Directed Buy-At-Bulk Spanners

Elena Grigorescu
Department of Computer Science, University of Waterloo, Canada

Nithish Kumar
Department of Computer Science, Purdue University, West Lafayette, IN, USA

Young-San Lin
Melbourne Business School, Australia

Abstract

We present a framework that unifies directed buy-at-bulk network design and directed spanner problems, namely, *buy-at-bulk spanners*. The goal is to find a minimum-cost routing solution for network design problems that captures economies at scale, while satisfying demands and distance constraints for terminal pairs. A more restricted version of this problem was shown to be $O(2^{\log^{1-\varepsilon} n})$-hard to approximate, where n is the number of vertices, under a standard complexity assumption, by Elkin and Peleg (Theory of Computing Systems, 2007). Our results for buy-at-bulk spanners are the following.

1. When the edge lengths are *integral* with magnitude *polynomial* in n we present:
   a. An $\tilde{O}(n^{4/5+\varepsilon})$-approximation polynomial-time randomized algorithm for *uniform* demands.
   b. An $\tilde{O}(k^{1/2+\varepsilon})$-approximation polynomial-time randomized algorithm for general demands, where k is the number of terminal pairs. This can be improved to an $\tilde{O}(k^{\varepsilon})$-approximation algorithm for the single-source problem. The same approximation ratios hold in the online setting.
2. When the edge lengths are *rational* and *well-conditioned*, we present an $\tilde{O}(k^{1/2+\varepsilon})$-approximation polynomial-time randomized algorithm that may slightly violate the distance constraints. The result can be improved to an $\tilde{O}(k^{\varepsilon})$-approximation algorithm for the single-source problem. The same approximation ratios hold for the online setting when the condition number is given in advance.

To the best of our knowledge, these are the first sublinear factor approximation algorithms for directed buy-at-bulk spanners. We allow the edge lengths to be *negative* and the demands to be *non-unit*, unlike the previous literature. Our approximation ratios match the state-of-the-art ratios in special cases, namely, buy-at-bulk network design by Antonakopoulos (WAOA, 2010) and (online) weighted spanners by Grigorescu, Kumar, and Lin (APPROX 2023). Furthermore, we improve the competitive ratio for online buy-at-bulk by Chakrabarty, Ene, Krishnaswamy, and Panigrahi (SICOMP, 2018) by a factor of $\log R$, where R is the ratio between the maximum demand and the minimum demand.

2012 ACM Subject Classification Theory of computation → Online algorithms; Theory of computation → Routing and network design problems; Theory of computation → Rounding techniques

Keywords and phrases buy-at-bulk spanners, minimum density junction tree, resource constrained shortest path

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2025.22

Category APPROX

Related Version *Full Version*: `https://arxiv.org/pdf/2404.05172` [33]

Funding *Elena Grigorescu*: Supported in part by NSF CCF-1910659, NSF CCF-1910411, and NSF CCF-2228814.
Nithish Kumar: Partially supported by NSF CCF-1910411, NSF CCF-2228814, NSF Award CCF-2127806 and ONR Award N00014-24-1-2695.

1 Introduction

A core network connectivity problem is the *buy-at-bulk* problem [5, 7, 12, 14, 17, 37, 49, 55, 57], in which the goal is to route resources between pairs of source and destination locations. Each pair has an associated demand, i.e., the load of the resources to be delivered. To model economies of scale, each edge in the network is associated with a *cost* given by a subadditive function for the total load of resources delivered through an edge. A feasible solution to the problem is a collection of routes for each demand. The goal is to find a feasible solution that minimizes the overall cost.

The *spanner* problem [9–11, 18, 21, 22, 26, 27, 32, 46], on the other hand, is a fundamental network connectivity problem where each edge is assigned a *length*, and each terminal pair is associated with a *distance budget*. The goal is to find a minimum-size subgraph such that each pair is connected within its target distance.

Both the buy-at-bulk and spanner problems have been well-studied problems in their own right as they have a plethora of applications in theory and practice. One limitation of the buy-at-bulk problem is that it does not account for distance constraints. For example, although the buy-at-bulk cost formulation captures the economies of scale excellently in communication networks such as the Internet, it does not account for latency [1]. On the other hand, spanners can capture distance-constrained connectivity but do not account for economies of scale cost when the terminal pairs have various demands.

As a specific example, consider a situation where the city council wishes to modernize the city transportation network in order to minimize the carbon footprint of commuters. This could be captured by a buy-at-bulk formulation: if more commuters use a single transportation link, say a train, then it is natural to assume that the per-commuter carbon footprint will decrease. However, commuters also have their self-interest in mind, which may be a time constraint. How to minimize the carbon footprint while ensuring that the commuters don't experience significant delays?

Complex network design problems often need to be modeled by buy-at-bulk costs, while also satisfying spanner-like distance constraints, which leads to the following question:

How can we solve buy-at-bulk network design problems
while also satisfying spanner-like distance constraints?

Typical spanner problems have only dealt with positive edge lengths. However, one encounters natural applications in which the resource cost may be modeled as being *negative*, i.e., a resource gain. Consider a commuter who drives an electric car as part of his commute. This car gets recharged when going downhill, which may be captured by a gain in money when traveling on such an edge since recharging means spending less on a charging station. Hence, we may further ask:

How can we solve buy-at-bulk network design problems
with distance constraints and negative edge lengths?

Motivated by these questions, we study a general multi-commodity problem in directed graphs, namely, the *buy-at-bulk spanner* problem. We obtain the first results for this general formulation, which are also comparable with the best-known results from the literature on the two individual problems. We now proceed to formalize the buy-at-bulk spanner problem.

[1] Consider a situation where your service provider picks a solution that increases your latency by a factor of 10 just to save a few hundred dollars for them! The Buy-at-bulk formulation will always prefer the cheapest solution, even if it has very high latency. Note that cost and latency of an edge are both independent and possibly inversely related metrics. Consider for example using air travel (which is expensive but quick) as opposed to trains.

1.1 Buy-at-bulk spanners

In the buy-at-bulk spanner problem, we are given a directed simple graph $G = (V, E)$ with n vertices, and a set of k terminal pairs $D \subseteq V \times V$. Each pair $(s,t) \in D$ is associated with a *distance budget* given by the function $\textsc{Dis} : D \to \mathbb{R}$ and a *demand* given by the function $\textsc{Dem} : D \to \mathbb{Z}_{>0}$. When $\textsc{Dem}(s,t) = 1$ for all $(s,t) \in D$, we say that the problem has *unit* demands. Each edge $e \in E$ is associated with a *cost* given by a subadditive function $f_e : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$, satisfying $f_e(x+y) \leq f_e(x) + f_e(y)$ for all $x, y \geq 0$, and a length $\ell_e \in \mathbb{R}$. Unlike previous work, our work allows edge lengths to be *negative* which captures the notion of *gain* of a resource/commodity when traversing an edge [2].

A feasible solution to the problem is a collection of paths $\mathcal{P} := \{p(s,t)\}_{(s,t)\in D}$ where $p(s,t)$ is a feasible $s \rightsquigarrow t$ path i.e., if $\sum_{e \in p(s,t)} \ell_e \leq \textsc{Dis}(s,t)$. Let the *load* of edge e be $\textsc{Load}(e) := \sum_{(s,t)\in D : e \in p(s,t)} \textsc{Dem}(s,t)$. The goal is to find a feasible solution that minimizes the objective $\sum_{e \in E} f_e(\textsc{Load}(e))$.[3]

The buy-at-bulk spanner problem captures a wide range of network connectivity problems that are motivated by common scenarios, such as product delivery, transportation, electricity, and internet construction.

This general formulation has been studied under many variants: without distance constraints, it is equivalent to the *buy-at-bulk* problem [55]; when the edge cost is a fixed value once used, it captures the *weighted spanner* problem [32]. The weighted spanner problem is a generalization of spanners, distance preservers, and Steiner forests, which have found applicability in various domains such as multi-commodity network design [30,36], approximate shortest paths [8,24,25], distributed computation [6,52], and routing schemes [20,51,53,54].

