms to build two edge sets, $E'$ and $E''$, such that all thick demands are settled by $E'$ and all thin demands are settled by $E''$. The set $E'$ will have cost $O(bn \ln n)$ in expectation, while $E''$ will have cost $O((n/b) \ln n \cdot \mathsf{OPT})$ in expectation. The optimal tradeoff of these algorithms has $b = \sqrt{\mathsf{OPT}}$, so each edge set will have cost $O(n \ln n \cdot \sqrt{\mathsf{OPT}})$ in expectation.

### 4.2.1 Star-Sampling Algorithm for Thick Demands

We describe the random sampling subroutine for constructing the edge set $E'$, which will settle all thick demands (Algorithm 1).

**Algorithm 1** Star-Sampling Algorithm.

**Input:** Graph $G_M = (V, E_M)$
Let $E' \leftarrow \emptyset$, $S \leftarrow \emptyset$ `// Set S is only used for the analysis`
**foreach** *index* $i = 1, 2, \ldots, b \ln n$ **do**
  $v \leftarrow$ a uniformly random element from $V$
  $T_v^{in} \leftarrow$ inward star of $G_M$ rooted at $v$
  $T_v^{out} \leftarrow$ outward star of $G_M$ rooted at $v$
  $E' \leftarrow E' \cup T_v^{in} \cup T_v^{out}$, $S \leftarrow S \cup \{v\}$
**foreach** *unsettled demand* $(s,t) \in \mathcal{D}_{thick}$ **do**
  $E' \leftarrow E' \cup (s,t)$
**Return** $E'$

This algorithm is nearly identical to that of [31]. The only difference is that, since we operate on the weighted transitive closure of $G$, we build directed in- and out-stars as opposed to the shortest path in- and out-arborescences used for the spanner setting.

We now show that $E'$ has the desired cost in expectation. While [31] proves this for spanners, it is easy to see that a near identical argument also holds for hopsets in $G_M$. We restate the proof in the full version for completeness [29].

▶ **Lemma 18** ([31]). *Algorithm 1, in polynomial time, computes an edge set $E'$ that settles all thick demands and has expected cost $O(bn \ln n)$. If $b = \sqrt{\mathsf{OPT}}$, then the expected cost is $O(n \ln n \cdot \sqrt{\mathsf{OPT}})$.*

### 4.2.2 Randomized LP Rounding Algorithm for Thin Demands

We now give the algorithm for finding a set $E''$ to settle thin demands. [6] introduces the notion "anti-spanners," which is crucial for the algorithm and analysis for settling thin demands. In particular, they formulate an anti-spanner covering LP that captures the problem of settling all thin demands. They then solve the LP (with high probability) by constructing a separation oracle that utilizes randomized rounding. We also use randomized LP rounding, though instead of rounding the solution to an "anti-hopset" covering LP, we will round based on LP(Hopset). Our LP is stronger than the "anti-hopset" covering LP, since our LP is for *fractional* cuts against valid paths, while the anti-hopset covering LP is only for integer cuts.

Going forward, we assume without loss of generality that we know the value of the optimal solution–$\mathsf{OPT}$ is in $\{0, 1, \ldots, n^2\}$, so we can just try each of these values for $\mathsf{OPT}$ and return the smallest hopset found over all tries. We can therefore replace the objective function of LP(Hopset) with the following:

$$\sum_{e \in \widetilde{E}} x_e \leq \mathsf{OPT} \tag{4}$$

We use this modified version of LP(Hopset) for the randomized rounding algorithm. Given a fractional solution $\mathbf{x}^*$ to LP(Hopset), our algorithm will return an edge set $E''$ that, with high probability, will cost at most $2\mathsf{OPT} \cdot 2(n/b) \ln n$ and satisfy all thin demands (see Algorithm 2). We say that the algorithm *fails* if $c(E'') > 2\mathsf{OPT} \cdot 2(n/b) \ln n$ or if $E''$ doesn't satisfy all thin demands.

We now show that the probability is exponentially small that the algorithm fails. The argument is very similar to that given by [6] for spanners; we state it in the full version for completeness [29].