1.1.1 A two-metric-based buy-at-bulk formulation

Previous work [5,12,17,40] reduces the buy-at-bulk problem to the *two-metric network design* problem, with only a constant factor loss in the approximation guarantee.

In this problem, each edge $e \in E$ has a one-time setup cost $\sigma(e)$ and a maintenance cost $\delta(e)$ [4]. The objective is to minimize the cost

$$\sum_{e \in \cup_{(s,t)\in D} p(s,t)} \sigma(e) + \sum_{(s,t)\in D} \sum_{e \in p(s,t)} \delta(e) \cdot \textsc{Dem}(s,t). \tag{1}$$

▶ **Lemma 1** ([17]). *Given any feasible solution with objective value OBJ_{BB} for the buy-at-bulk problem, there exists an instance of the two-metric network design problem that has a feasible solution with objective value OBJ_{2M}, such that $\mathit{OBJ}_{2M} \leq \mathit{OBJ}_{BB} \leq (2+\varepsilon)\mathit{OBJ}_{2M}$.*

We note that adding distance constraints does not affect the reduction. Let $\mathcal{D} \subseteq \mathbb{R}$ be the domain for the length of each edge. We consider the following problem throughout the paper with different options of $\mathcal{D}$.

[2] The distance budget can be negative. We further assume that there are *no negative length cycles* in G.
[3] We note while alternate formulations (such as one where the global cost is a constraint and the distance travelled by individual users is an objective) are also interesting, we aim to make our formulation here as close to previous literature on buy-at-bulk network design and spanners as possible.
[4] Previous literature uses the term *length* to refer to $\delta(e)$. In our case, $\delta(e)$ does not capture any local constraints; it can only handle a global objective. In the context of say the internet, $\delta(e)$ would be the maintenance cost of the internet due to the load; not the latency of individual users.

▶ **Definition 2.** *Buy-at-bulk Spanner on $\mathcal{D}$*

***Instance**: A directed graph $G = (V, E)$, where*

- *each edge $e \in E$ has an upfront cost $\sigma : E \to \mathbb{Q}_{\geq 0}$, a pay-per-use cost $\delta : E \to \mathbb{Q}_{\geq 0}$, and a distance $\ell_e \in \mathcal{D}$. We assume that there are no negative cycles induced by $\{\ell_e\}_{e \in E}$.*
- *We are given a set $D \subseteq V \times V$ of ordered pairs, where each pair has a demand captured by the function $\textsc{Dem} : D \to \mathbb{Z}_{\geq 0}$ and a distance budget captured by the function $\textsc{Dis} : D \to \mathbb{R} \setminus \{0\}$ [5]. Furthermore, we assume that there exists an $s \rightsquigarrow t$ path that satisfies the distance constraint for each $(s, t) \in D$.*

***Objective**: Find a collection of feasible $s \rightsquigarrow t$ paths $\mathcal{P} := \{p(s, t)\}_{(s,t) \in D}$ such that the cost (1) is minimized. An $s \rightsquigarrow t$ path $p(s, t)$ is* feasible *if $\sum_{e \in p(s,t)} \ell_e \leq \textsc{Dis}(s, t)$.*

The performance of an approximation algorithm is measured by the *approximation ratio*, the ratio between the cost of the approximate solution and that of the optimal solution.

For Buy-at-bulk Spanner, we consider two domains for edge lengths, $\mathcal{D} = [\mathsf{poly}(n)]_{\pm} := \{j \in \mathbb{Z} \mid |j| \leq \mathsf{poly}(n)\}$ and $\mathcal{D} = \mathbb{R}$. When the problem has unit demands, Buy-at-bulk Spanner is termed Unit-demand Buy-at-bulk Spanner. In a special case of Buy-at-bulk Spanner where the source vertex $s \in V$ is fixed, we call this problem Single-source Buy-at-bulk Spanner. For notation convenience, we have $D \subseteq \{s\} \times V$. We say that s is the *root* vertex. The definition for a single-sink buy-at-bulk spanner where the terminal pairs share the same sink is defined similarly.

In the *online* buy-at-bulk spanner problem, each terminal pair and its corresponding demand and budget arrive in an online fashion, one at a time. The other parts of the input, including the graph, edge costs, and edge lengths, are provided in advance. The goal is to irrevocably select a path to satisfy the terminal demand pair within the distance budget. We add the prefix Online to the problem of interest to denote the online version of that problem. The vertex pair (s_i, t_i) denotes the i-th pair that arrives online.

1.2 Our Contributions

To the best of our knowledge, we give the first efficient sublinear factor approximation algorithm for Buy-at-bulk Spanner on $[\mathsf{poly}(n)]_{\pm}$ and $\mathbb{R}$. Our online results even allow demands to be non-unit without paying an extra cost that depends on the demands. We allow edge lengths to be negative. To the best of our knowledge, negative edge lengths were not well-studied even for simpler cases such as spanners and restricted shortest paths.

In Section 1.2.1, we present our results for Buy-at-bulk Spanner. In Section 1.2.2 we present our results for Online Pairwise Spanner. In Sections 1.2.3 and 1.2.4, we present our technical tools for solving Buy-at-bulk Spanner, namely, *distance-constrained junction trees* and *resource-constrained shortest paths with negative consumption*, respectively.

1.2.1 Buy-at-bulk spanners

In Section 2, we prove the following for Unit-demand Buy-at-bulk Spanner on $[\mathsf{poly}(n)]_{\pm}$.

▶ **Theorem 3.** *For any constant $\varepsilon > 0$, there exists a polynomial-time randomized algorithm for Unit-demand Buy-at-bulk Spanner on $[\mathsf{poly}(n)]_{\pm}$ with approximation ratio $\tilde{O}(n^{4/5+\varepsilon})$ and the distance constraints for all $(s, t) \in D$ are satisfied with high probability.*[6]

[5] One workaround if we want to set a specific distance constraint as 0 is to set it to a small number that is close enough to 0 like say 10^{-c} (where $c > 0$) while ensuring the problem is well-conditioned.

[6] Throughout this paper, when we say high probability, we mean probability at least $1 - 1/n$.

Theorem 3 generalizes the $\tilde{O}(n^{4/5+\varepsilon})$-approximation algorithm for the unit-demand directed buy-at-bulk network design problem [5] and directed weighted spanners [32], by allowing distance constraints.

Next, we turn to online buy-at-bulk spanners. The performance of an online algorithm is measured by its *competitive ratio*, that is, the ratio between the cost of the online solution and that of an optimal offline solution. With non-unit demand, we show the following.

▶ **Corollary 4.** *For any constant $\varepsilon > 0$, there are polynomial-time randomized online algorithms for* Online Buy-at-bulk Spanner *on $[\mathsf{poly}(n)]_{\pm}$ with competitive ratio $\tilde{O}(k^{1/2+\varepsilon})$ and for* Online Single-source Buy-at-bulk Spanner *on $[\mathsf{poly}(n)]_{\pm}$ with competitive ratio $\tilde{O}(k^{\varepsilon})$ satisfying the distance constraints with high probability.*

Corollary 4 generalizes the $\tilde{O}(k^{1/2+\varepsilon})$-competitive algorithm for directed weighted spanners [32] and buy-at-bulk network design [5, 12], and the $\tilde{O}(k^{\varepsilon})$-competitive algorithm for the single-source version of these problems, by allowing general demands, pay-per-use cost, and distance constraints.

Without distance constraints, Online Buy-at-bulk Spanner captures the online buy-at-bulk problem. Set the distance of each edge to one and the distance budget of each terminal pair to n. The state-of-the-art results are $\tilde{O}(k^{1/2+\varepsilon}\log R)$-competitive for online buy-at-bulk and $\tilde{O}(k^{\varepsilon}\log R)$-competitive for online single-source buy-at-bulk. Here, R is the ratio between the maximum and minimum demand [12]. We show the following result for Online Buy-at-bulk Spanner and Online Single-source Buy-at-bulk Spanner on $[\mathsf{poly}(n)]_{\pm}$, which removes the dependency of R from the competitive ratio in the result of [12]. Note that R is independent of n or k and in practice could be very large.

Next, we consider Online Buy-at-bulk Spanner on $\mathbb{R}$ by slightly relaxing the distance constraints. This also allows us to obtain approximation algorithms for Online Buy-at-bulk Spanner on $[\mathsf{poly}(n)]_{\pm}$ in terms of k (Corollary 4). For notation convenience, we define some condition numbers. Let

$$\eta := \frac{|\min\{\min_{e\in E}\{\ell_e\}, 0\}|}{\min_{(s,t)\in D}\{|\textsc{Dis}(s,t)|\}} \text{ and } \xi := \frac{\max_{(s,t)\in D}\{|\textsc{Dis}(s,t)|\}}{\min_{(s,t)\in D}\{|\textsc{Dis}(s,t)|\}}. \tag{2}$$

Intuitively, η denotes the ratio between the magnitude of the most negative edge length and the smallest absolute value of the budget.[7] If edge lengths are all non-negative, then $\eta = 0$. Similarly, ξ denotes the ratio between the largest and the smallest absolute value of the budget. To accommodate a negative distance budget, we use the sgn function

$$\operatorname{sgn}(x) = \begin{cases} -1 & \text{if } x < 0, \\ 0 & \text{if } x = 0, \\ 1 & \text{if } x > 0. \end{cases}$$

For Online Buy-at-bulk Spanner on $\mathbb{R}$, we assume that the condition numbers η and ξ are given in advance. This is not necessary for the offline version of this problem or for the Online Buy-at-bulk Spanner on $[\mathsf{poly}(n)]_{\pm}$.

Suppose we are given a tolerance parameter $\theta > 0$. When the distance budget is positive, the goal is to satisfy the distance constraint within a factor of $(1+\theta)$. When the distance budget is negative, the distance between the terminal pair is below $(1-\theta)$ times the budget. We prove Theorem 6 in the full version paper [33].

[7] η cares about $\min_{e\in E}\{\ell_e\}$, but not about $\max_{e\in E}\{\ell_e\}$. This is because we can safely ignore edges that are much longer than the budget, but we cannot do so for edges (with negative lengths) that are much shorter than the budget.

▶ **Definition 5** (θ-Feasibility). *For a given $\theta > 0$, an $s \rightsquigarrow t$ path $p(s,t)$ is θ-feasible if $\sum_{e \in p(s,t)} \ell_e \leq (1 + \theta \cdot \operatorname{sgn}(\textsc{Dis}(s,t))) \cdot \textsc{Dis}(s,t)$. Note that when $\theta = 0$, θ-feasibility is no different from regular feasibility.*

▶ **Theorem 6.** *For any $\theta > 0$ and constant $\varepsilon > 0$, there are $\mathsf{poly}(n, 1/\theta, \eta, \xi)$-time randomized algorithms for Online Buy-at-bulk Spanner on $\mathbb{R}$ with competitive ratio $\tilde{O}(k^{1/2+\varepsilon})$ and for Online Single-source Buy-at-bulk Spanner on $\mathbb{R}$ with competitive ratio $\tilde{O}(k^{\varepsilon})$, both returning a set of θ-feasible paths $p(s,t)$ for each $(s,t) \in D$, with high probability. Here, η and ξ are the condition numbers defined in (2), respectively. When $1/\theta, \eta, \xi \in \mathsf{poly}(n)$, the algorithm runs in polynomial time.*

When the edge lengths are non-negative, $\eta = 0$, so the running time is $\mathsf{poly}(n, 1/\theta, \xi)$.

When the domain $\mathcal{D} = [\mathsf{poly}(n)]_{\pm}$, observe that it is sufficient to consider that $\textsc{Dis}(s,t) \in \mathsf{poly}(n)$. This in turn guarantees a $\mathsf{poly}(n)$ upper bound for ξ and η. This upper bound can be obtained by processing the input graph (even when we don't know the distance constraints) and therefore for Online Buy-at-bulk Spanner on $[\mathsf{poly}(n)]_{\pm}$, we don't need to know the condition numbers in advance. From Theorem 6, when the domain $\mathcal{D} = [\mathsf{poly}(n)]_{\pm}$, we can set $\theta = 1/\mathsf{poly}(n)$ (based on the upper bounds of ξ and η) such that the distance constraints are satisfied. This implies Corollary 4.

1.2.2 Online Pairwise Spanner

In one special case of Online Buy-at-bulk Spanner where for all edges, the upfront cost is one, the maintenance cost is zero, and the lengths are positive, the problem is termed Online Pairwise Spanner. Theorem 6 and Corollary 4 imply Corollaries 8 and 9, respectively, using the following lemma.

▶ **Lemma 7** ([34]). *For any constant $\varepsilon > 0$, an $\tilde{O}(k^{1/2+\varepsilon})$-competitive randomized online algorithm with running time T for Online Pairwise Spanner implies an $\tilde{O}(n^{2/3+\varepsilon})$-competitive randomized online algorithm for Online Pairwise Spanner with running time $T + \mathsf{poly}(n, \log(\max_{e \in E}\{|\ell_e|\}))$.*

▶ **Corollary 8.** *For any constant $\varepsilon > 0$, there exists a $\mathsf{poly}(n, 1/\theta, \xi, \log(\max_{e \in E}\{|\ell_e|\}))$-time randomized algorithm for Online Pairwise Spanner with competitive ratio $\tilde{O}(n^{2/3+\varepsilon})$ returning a set of θ-feasible paths $p(s,t)$ for each $(s,t) \in D$, with high probability. Hernumbere, ξ is the condition numbers defined in (2). When $1/\theta, \xi \in \mathsf{poly}(n)$, the algorithm runs in polynomial time.*

▶ **Corollary 9.** *For any $\theta > 0$ and constant $\varepsilon > 0$, there exists a polynomial-time randomized algorithm for Online Pairwise Spanner on $[\mathsf{poly}(n)]$ with competitive ratio $\tilde{O}(n^{2/3+\varepsilon})$ satisfying the distance constraints with high probability.*

Corollary 8 is comparable with the $\tilde{O}(n^{4/5+\varepsilon})$-competitive result for Online Pairwise Spanner and the $\tilde{O}(n^{2/3+\varepsilon})$-competitive result for Online Pairwise Spanner with unit edge lengths [34]. The improvement in the competitive ratio and the domain of the edge lengths has a trade-off of slightly violating the distance constraints. Corollary 9 generalizes the result for Online Pairwise Spanner from unit edge lengths to edge lengths in $[\mathsf{poly}(n)]$.

We emphasize again that Buy-at-bulk Spanner unifies the buy-at-bulk network design and spanners problems. Our results expand the domain for some results in the existing literature on these two problems. We allow edge lengths to be negative and of arbitrary magnitude (provided they are well-conditioned). There are no directed spanner results that account for edge lengths with negative values and arbitrary magnitude that we are aware of.

1.2.3 Distance-constrained junction trees (Main technical results)

In previous literature, one main engine for solving the directed pairwise spanner problem [18, 34], the directed buy-at-bulk network design problem [5, 12, 17, 56], and the directed Steiner forest problem [9, 13, 16, 28] is the notion of *junction tree*. The notion of junction tree used for these problems is a union of an in-arborescence and an out-arborescence both rooted at the same vertex. For our purpose, we extend the definition of junction trees to handle both buy-at-bulk costs and distance constraints.

To further extend our results to the domain $\mathbb{R}$ for edge lengths, we define and use a relaxed version of distance-constrained junction trees. This allows us to obtain approximation algorithms for (θ-RELAXED) MINIMUM-DENSITY DISTANCE-CONSTRAINED JUNCTION TREE (Theorem 12 and Corollary 13) later on.

▶ **Definition 10.** *Given an instance of BUY-AT-BULK SPANNER, a root vertex $r \in V$ and a constant $\theta > 0$, a* θ-relaxed distance-constrained junction tree $\mathcal{J} := \{p(s,r,t)\}_{(s,t)\in D'}$ *rooted at r is a non-empty collection of θ-feasible $s \rightsquigarrow r \rightsquigarrow t$ paths in G for some $D' \subseteq D$. The $s \rightsquigarrow r$ paths form an in-arborescence and the $r \rightsquigarrow t$ paths form an out-arborescence, both rooted at r.*

The crucial subproblem for solving BUY-AT-BULK SPANNER on $\mathbb{R}$ is defined as follows.