**Algorithm 2** Randomized LP Rounding Algorithm.

**Input:** Graph $G_M = (V, E_M)$, LP(Hopset) fractional solution $\mathbf{x}^*$
Let $E'' \leftarrow \emptyset$

`// sample edges into` $E''$
**foreach** *edge* $e \in E_M$ **do**
  Let $p_e \leftarrow \min(1, 2(n/b) \ln n \cdot x^*_e)$
  Add $e$ to $E''$ with probability $p_e$

**if** $E''$ *settles all thin demands* **then**
  **Return** $E''$
**else**
  **Return** $E_M \setminus E$

▶ **Lemma 19** (Theorem 2.2 of [6]). *The probability that Algorithm 2 fails is exponentially small in $n$.*

### 4.2.3 Proof of Theorem 15

**Proof.** All thick demands can be satisfied by running Algorithm 1 to build $E'$, which has expected cost $O(n \ln n \cdot \sqrt{\mathsf{OPT}})$ (by Lemma 18) and runs in polynomial time. The thin demands can be satisfied by running Algorithm 2, which runs in polynomial time. Algorithm 2 fails with exponentially small probability (in which case we return all possible hopset edges, $\widetilde{E}$), and thus the expected cost of $E''$ is at most $\mathsf{OPT} \cdot 2(n/b) \ln n + o(1) = O(n \ln n \cdot \sqrt{\mathsf{OPT}})$ (Lemma 19). Thus the overall approximation ratio is $O(n \ln n / \sqrt{\mathsf{OPT}})$. ◀

## 4.3 Trade-Offs with Existential Bounds

There are a number of constructive existential results for hopsets that we trade off with our junction tree-based algorithm from Section 4.1 (an $\widetilde{O}(\beta n^{\epsilon} \cdot \mathsf{OPT})$-approximation) and our star-sampling/randomized-rounding algorithm from Section 4.2 (an $\widetilde{O}(n/\sqrt{\mathsf{OPT}})$-approximation) to give approximations in several regimes. Our junction tree algorithm gives much better approximations than all other existential results when $\mathsf{OPT}$ and $\beta$ are relatively small, so it will be used in the trade off for all regimes. The star-sampling/randomized-rounding algorithm gives improved approximations over the junction tree algorithm as $\mathsf{OPT}$ gets larger.

We trade off a folklore existential result for directed, exact hopsets, along with the existential results of [9] (directed, $(1+\epsilon)$-stretch), [37] (undirected, $(1+\epsilon)$-stretch), and [67] (undirected, odd stretch) to get improved approximations over the general problem in these regimes. We also note that a trade off can be done with the existential results of [5] for improved approximations in the constant stretch, $\Omega(\log n)$-hopbound regime for undirected graphs. Further discussions of each trade-off can be found in the full version [29].

### 4.3.1 Directed Graphs with Arbitrary Distance Bounds

We trade off our junction tree and star-sampling/randomized-rounding algorithms with the folklore construction to achieve the following for directed Generalized $\beta$-Hopset:

▶ **Theorem 20.** *There is a randomized polynomial-time $\widetilde{O}(n^{4/5+\epsilon})$-approximation for directed* Generalized $\beta$-Hopset.

Where $\epsilon > 0$ is an arbitrarily small constant. When $\beta$ is smaller, the folklore construction gives worse approximations than the trade-off between our other algorithms. In this regime, we achieve a better approximation:

▶ **Theorem 21.** *When $\beta = \widetilde{O}(n^{2/5})$, there is a randomized polynomial-time $\widetilde{O}(\beta^{1/3} \cdot n^{2/3+\epsilon})$-approximation for directed* Generalized $\beta$-Hopset*.*

We also get improved approximations in the large $\beta$ setting by trading off just the junction tree algorithm with the folklore construction:

▶ **Theorem 22.** *When $\beta = \widetilde{\Omega}(n^{2/5})$, there is a randomized polynomial-time $\widetilde{O}(n^{1+\epsilon}/\sqrt{\beta})$-approximation for directed* Generalized $\beta$-Hopset*.*

### 4.3.2 Directed Hopsets with Small Stretch

Using the existential bound of [9], we get an improved approximation for directed Generalized $\beta$-Hopset when we restrict to $(1+\epsilon)$ stretch.