▶ **Definition 11.** *θ-RELAXED MINIMUM-DENSITY DISTANCE-CONSTRAINED JUNCTION TREE*
***Instance**: Same as BUY-AT-BULK SPANNER on $\mathbb{R}$.*
***Objective**: Find a root $r \in V$, a distance-constrained junction tree $\mathcal{J}$ rooted at r, such that the ratio of the cost (1) of $\mathcal{J}$ to the number of (s,t) pairs that satisfy the θ-feasibility requirement (recall Definition 5) is minimized. Specifically, the goal is the following:*

$$\min_{r\in V,\mathcal{J}} \frac{\mathit{cost}(\mathcal{J})}{|\{(s,t)\in D \mid \exists \text{ a } \theta\text{-feasible path } p(s,r,t) \text{ in } \mathcal{J}|}. \tag{3}$$

▶ **Theorem 12** (Main Theorem). *For any constant $\varepsilon > 0$, there is a $\mathsf{poly}(n, 1/\theta, \eta, \xi)$-time randomized algorithm that gives an $\tilde{O}(k^{\varepsilon})$-approximation for θ-RELAXED MINIMUM-DENSITY DISTANCE-CONSTRAINED JUNCTION TREE with high probability. When $1/\theta, \eta, \xi \in \mathsf{poly}(n)$, the algorithm runs in polynomial time.*

When the edge lengths are non-negative, $\eta = 0$, so the running time is $\mathsf{poly}(n, 1/\theta, \xi)$.

When the domain $\mathcal{D} = [\mathsf{poly}(n)]_{\pm}$ and it is required to exactly satisfy the distance constraints, the problem θ-RELAXED MINIMUM-DENSITY DISTANCE-CONSTRAINED JUNCTION TREE is termed MINIMUM-DENSITY DISTANCE-CONSTRAINED JUNCTION TREE. From Theorem 12, we can set $\theta = 1/\mathsf{poly}(n)$ (for a sufficiently large $\mathsf{poly}(n)$) such that the distance constraints are satisfied and the condition numbers η and ξ are polynomial in n (since it is sufficient to consider that $\textsc{Dis}(s,t) \in \mathsf{poly}(n)$). This implies the following.

▶ **Corollary 13.** *For any constant $\varepsilon > 0$, there is a polynomial-time randomized algorithm that gives an $\tilde{O}(k^{\varepsilon})$-approximation for MINIMUM-DENSITY DISTANCE-CONSTRAINED JUNCTION TREE with high probability.*

Corollary 13 generalizes the minimum density junction tree used for buy-at-bulk network design [5], pairwise spanners [18], and weighted spanners [32], with the same approximation ratio. We emphasize that our minimum-density junction tree framework accounts for buy-at-bulk costs and distance constraints with negative edge lengths. Furthermore, since junction trees and their variants are used in several algorithms in network design, we believe

Theorem 12 and Corollary 13 which extend the junction tree black box in multiple ways are of independent interest. For instance, Corollaries 8 and 9 which improve existing results for Online Pairwise Spanner are obtained by simple application of our improved blackbox. An interesting question to ask is if Theorem 12 can be used in the same way to extend other such results (see for instance [18]).

1.2.4 Resource-constrained shortest paths with negative consumption

In previous literature, one crucial case analysis for solving directed spanner problems [9, 18, 21, 32, 34] and the directed buy-at-bulk network design problem [5] is via *flow-based* linear programs (LP). The LP formulations for spanners potentially contain an exponential number of constraints and thus require an efficient black-box subroutine to be solved in polynomial time. A fully polynomial time approximation scheme (FPTAS) for the *restricted shortest path problem* [43, 47] is treated as an approximate separation oracle to approximately solve the flow-based LP for spanners.

We design and use an approximation scheme for the resource-constrained shortest paths problem with *negative* resource consumption, a generalization of the FPTAS for the restricted shortest path problem [44] in order to further accommodate buy-at-bulk costs for spanners with negative edge lengths. We believe this generalization is of independent interest since it can be used to extend the results which use [43, 44, 47].

▶ **Definition 14.** *Resource-constrained Shortest Path (m-RCSP)*

***Instance**: An n-vertex directed graph $G = (V, E)$, with edge costs $c : E \to \mathbb{R}_{\geq 0}$, a terminal pair (s, t) with $s, t \in V$, and a resource budget $\boldsymbol{L} = (L_1, ..., L_m) \in (\mathbb{R} \setminus \{0\})^m$. For each edge $e \in E$, we also have a resource consumption vector $\boldsymbol{w(e)} = (w_1(e), w_2(e), \ldots, w_m(e)) \in \mathbb{R}^m$. We assume that there are no negative cycles induced by $\{w_i(e)\}_{e \in E}$ for each $i \in [m]$.*

***Objective**: Find a min-cost $s \rightsquigarrow t$ path P such that $\sum_{e \in P} w_i(e) \leq L_i$, $\forall i \in [m]$.*

We note that when $m = 1$ and the resource consumption is non-negative, m-RCSP captures the restricted shortest path problem. In the LP for Buy-at-bulk Spanner, the constraints can be captured by solving m-RCSP. For m-RCSP, we design an algorithm that finds an optimal solution that slightly violates the resource budget.

▶ **Definition 15** ($\boldsymbol{\mathcal{E}}$-Multicriteria-Feasibility)**.** *Let $\boldsymbol{\mathcal{E}} = (\varepsilon_1, \varepsilon_2 ..., \varepsilon_m) \in \mathbb{R}^m_{>0}$ be a given error tolerance vector. Given a terminal pair (s, t), an $s \rightsquigarrow t$ path $p(s, t)$ is ε_i-feasible for resource $i \in [m]$ if $\sum_{e \in p(s,t)} w_i(e) \leq (1 + \varepsilon_i \cdot \text{sgn}(L_i))L_i$. An $s \rightsquigarrow t$ path $p(s, t)$ is $\boldsymbol{\mathcal{E}}$-multicriteria-feasible if it is ε_i-feasible for all resources $i \in [m]$.*

Let OPT_{RCSP} be the cost of any minimum cost $s \rightsquigarrow t$ path that satisfies the resource constraints. Given a tolerance vector $(\varepsilon_1, \varepsilon_2 ..., \varepsilon_m) \in \mathbb{R}^m_{>0}$, a $(1; 1 + \varepsilon_1, 1 + \varepsilon_2 ..., 1 + \varepsilon_m)$-approximation scheme finds a $\boldsymbol{\mathcal{E}}$-multicriteria-feasible (see definition 15) $s \rightsquigarrow t$ path whose cost is at most OPT_{RCSP}. Let the condition number

$$\gamma_i := \frac{|\min\{\min_{e \in E}\{w_i(e)\}, 0\}|}{|L_i|} \tag{4}$$

denote the ratio between the magnitude of the most negative i-th resource consumption among the edges and the absolute value of the budget for the i-th resource.

▶ **Theorem 16.** *There exists a $\mathsf{poly}(n^m, \gamma_1, ..., \gamma_m, 1/|\varepsilon_1|, ..., 1/|\varepsilon_m|)$-time $(1; 1+\varepsilon_1, 1+\varepsilon_2 ..., 1+\varepsilon_m)$-approximation scheme for m-RCSP. When $\gamma_i, 1/|\varepsilon_i| \in \mathsf{poly}(n)$ $\forall i \in [m]$ and m is a constant, the approximation scheme runs in polynomial time.*

When $w_i(e) > 0$ $\forall i \in [m], e \in E$, Theorem 16 directly recovers the FPTAS from [44].

1.2.5 Summary

We summarize our main results for Buy-at-bulk Spanner in Table 1 by listing the approximation ratios and contrasting them with the corresponding known approximation ratios. The running time for Buy-at-bulk Spanner and Single-source Buy-at-bulk Spanner on $[\mathsf{poly}(n)]_{\pm}$ is polynomial. The running time for the θ-feasible or relaxed results on $\mathbb{R}$ is $\mathsf{poly}(1/\theta, \eta, \xi, n)$ ($\mathsf{poly}(n, 1/\theta, \xi, \log(\max_{e\in E}\{|\ell_e|\}))$ for Online Pairwise Spanner).

Table 1 Summary of the approximation and competitive ratios. Here, n refers to the number of vertices and k refers to the number of terminal pairs. The edge costs σ and δ are non-negative rational numbers. Input graphs do not have negative cycles induced by the edge lengths (or resources for m-RCSP). R is the ratio between the maximum and minimum demand. The weighted spanner problem [32] considers edge lengths in $[\mathsf{poly}(n)]$.

Problem	Our Results	Previous Problems and Results
Buy-at-bulk Spanner on $[\mathsf{poly}(n)]_{\pm}$	$\tilde{O}(n^{4/5+\varepsilon})$ (unit-demand, Thm 3) $\tilde{O}(k^{1/2+\varepsilon})$ (Cor 4) $\tilde{O}(k^{\varepsilon})$ (single-source, Cor 4)	$O(n^{4/5+\varepsilon})$ (unit-demand buy-at-bulk) [5] $\tilde{O}(n^{4/5+\varepsilon})$ (weighted spanners) [32] $O(k^{1/2+\varepsilon})$ (buy-at-bulk) [5] $O(k^{\varepsilon})$ (single-source buy-at-bulk) [5] $\tilde{O}(k^{1/2+\varepsilon})$ (weighted spanners) [32] $\tilde{O}(k^{\varepsilon})$ (single-source weighted spanners) [32] $\tilde{O}(n^{3/5+\varepsilon})$ (pairwise spanners) [18]
Buy-at-bulk Spanner on $\mathbb{R}$	$\tilde{O}(k^{1/2+\varepsilon})$ (θ-feasible, Thm 6) $\tilde{O}(k^{\varepsilon})$ (θ-feasible, single-source, Thm 6)	
Online Buy-at-bulk Spanner on $[\mathsf{poly}(n)]_{\pm}$	$\tilde{O}(k^{1/2+\varepsilon})$ (Cor 4) $\tilde{O}(k^{\varepsilon})$ (single-source, Cor 4)	$\tilde{O}(n^{4/5})$ (online pairwise spanners) [34] $\min\{\tilde{O}(k^{1/2+\varepsilon}), \tilde{O}(n^{2/3+\varepsilon})\}$ (online spanners with unit edge lengths) [34] $O(k^{1/2+\varepsilon}\log R)$ (online buy-at-bulk) [12] $O(k^{\varepsilon}\log R)$ (online single-source buy-at-bulk) [12] $\tilde{O}(k^{1/2+\varepsilon})$ (online weighted spanners) [32] $\tilde{O}(k^{\varepsilon})$ (online single-source weighted spanners) [32]
Online Buy-at-bulk Spanner on $\mathbb{R}$	$\tilde{O}(k^{1/2+\varepsilon})$ (θ-feasible, Thm 6) $\tilde{O}(k^{\varepsilon})$ (θ-feasible, single-source, Thm 6)	
Online Pairwise Spanner	$\tilde{O}(n^{2/3+\varepsilon})$ (θ-feasible, Cor 8) $\tilde{O}(n^{2/3+\varepsilon})$ (on $[\mathsf{poly}(n)]$, Cor 9)	
Minimum-density Distance-constrained Junction Tree	$\tilde{O}(k^{\varepsilon})$ (on $[\mathsf{poly}(n)]_{\pm}$, Cor 13) $\tilde{O}(k^{\varepsilon})$ (on $\mathbb{R}$, θ-relaxed, Thm 12)	$O(k^{\varepsilon})$ (buy-at-bulk) [5] $\tilde{O}(k^{\varepsilon})$ (pairwise spanners) [18] $\tilde{O}(k^{\varepsilon})$ (weighted spanners) [32]
m-RCSP with negative resource consumption	an $(1; 1+\varepsilon_1, 1+\varepsilon_2 ..., 1+\varepsilon_m)$-approximation scheme (Thm 16)	an $(1; 1+\varepsilon, 1+\varepsilon ..., 1+\varepsilon)$-approximation scheme for m-RCSP with non-negative consumption

1.3 High-level technical overview

1.3.1 The $\tilde{O}(k^{\varepsilon})$-approximation for Minimum-density Distance-constrained Junction Tree

We now sketch the proof of Theorem 12. This is our main technical result and an important tool for deriving many of our other results, i.e., Theorems 3 and Corollary 8.

At a high level, junction trees form a cheap partial solution that connects a subset of terminal pairs. For Buy-at-bulk Spanner, a potential approach is to iteratively select a low-density junction tree in case there exist edges that are crucial for connecting terminal pairs. For our purpose, we have to construct *feasible* junction trees that satisfy the distance constraints while connecting terminal pairs. The modified junction-tree-based approach is the main engine of our framework.

Suppose we have a fixed root vertex $r \in V$, and the goal is to find a minimum-density junction tree rooted at r. Here, the *density* is defined as the cost of the junction tree divided by the number of terminal pairs connected. The framework in [18] used for pairwise spanners

with unit lengths has three main steps: 1) construct a layered graph from G to capture the distance constraints and a junction tree rooted at r, 2) use the *height reduction* technique to construct a tree-like graph from the layered graph by paying a small approximation ratio, and 3) solve a corresponding *minimum density Steiner label cover* problem on the tree-like graph. The main reason for the second and third steps is that the tree-like graph is well-structured, which allows one to find a low-density junction tree by paying a polylogarithmic factor.

To capture distance constraints with negative edge lengths and the flow-based cost in the buy-at-bulk problem, our first two steps (scaling the edge lengths and building the layered graph) require multiple new ideas, which significantly depart from the analysis of [18] in several aspects, as we describe below. Although some of our ideas are natural, we do not believe they are obvious beforehand.

1.3.1.1 Scaling and rounding the edge lengths

In our first step, we use an approach similar to [44] to properly scale and round the edge lengths. The edge lengths are rounded up so that the distances between the terminal pairs might be slightly overestimated. This allows us to construct a layered graph of size polynomial in the condition numbers which approximately preserves the edge lengths and terminal distances. Note that this scaling process has been used only in the single commodity setting [44, 47]. Using this technique in the multi-commodity setting creates a new set of challenges as the scaling parameters depend on the distance constraint and in the multi-commodity setting we have multiple distance constraints. Our techniques effectively adapt the scaling process to the multi-commodity setting without paying a significant overhead.

1.3.1.2 Turning distance constraints into connectivity constraints

In our second step, we construct a layered graph that approximately preserves the edge lengths. In the layered graph, each layer captures the distance to (from) the root vertex. To handle negative edge lengths, a key modification is to allow edges to go *backward*, instead of always forward as in [18]. Furthermore, since we have general edge lengths instead of unit edge lengths, it is no longer the case that only neighboring layers have edges between them. Edges are added from one layer to another whenever their length corresponds to the distance between layers. Finally, one also needs to be careful in building the layers themselves – our distances are not bound by the maximum number of hops and thus we need more layers (it is for this reason we need scaling - otherwise we would have too many layers).

1.3.1.3 Handling pay-per-use cost

In our third step, the main challenge is to handle the flow-based cost properly. Fortunately, following the height reduction technique in [12] allows us to reduce the two-metric problem to a variant of the Steiner problem, namely, *minimum density Steiner label cover*, which only accounts for the upfront cost. In this problem, the goal is to find a minimum density subgraph that connects *pairs of terminal vertex sets*, subject to a relation induced by the distance constraints. This process also needs to be able to handle general demands (as opposed to [12] which only handles uniform demands). We use the algorithm in [18] to solve the minimum density Steiner label cover problem.

1.3.2 The online algorithms for Buy-at-bulk Spanner

We now present a sketch of the proof of Theorem 6. This is our approximation algorithm for Online Buy-at-bulk Spanner and it removes the dependency on R (i.e., the ratio between the maximum and minimum demand) from previous literature.