▶ **Theorem 23.** *When $\beta \geq 20 \log n$ and $\epsilon \in (0,1)$, there is a randomized polynomial-time $\widetilde{O}(n^{3/4+\epsilon'} \cdot \epsilon^{-\frac{1}{4}})$-approximation for directed stretch-$(1+\epsilon)$* Generalized $\beta$-Hopset*, where $\epsilon' > 0$ is an arbitrarily small constant.*

### 4.3.3 Undirected Hopsets with Small Stretch

Let $W_0(x)$ be the principle branch of the Lambert $W$ function, which is upper bounded by $\ln x - (1/2) \ln \ln x$ when $x \geq 3$. With the existential bound of [37], we get the following:

▶ **Theorem 24.** *Let $\eta = \lfloor \beta^{1/W_0(\ln \beta)} \rfloor > \beta^{1/(\ln \ln \beta - \frac{1}{2} \ln \ln \ln \beta)}$ (inequality holds for $\beta \geq 3$). When $\beta = \widetilde{O}(n^{\frac{1}{2} - \frac{1}{2\eta}})$, there is a randomized polynomial-time $\widetilde{O}(\sqrt{\beta} \cdot n^{\frac{1}{2} + \frac{1}{2\eta} + \epsilon'})$-approximation for undirected $(1+\epsilon)$-stretch* Generalized $\beta$-Hopset*, where $\epsilon \in (0,1)$, and $\epsilon' > 0$ is an arbitrarily small constant.*

For some insight into the behavior of $\eta$, note first that for all $\beta \geq 3$, $\eta \geq 6$. Additionally, the $\eta$ function grows faster than $\ln \beta$, but much slower than $\beta$.

### 4.3.4 Undirected Hopsets with Odd Stretch

Trading off with the existential result of [67], we get the following:

▶ **Theorem 25.** *Let $k \geq 1$ be an integer. When $\beta = \widetilde{O}(k^{-1/2} \cdot n^{\frac{1}{2} - \frac{1}{2k}})$, there is a polynomial-time $\widetilde{O}(\sqrt{k\beta} \cdot n^{\frac{1}{2} + \frac{1}{2k} + \epsilon})$-approximation for undirected stretch-$(2k-1)$* Generalized $\beta$-Hopset*, where $\epsilon > 0$ is an arbitrarily small constant.*

## References


1. Amir Abboud and Greg Bodwin. Reachability preservers: New extremal bounds and approximation algorithms. *SIAM Journal on Computing*, 53(2):221–246, 2024. doi:10.1137/21M1442176.
2. Amir Abboud, Greg Bodwin, and Seth Pettie. A hierarchy of lower bounds for sublinear additive spanners. *SIAM Journal on Computing*, 47(6):2203–2236, 2018. doi:10.1137/16M1105815.
3. Ingo Althöfer, Gautam Das, David P. Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discret. Comput. Geom.*, 9:81–100, 1993. doi:10.1007/BF02189308.
4. Maxim Babenko, Andrew V. Goldberg, Anupam Gupta, and Viswanath Nagarajan. Algorithms for hub label optimization. *ACM Trans. Algorithms*, 13(1), November 2016. doi:10.1145/2996593.

5 Uri Ben-Levy and Merav Parter. New ($\alpha$, $\beta$) spanners and hopsets. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1695–1714. SIAM, 2020. doi:10.1137/1.9781611975994.104.

6 Piotr Berman, Arnab Bhattacharyya, Konstantin Makarychev, Sofya Raskhodnikova, and Grigory Yaroslavtsev. Improved approximation for the directed spanner problem. In *Proceedings of the 38th International Colloquim Conference on Automata, Languages and Programming - Volume Part I*, ICALP'11, pages 1–12, Berlin, Heidelberg, 2011. Springer-Verlag. doi:10.1007/978-3-642-22006-7_1.

7 Aaron Bernstein, Maximilian Probst Gutenberg, and Thatchaphol Saranurak. Deterministic decremental sssp and approximate min-cost flow in almost-linear time. In *62 Annual IEEE Symposium on Foundatios of Computer Science (FOCS 2022)*, 2021.

8 Aaron Bernstein and Liam Roditty. Improved dynamic algorithms for maintaining approximate shortest paths under deletions. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 1355–1365. SIAM, 2011. doi:10.1137/1.9781611973082.104.

9 Aaron Bernstein and Nicole Wein. Closing the gap between directed hopsets and shortcut sets. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 163–182. SIAM, 2023. doi:10.1137/1.9781611977554.CH7.

10 Arnab Bhattacharyya, Elena Grigorescu, Kyomin Jung, Sofya Raskhodnikova, and David P. Woodruff. Transitive-closure spanners, 2008. arXiv:0808.1787.

11 Greg Bodwin and Gary Hoppenworth. Folklore sampling is optimal for exact hopsets: Confirming the $\sqrt{n}$ barrier. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 701–720. IEEE, 2023.

12 Greg Bodwin, Gary Hoppenworth, and Ohad Trabelsi. Bridge girth: A unifying notion in network design. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 600–648. IEEE, 2023. doi:10.1109/FOCS57990.2023.00043.

13 Nairen Cao and Jeremy T Fineman. Parallel exact shortest paths in almost linear work and square root depth. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4354–4372. SIAM, 2023. doi:10.1137/1.9781611977554.CH166.

14 Nairen Cao, Jeremy T Fineman, and Katina Russell. Efficient construction of directed hopsets and parallel approximate shortest paths. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 336–349, 2020. doi:10.1145/3357713.3384270.

15 Nairen Cao, Jeremy T Fineman, and Katina Russell. Improved work span tradeoff for single source reachability and approximate shortest paths. In *ACM Symposium on Parallelism in Algorithms and Architectures*, 2020.

16 Keren Censor-Hillel, Michal Dory, Janne H Korhonen, and Dean Leitersdorf. Fast approximate shortest paths in the congested clique. *Distributed Computing*, 34:463–487, 2021. doi:10.1007/S00446-020-00380-5.

17 Parinya Chalermsook, Yonggang Jiang, Sagnik Mukhopadhyay, and Danupon Nanongkai. Shortcuts and transitive-closure spanners approximation, 2025. doi:10.48550/arXiv.2502.08032.

18 Shiri Chechik. Near-optimal approximate decremental all pairs shortest paths. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 170–181. IEEE, 2018. doi:10.1109/FOCS.2018.00025.

19 Chandra Chekuri, Guy Even, Anupam Gupta, and Danny Segev. Set connectivity problems in undirected graphs and the directed steiner network problem. *ACM Trans. Algorithms*, 7(2), March 2011. doi:10.1145/1921659.1921664.

20 Li Chen, Rasmus Kyng, Yang P Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum flow and minimum-cost flow in almost-linear time. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 612–623. IEEE, 2022. doi:10.1109/FOCS54457.2022.00064.

21 Eden Chlamtác and Michael Dinitz. Lowest-degree k-spanner: Approximation and hardness. *Theory Comput.*, 12(1):1–29, 2016. doi:10.4086/TOC.2016.V012A015.

22 Eden Chlamtác, Michael Dinitz, Guy Kortsarz, and Bundit Laekhanukit. Approximating spanners and directed steiner forest: Upper and lower bounds. *ACM Trans. Algorithms*, 16(3):33:1–33:31, 2020. doi:10.1145/3381451.

23 Eden Chlamtac, Michael Dinitz, and Robert Krauthgamer. Everywhere-sparse spanners via dense subgraphs. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 758–767. IEEE Computer Society, 2012. doi:10.1109/FOCS.2012.61.

24 Eden Chlamtác, Michael Dinitz, and Thomas Robinson. Approximating the norms of graph spanners. In Dimitris Achlioptas and László A. Végh, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2019, September 20-22, 2019, Massachusetts Institute of Technology, Cambridge, MA, USA*, volume 145 of *LIPIcs*, pages 11:1–11:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICS.APPROX-RANDOM.2019.11.

25 Edith Cohen. Polylog-time and near-linear work approximation scheme for undirected shortest paths. *Journal of the ACM (JACM)*, 2000.

26 Edith Cohen, Eran Halperin, Haim Kaplan, and Uri Zwick. Reachability and distance queries via 2-hop labels. *SIAM Journal on Computing*, 32(5):1338–1355, 2003. doi:10.1137/S0097539702403098.

27 Emilio Cruciani, Sebastian Forster, Gramoz Goranci, Yasamin Nazari, and Antonis Skarlatos. Dynamic algorithms for k-center on graphs. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3441–3462. SIAM, 2024. doi:10.1137/1.9781611977912.123.

28 Erik D Demaine and Morteza Zadimoghaddam. Minimizing the diameter of a network using shortcut edges. In *Scandinavian Workshop on Algorithm Theory*, pages 420–431. Springer, 2010. doi:10.1007/978-3-642-13731-0_39.

29 Michael Dinitz, Ama Koranteng, and Yasamin Nazari. Approximation algorithms for optimal hopsets, 2025. doi:10.48550/arXiv.2502.06522.

30 Michael Dinitz, Guy Kortsarz, and Ran Raz. Label cover instances with large girth and the hardness of approximating basic $k$-spanner. *ACM Trans. Algorithms*, 12(2):25:1–25:16, 2016. doi:10.1145/2818375.