In the offline setting, an $\tilde{O}(k^{\varepsilon})$-approximation algorithm for Single-source Buy-at-bulk Spanner directly follows by Theorem 12. An offline approximation algorithm for Buy-at-bulk Spanner follows a standard iterative density procedure for directed Steiner forests [16] and spanners [32, 34]. Iteratively picking minimum-density distance-constrained junction trees only pays a factor of $O(\sqrt{k})$. Combining this and Theorem 12 results an offline $\tilde{O}(k^{1/2+\varepsilon})$-approximation. Finding an online solution requires several technical modifications. Instead of using an iterative procedure, which inherently uses the global structure of the problem in the offline setting, we construct junction trees in an online fashion similar to online buy-at-bulk [12], by considering all vertices as the candidate junction tree root simultaneously. Before terminal pairs arrive, we pre-process the input graph as follows. To handle distance constraints, we first construct a collection of layered graphs that approximately preserves the edge lengths and captures the distance to (from) each candidate root vertex of the junction tree. Then we use the height reduction technique to construct a forest, which is well-structured for deriving a reduction to the *online Steiner label cover* problem on the forest. An arriving terminal pair introduces a partial input of the online Steiner label cover problem on the forest, which can be approximately solved online [34]. Ultimately, we recover a solution in the original graph from the Steiner label cover solution in the forest.

1.3.3 The $\tilde{O}(n^{4/5+\varepsilon})$-approximation algorithm for Buy-at-bulk Spanner on $[\mathbf{poly}(n)]_{\pm}$

We now describe the high-level overview of the proof of Theorem 3. This result gives a sublinear approximation for Buy-at-bulk Spanner. The overall strategy here is to classify all terminal pairs as good or bad based on some parameters and tackle them separately. This approach is followed by multiple publications in the past such as [9, 21, 28, 32]. But all of these results have key differences in their details which give rise to significant differences in their results. Our proof of Theorem 3 also has several seemingly minor changes which require delicate incorporation to achieve our general result.

Similarly to the Steiner forest framework [9, 28] and the buy-at-bulk network design framework [5], we classify the terminal pairs as follows:

- A pair $(s,t) \in D$ is *good* if the feasible (i.e., satisfying the distance constraint) and cheap (low upfront cost and low pay-per-use cost) $s \rightsquigarrow t$ paths span a great number of vertices.
- A pair $(s,t) \in D$ is *bad* otherwise.

Different from the previous literature, our main technical challenge is to accommodate distance constraints. With distance constraints, we have to handle *all* the cases carefully, without destroying the desired structural properties. The analysis significantly departs from [5, 9, 28, 32] in several aspects, as we describe below.

1.3.3.1 Handling good pairs

For the first case, a standard approach is to sample a sufficient number of intermediary vertices from V and add cheap and feasible paths to connect the good pairs. For directed Steiner forests, edges only have upfront cost and there are no distance constraints, so it is sufficient to add cheap paths [9, 28]. For weighted spanners [32], edges only have upfront cost and positive edge lengths, so it is sufficient to find a *shortest cheap path* by using the restricted shortest path FPTAS [43, 47] as a subroutine.

For Buy-at-bulk Spanner on [poly(n)], edges have positive edge lengths, upfront cost, and pay-per-use cost, one can carefully use the resource-constrained shortest path (RCSP) FPTAS from [44] as the subroutine to handle distance constraints. This approach connects the sources to the intermediary vertices and the intermediary vertices to the sinks, where the connecting paths have short distances and *low upfront cost and pay-per-use cost*. Handling negative edge lengths requires a more general subroutine to connect the terminal vertices to (from) the intermediary vertices, so we modify the FPTAS for RCSP to accommodate negative edge lengths (described in more detail in Section 1.3.4). We note that the process of using the FPTAS we design isn't straightforward either and requires some careful observations.

1.3.3.2 Handling bad pairs

After the good pairs are resolved, we iteratively use a greedy algorithm based on a density argument. The greedy algorithm constructs a low-density partial solution in each iteration. Adding low-density partial solutions iteratively guarantees a global solution of approximately minimum cost. For ease of our analysis, we partition the bad pairs into three classes based on an unknown optimal solution $\mathcal{P}^* = \{p^*(s,t)\}_{(s,t)\in D}$.

- A bad pair $(s,t) \in D$ is in *class 1* if $p^*(s,t)$ has a high pay-per-use cost.
- A bad pair $(s,t) \in D$ is in *class 2* if $p^*(s,t)$ has a high upfront cost and a low pay-per-use cost.
- A bad pair $(s,t) \in D$ is in *class 3* if $p^*(s,t)$ has low upfront cost and low pay-per-use cost.

We consider three subcases for the bad pairs.

- Case 1: the number of bad pairs is small.
- Case 2: the number of class 2 bad pairs is larger than the number of class 3 bad pairs.
- Case 3: the number of class 2 bad pairs is at most the number of class 3 bad pairs.

The iterative procedure continues by picking the better solution by trying out case 2 and case 3 until the number of bad pairs is small (case 1). Once we reach case 1, we use Corollary 4 to resolve all the bad pairs. Since the number of unresolved vertex pairs is small in this case, the approximation ratio is sufficiently small.

A key observation is that the number of class 1 bad pairs is always smaller than the threshold used for case 1. Therefore, it suffices to consider cases 2 and 3 before we run out of class 2 and class 3 bad pairs. For case 2, there are more class 2 pairs than class 3 pairs. Since the class 2 bad pairs have high upfront cost, we must have an edge that belongs to a sufficient number of paths that connect the terminal pairs within the required distance in an optimal solution. This implies that there must be a vertex that lies on an edge used plenty of times, so we can use Corollary 13 (obtained from Theorem 12) to find a low-density junction tree rooted at that vertex as our partial solution.

Recall that for each bad pair, the number of vertices spanned by its feasible paths is small. In case 3, there are more class 3 pairs than class 2 pairs. Therefore, there are sufficient terminal pairs whose upfront cost and pay-per-use cost are low and whose feasible paths span a small number of vertices. A natural approach that addresses this case is via a flow-based LP formulation. Intuitively, when we have a large number of terminal pairs with low upfront cost and pay-per-use cost and a small number of vertices spanned by feasible paths, the edge indicators of the LP must be large enough to construct a feasible fractional solution. To solve the LP, we use our RCSP framework that accommodates negative resource consumption as a separation oracle to extract violating constraints. After using a careful pruning procedure as in [5] and a rounding scheme similar to the ones used for Steiner forests [9] and weighted spanners [32], we extract the edges with high indicator values in the LP and recover the collection of paths as our partial solution.

1.3.4 The approximation scheme for RCSP with negative consumption

We describe the proof of Theorem 16. This is our approximation algorithm for Resource-constrained Shortest Path and it allows edge lengths to be negative (as opposed to [43, 44, 47]). We use this result in the proof of Theorem 3. We also believe it is of independent interest since it can be used to extend the results which use [43, 44, 47]).

Recall that for m-RCSP, we are given a directed graph with a terminal vertex pair. Each edge is associated with a cost and an m-dimensional resource consumption vector where entries can be negative. There is no negative cycle induced by any resource type. We also have an m-dimensional budget for the resource consumption. The goal is to find a cheap path to connect the terminal pair without exceeding the budget.

To construct the approximation scheme, we follow the dynamic programming (DP) paradigm in [44]. To approximately preserve feasibility, [44] scales the resource consumption properly, and the DP memorizes the feasible resource consumption *patterns* while reaching a specific vertex from the source. To accommodate negative resource consumption, the main challenge is the negative resource consumption can be *unbounded* in terms of the scale for the non-negative consumption. Addressing these challenges requires several modifications. Not only do we construct a larger DP table, this DP table needs a different algorithm since the old algorithm fails when the weights are allowed to be negative. With the assumption that negative cycles do not exist, our DP needs to consider *hop counts* in a fashion similar to the Bellman-Ford algorithm. The number of resource consumption patterns in our setting depends on the condition numbers γ_i and this requires some careful analysis.