31 Michael Dinitz and Robert Krauthgamer. Directed spanners via flow-based linear programs. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*, STOC '11, pages 323–332, New York, NY, USA, 2011. Association for Computing Machinery. doi:10.1145/1993636.1993680.

32 Michael Dinitz and Yasamin Nazari. Massively parallel approximate distance sketches. *OPODIS*, 2019.

33 Michael Dinitz, Yasamin Nazari, and Zeyu Zhang. Lasserre integrality gaps for graph spanners and related problems. In *Approximation and Online Algorithms: 18th International Workshop, WAOA 2020, Virtual Event, September 9–10, 2020, Revised Selected Papers*, pages 97–112, Berlin, Heidelberg, 2020. Springer-Verlag. doi:10.1007/978-3-030-80879-2_7.

34 Michal Dory and Shaked Matar. Massively parallel algorithms for approximate shortest paths. In *Proceedings of the 36th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '24, pages 415–426, New York, NY, USA, 2024. Association for Computing Machinery. doi:10.1145/3626183.3659978.

35 Michael Elkin and Ofer Neiman. Near-optimal distributed routing with low memory. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*. ACM, 2018.

36 Michael Elkin and Ofer Neiman. Hopsets with constant hopbound, and applications to approximate shortest paths. *SIAM Journal on Computing*, 2019.

**37** Michael Elkin and Ofer Neiman. Linear-size hopsets with small hopbound, and constant-hopbound hopsets in rnc. In *The 31st ACM Symposium on Parallelism in Algorithms and Architectures*, pages 333–341, 2019. doi:10.1145/3323165.3323177.

**38** Michael Elkin and Ofer Neiman. Near-additive spanners and near-exact hopsets, a unified view. *arXiv preprint arXiv:2001.07477*, 2020. arXiv:2001.07477.

**39** Michael Elkin and David Peleg. The hardness of approximating spanner problems. *Theory Comput. Syst.*, 41(4):691–729, 2007. doi:10.1007/S00224-006-1266-2.

**40** Moran Feldman, Guy Kortsarz, and Zeev Nutov. Improved approximation algorithms for directed steiner forest. *Journal of Computer and System Sciences*, 78(1):279–292, 2012. doi:10.1016/J.JCSS.2011.05.009.

**41** Jeremy T Fineman. Nearly work-efficient parallel algorithm for digraph reachability. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 457–470, 2018. doi:10.1145/3188745.3188926.

**42** Elena Grigorescu, Nithish Kumar, and Young-San Lin. Approximation Algorithms for Directed Weighted Spanners. In Nicole Megow and Adam Smith, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023)*, volume 275 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 8:1–8:23, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.APPROX/RANDOM.2023.8.

**43** Elena Grigorescu, Nithish Kumar, and Young-San Lin. Directed buy-at-bulk spanners, 2024. doi:10.48550/arXiv.2404.05172.

**44** Elena Grigorescu, Nithish Kumar, and Young-San Lin. Multicriteria spanners – a new tool for network design, 2024. doi:10.48550/arXiv.2412.05526.

**45** Elena Grigorescu, Young-San Lin, and Kent Quanrud. Online directed spanners and steiner forests. In Mary Wootters and Laura Sanità, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2021, August 16-18, 2021, University of Washington, Seattle, Washington, USA (Virtual Conference)*, volume 207 of *LIPIcs*, pages 5:1–5:25. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICS.APPROX/RANDOM.2021.5.

**46** Maximilian Probst Gutenberg and Christian Wulff-Nilsen. Deterministic algorithms for decremental approximate shortest paths: Faster and simpler. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2522–2541. SIAM, 2020. doi:10.1137/1.9781611975994.154.

**47** Refael Hassin. Approximation schemes for the restricted shortest path problem. *Mathematics of Operations Research*, 17(1):36–42, 1992. doi:10.1287/MOOR.17.1.36.

**48** Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. Decremental single-source shortest paths on undirected graphs in near-linear total update time. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 146–155. IEEE, 2014. doi:10.1109/FOCS.2014.24.

**49** William Hesse. Directed graphs requiring large numbers of shortcuts. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 665–669, 2003. URL: http://dl.acm.org/citation.cfm?id=644108.644216.

**50** Gary Hoppenworth, Yinzhan Xu, and Zixuan Xu. New separations and reductions for directed hopsets and preservers. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4405–4443. SIAM, 2025. doi:10.1137/1.9781611978322.150.

**51** Markó Horváth and Tamás Kis. Multi-criteria approximation schemes for the resource constrained shortest path problem. *Optimization Letters*, 12(3):475–483, 2018. doi:10.1007/S11590-017-1212-Z.

**52** Shang-En Huang and Seth Pettie. Thorup–zwick emulators are universally optimal hopsets. *Information Processing Letters*, 142:9–13, 2019. doi:10.1016/J.IPL.2018.10.001.

**53** Shang-En Huang and Seth Pettie. Lower bounds on sparse spanners, emulators, and diameter-reducing shortcuts. *SIAM Journal on Discrete Mathematics*, 35(3):2129–2144, 2021. doi:10.1137/19M1306154.

**54** Arun Jambulapati, Yang P Liu, and Aaron Sidford. Parallel reachability in almost linear work and square root depth. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1664–1686. IEEE, 2019. doi:10.1109/FOCS.2019.00098.

**55** Philip N Klein and Sairam Subramanian. A randomized parallel algorithm for single-source shortest paths. *Journal of Algorithms*, 1997.

**56** Shimon Kogan and Merav Parter. Having hope in hops: New spanners, preservers and lower bounds for hopsets. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 766–777. IEEE, 2022. doi:10.1109/FOCS54457.2022.00078.

**57** Shimon Kogan and Merav Parter. New diameter-reducing shortcuts and directed hopsets: Breaking the barrier. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1326–1341. SIAM, 2022. doi:10.1137/1.9781611977073.55.

**58** Shimon Kogan and Merav Parter. Towards bypassing lower bounds for graph shortcuts. In *31st Annual European Symposium on Algorithms (ESA 2023)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023.

**59** Guy Kortsarz. On the hardness of approximating spanners. *Algorithmica*, 30(3):432–450, 2001. doi:10.1007/S00453-001-0021-Y.

**60** Guy Kortsarz and David Peleg. Generating sparse 2-spanners. *J. Algorithms*, 17(2):222–236, 1994. doi:10.1006/JAGM.1994.1032.

**61** Guy Kortsarz and David Peleg. Generating low-degree 2-spanners. *SIAM J. Comput.*, 27(5):1438–1456, 1998. doi:10.1137/S0097539794268753.

**62** Jakub Łącki and Yasamin Nazari. Near-optimal decremental hopsets with applications. In *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.

**63** Dean H. Lorenz and Danny Raz. A simple efficient approximation scheme for the restricted shortest path problem. *Oper. Res. Lett.*, 28(5):213–219, June 2001. doi:10.1016/S0167-6377(01)00069-4.

**64** Aleksander Madry. Faster approximation schemes for fractional multicommodity flow problems via dynamic graph algorithms. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 121–130, 2010. doi:10.1145/1806689.1806708.

**65** Gary L Miller, Richard Peng, Adrian Vladu, and Shen Chen Xu. Improved parallel algorithms for spanners and hopsets. In *Proceedings of the Symposium on Parallelism in Algorithms and Architectures*. ACM, 2015.

**66** Cynthia A. Phillips. The network inhibition problem. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '93, pages 776–785, New York, NY, USA, 1993. Association for Computing Machinery. doi:10.1145/167088.167286.

**67** Mikkel Thorup and Uri Zwick. Approximate distance oracles. *J. ACM*, 52(1):1–24, January 2005. doi:10.1145/1044731.1044732.

**68** Jeffery Ullman and Mihalis Yannakakis. High-probability parallel transitive closure algorithms. In *Proceedings of the second annual ACM symposium on Parallel algorithms and architectures*, pages 200–209, 1990.

**69** Arthur Warburton. Approximation of pareto optima in multiple-objective, shortest-path problems. *Oper. Res.*, 35:70–79, 1987. doi:10.1287/OPRE.35.1.70.

**70** Virginia Vassilevska Williams, Yinzhan Xu, and Zixuan Xu. Simpler and higher lower bounds for shortcut sets. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2643–2656. SIAM, 2024. doi:10.1137/1.9781611977912.94.

**71** Guoliang Xue, Weiyi Zhang, Jian Tang, and Krishnaiyan Thulasiraman. Polynomial time approximation algorithms for multi-constrained qos routing. *IEEE/ACM Transactions on Networking*, 16(3):656–669, 2008. doi:10.1109/TNET.2007.900712.