1.4 Related work

1.4.1 Directed spanners

A well-studied variant of spanners is called the *directed s-spanner* problem, where there is a fixed value $s \geq 1$ called the *stretch*, and the goal is to find a subgraph with a minimum number of edges such that the distance between *every* pair of vertices is preserved within a factor of s in the original graph. When the lengths of the edges are uniform and $s = 2$, there is a tight $\Theta(\log n)$-approximation algorithm [26, 46]. When $s = 3, 4$ there are $\tilde{O}(n^{1/3})$-approximation algorithms [9, 22]. When $s > 4$, the best known approximation factor is $\tilde{O}(n^{1/2})$ [9]. The problem is hard to approximate within an $O(2^{\log^{1-\varepsilon} n})$ factor for $3 \leq s = O(n^{1-\delta})$ and any $\varepsilon, \delta \in (0, 1)$, unless $NP \subseteq DTIME(n^{\text{polylog}\, n})$ [27]. More general variants consider the *pairwise spanner* problem [18], and the *client-server* model [10, 26], where the set of terminals is arbitrary $D = \{(s_i, t_i) \mid i \in [k]\} \subseteq V \times V$, and each terminal pair (s_i, t_i) has its own target distance d_i. The goal is to compute a minimum cardinality subgraph in which for each i, the distance from s_i to t_i is at most d_i. For the pairwise spanner problem with uniform lengths, [18] obtains an $\tilde{O}(n^{3/5+\varepsilon})$ approximation. Recently, [32] studied the *weighted* spanner problem for arbitrary terminal pairs, which has a closer formulation to buy-at-bulk spanners. [34] studied online directed spanners. We refer the reader to the excellent survey [2] for a more comprehensive exposition.

1.4.2 Buy-at-bulk network design

The buy-at-bulk problem has received considerable attention and has been well-studied in the past few decades. Most of the previous buy-at-bulk literature focused on undirected networks, as listed below.

The problem was first introduced in [55], where the subadditive load function was used to capture the economy of scale in network design. [55] showed that the problem is NP-hard, and gave an $O(\min\{\log n, \log D_{\max}\})$-approximation algorithm for the single-source problem, where $D_{\max}$ is the maximum demand. When the edge costs are uniform, there is a $\mathsf{polylog}(n)$-approximation algorithm for the multi-commodity problem [7] and $O(1)$-approximation algorithms for the single-source problem [35,37,57]. With non-uniform load functions, the first nontrivial result for the multi-commodity problem was $O(\log D_{\max} \exp(O(\sqrt{\log n \log\log n})))$-approximate [14], later improved to $\mathsf{polylog}(n)$-approximate [17]; for the single-source problem, there is an $O(\log k)$-approximate algorithm [49]. The buy-at-bulk problem is more intractable on directed graphs. The state-of-the-art is an $\min\{\tilde{O}(k^{1/2+\varepsilon}), \tilde{O}(n^{4/5+\varepsilon})\}$-approximate algorithm [5]. Even in the special case of directed Steiner forests, previous algorithms only gave $\mathsf{poly}(n)$ but sublinear (i.e., $o(n)$) approximation algorithms [1,9,18,28]. On the hardness side, for the undirected buy-at-bulk problem, the multi-commodity problem is $O(\log^{1/2-\varepsilon} n)$-hard to approximate when the costs are general and $O(\log^{1/4-\varepsilon} n)$-hard to approximate when the costs are uniform [4], and the single-source problem is $O(\log\log n)$-hard to approximate [19]. These results are based on the assumption that $NP \nsubseteq ZTIME(n^{\mathsf{polylog}(n)})$. For directed buy-at-bulk, the problem is hard to approximate within an $O(2^{\log^{1-\varepsilon} n})$ factor assuming that $NP \nsubseteq DTIME(n^{\mathsf{polylog}(n)})$, even in the special case of directed Steiner forests [23].

Note that previous literature, including [12, 31], uses the term *length* to refer to $\delta(e)$, where the notion is different from ours. In [12], length refers to a global parameter that needs to be minimized. We consider length as a constraint parameter. Furthermore, our setting accounts for three types of costs (two for buy-at-bulk and one for lengths), which is richer than the earlier work on buy-at-bulk and hop-constrained network design (see Section 1.4.4) with two cost functions.

1.4.3 Other variants of buy-at-bulk network design

Besides the edge-weighted buy-at-bulk problem, there are other variants, including the node-weighted problem and the prize-collecting problem. In the prize-collecting problem, each terminal pair also has a penalty, and one can choose not to connect the pair and incur the penalty in the cost. Most results are on undirected graphs. For undirected node-weighted buy-at-bulk, there exists a polylogarithmic polynomial-time approximation algorithm [15] and a polylogarithmic quasi-polynomial-time competitive online algorithm [12]. In a more restricted case, i.e., undirected node-weighted Steiner trees, a polylogarithmic approximation algorithm was presented in [45] and a polylogarithmic competitive online algorithm was presented in [50]. Following [50], [41] extends to online undirected node-weighted Steiner forests while [42] extends to the online prize-collecting versions. These results fall under the unifying framework of [3] which utilizes an online primal-dual LP rounding scheme. The work [12] further extends to the online price-collecting buy-at-bulk problem with the same competitive ratio as the standard edge-weighted problem on both directed and undirected graphs. The work [38] considers a metric-based variant of undirected online buy-at-bulk and presents a framework that finds a cheap subgraph (compared to the minimum spanning tree or the optimal Steiner forest) that connects the terminal pairs with low stretch.

1.4.4 Undirected bi-criteria network design

A general class of undirected bi-criteria network problems was introduced by [48], where one basic setting is the undirected Steiner tree problem. The goal is to connect a subset of vertices to a specified root vertex. In the bi-criteria problem, the distance from the root to a

target vertex is required to be at most the given global threshold. [48] presented a bi-criteria algorithm for undirected Steiner trees that is $O(\log n)$-approximate and violates the distance constraints by a factor of $O(\log n)$. Following [48], [40] extends this result to a more general buy-at-bulk bi-criteria network design problem, where the objective is $\mathsf{polylog}(n)$-approximate and violates the distance constraints by a factor of $\mathsf{polylog}(n)$.

[29, 39] studied the tree embedding technique used for undirected network connectivity problems with *hop* constraints. For a positively weighted graph with a global parameter $h \in [n]$, the *hop distance* between vertices u and v is the minimum weight among the u-v-paths using at most h edges. Under some mild assumptions, the tree embedding technique allows a $\mathsf{polylog}(n)$-approximation by relaxing the hop distance within a $\mathsf{polylog}(n)$ factor for a rich class of undirected network connectivity problems.

1.4.5 Resource-constrained shortest path

The resource-constrained shortest path problem was introduced in [44]. The input consists of m resource types and a directed graph where each edge is associated with a non-negative cost and a non-negative (for all coordinates) resource consumption vector. The goal is to find a minimum-cost path that connects the single source vertex to the single sink vertex while satisfying the resource constraint. When $m = 1$, this problem is equivalent to the restricted shortest path problem [43, 47], which has been extensively used in the literature of spanners [9, 18, 21, 28, 34]. The results of [44] show that when m is a constant, there exists an FPTAS that finds a path with a cost at most the same as the feasible minimum-cost path by violating each budget by a factor of $1 + \varepsilon$. The FPTAS for $m = 2$ is used to solve the weighted spanner problem [32].

1.5 Organization

We present the $\tilde{O}(n^{4/5+\varepsilon})$-approximation algorithm for Buy-at-bulk Spanner on $[\mathsf{poly}(n)]_{\pm}$ in Section 2. All missing proofs are deferred to the full version paper [33].

2 An $\tilde{O}(n^{4/5+\varepsilon})$-approximation for Buy-at-Bulk Spanners

In this section, we give a short sketch of Theorem 3. While some of our other results (such as Theorem 12) are also of independent utility and some involve higher technical complexity, we choose to emphasize Theorem 3 here as it uses several tools developed throughout this work. Through this section, we get a glimpse into a number of our other results as well.

Throughout this subsection, we set $\mathcal{D} = [\mathsf{poly}(n)]_{\pm}$.

As in [5, 9, 28, 32] let τ be our guess of the optimal solution - OPT such that $\mathsf{OPT} \leq \tau \leq 2 \cdot \mathsf{OPT}$. Throughout this section, we will assume that our demand is uniform (i.e., $\textsc{Dem}(s,t) = 1 \; \forall \, (s,t) \in D$).

We define some common notations used in the literature for Spanners, Steiner forests, and buy-at-bulk network design. Let $\beta = n^{3/5}$ and $L_1 = \tau/n^{4/5}, L_2 = n^{4/5}\tau/k$ - we will use these parameters later in our algorithm. We say that any path $p(s,t)$ connecting a terminal pair (s,t) is *feasible* if $\sum_{e \in p(s,t)} \ell_e \leq \textsc{Dis}(s,t)$.

We say that a path $p(s,t)$ has cheap investment if $\sigma(p(s,t)) = \sum_{e \in p(s,t)} \sigma(e) \leq L_1$; we say that a path has cheap maintenance if $\delta(p(s,t)) = \sum_{e \in p(s,t)} \delta(e) \leq L_2$. Further, we say that $p(s,t)$ is *cheap* if it has both cheap investment and cheap maintenance [8].

[8] We only deal with uniform demand - thus we don't need to multiply $\delta(e)$ with demand for a single path.

We call a terminal pair $(s,t) \in D$ as *good* if the *local graph* $G^{s,t} = (V^{s,t}, E^{s,t})$ induced by the vertices on feasible $s \rightsquigarrow t$ paths that are cheap has at least n/β vertices; we say it is *bad* otherwise. Let D_a be the set of good pairs and D_b be the set of bad pairs; also let $k_a = |D_a|$ and $k_b = |D_b|$. Our definition of good and bad pairs here has to account for both negative lengths and the addition of the pay-per-use cost unlike previous literature [9, 28, 32]. Finally, we state that a set of paths $\{p(s,t)\}_{(s,t)\in D}$ resolves (or settles) a pair $(s,t) \in D$ if it contains a feasible $s \rightsquigarrow t$ path.

2.1 Resolving good pairs

We first define, $S = \{s \mid \exists t : (s,t) \in D\}$ and $T = \{t \mid \exists s : (s,t) \in D\}$. In this subsection, we settle the good pairs with high probability. We do this by sampling some vertices using Algorithm 1 and then adding some incoming paths and outgoing paths from the samples to the vertices in S and T respectively using Algorithm 2. We ensure that any path we build is both feasible and cheap.

Algorithm 1 Sample$(G(V,E))$.

1: $R \leftarrow \phi$, $k \leftarrow 3\beta \ln n$.
2: Sample k vertices independently and uniformly at random and store them in the set R.
3: **return** R.

▷ Claim 17. Algorithm 1 selects a set of samples R such that with high probability any given good pair (s,t) has at least one vertex from its local graph in R.

In Algorithm 2, we call Algorithm 1 to get a set of samples R. For each $u \in R, s \in S, t \in T$, we try to add a set of $s \rightsquigarrow u$ paths and a set of $u \rightsquigarrow t$ path each of cost both upfront cost at most $O(L_1)$ and pay-per-use cost at most L_2. It is possible we can't find some of these paths and if that happens, we just ignore this pair and continue.

Now, we just need a black box algorithm that can add a $s \rightsquigarrow u$ path that fits requirements. We present a slightly modified version of Theorem 16 in Corollary 18 that can exactly satisfy $m-1$ resource constraints where the corresponding resources are integers polynomial in n; and approximately satisfy one resource constraint that where the corresponding resource is a non-negative rational number.

▶ **Corollary 18.** *When for all edges $e \in E$, $w_{e,i} \in [\mathsf{poly}(n)]_{\pm}$ $\forall i \in \{1,2,\ldots,m-1\}$ and $w_{e,m} \in Q_{\geq 0}$, there exists a fully polynomial time $(1;1,1,\ldots,1+\zeta)-$ algorithm for the $k-$Resource-constrained Shortest Path problem that runs in time polynomial in input size and $1/\zeta$.*

Since we have to ensure that both upfront cost and pay-per-use cost are low enough, we need to model one of them as a constraint and the other as an objective. Without loss of generality, we set the upfront cost as the objective in Corollary 18 and set the pay-per-use cost as the 2^{nd} constraint (which is allowed to be rational and non-negative). The length is modeled as the first constraint (it is an integer polynomial in n). Without Theorem 16 and its extensions, we would not have a black box that meets these requirements.

Now, we do not know the exact length we need for a $s \rightsquigarrow u$ path. But, what we can do is search for all possible lengths in the interval:$[n \cdot \textsc{Min length}, n \cdot \textsc{Max length}]$[9] for the lowest possible length as in [32]. Using Claim 18 as our black box, we get a *cheap* and feasible path $p(s,t)$ if such a path exists.

[9] This can be speed up by binary search. We use linear search to improve readability.

Algorithm 2 Good pairs resolver $(G(V,E), \{\ell(e), \sigma(e), \delta(e)\}_{e\in E})$.

$R \leftarrow \phi$, $P' \leftarrow \phi$.
$R \leftarrow \text{Sample}(G(V,E))$.
for $u \in R$ **do**
  for $s \in S$ **do**
    for length $\in [n \cdot \text{Min length}, n \cdot \text{Max length}]$ **do**
      Use Claim 18 to find the cheapest $s \rightsquigarrow u$ path (in terms of upfront cost) that has pay-per-use cost $\leq L_2$ and length$(p) \leq len$. If a path is found and it has upfront cost $\leq L_1$, add it to P' and go to the next pair of terminals.
    If no path can be found for any length, continue to the next iteration.
for $u \in R$ **do**
  for $t \in T$ **do**
    for $len \in [n \cdot \text{Min length}, n \cdot \text{Max length}]$ **do**
      Use Claim 18 to find the cheapest $u \rightsquigarrow t$ path (in terms of upfront cost) that has pay-per-use cost $\leq L_2$ and length$(p) \leq len$. If a path is found and it has upfront cost $\leq L_1$, add it to P' and go to the next pair of terminals.
    If no path can be found for any length, continue to the next iteration with the next pair of terminals.
for $(s,t) \in D$ **do**
  Find some feasible $s \rightsquigarrow t$ path p_3 that is obtained by joining a $s \rightsquigarrow u$ path $p_1 \in P'$ and $u \rightsquigarrow t$ path $p_2 \in P'$. This path will have pay-per-use cost at most $2(1+\zeta)L_2$, upfront cost at most $2L_1$ (if no such path can be formed from P', then ignore this pair).
  Add p_3 to $\mathcal{P}_g$.
return $\mathcal{P}_g$

▶ **Lemma 19.** *With high probability, the set of paths P_g returned by Algorithm 2 resolves all good pairs in D with a total cost $\tilde{O}(n^{4/5} \cdot \tau)$. Moreover, Algorithm 2 runs in polynomial time.*

2.2 Resolving bad pairs

For this section, we adapt and expand a proof in [5]. We also add some detail for some parts of the proof. We also need an entirely different proof technique (based on [32]) for some parts of our proof. Before running the following algorithm, we first run the algorithm for good pairs and remove every resolved pair.

Let $\mathcal{P}^* = \{p^*(u,v)\}_{(u,v)\in D}$ be an optimal solution (note that since it is an optimal solution, all paths here are feasible). Let

- $D_b^1 = \{(u,v) \in D_b \mid \delta(p^*(u,v)) > L_2\}$,
- $D_b^2 = \{(u,v) \in D_b \setminus D_b^1 \mid \sigma(p^*(u,v)) > L_1\}$, and
- $D_b^3 = D_b \setminus (D_b^1 \cup D_b^2)$.

If $k_b \leq 4n^{6/5}$, then we can use Corollary 4 to resolve all the bad pairs with cost $\leq \tilde{O}(n^{3/5+\varepsilon})$ for some $\varepsilon > 0$. Note that other junction tree results from previous literature cannot be used here for they do not handle pay-per-use cost with distance constraints. We need Corollary 4 (and consequently all of its predecessors) to handle this. Thus, we can assume that $k_b > 4n^{6/5}$. Now, since each $(u,v) \in D_b^1$ has cost at least $L_2 = n^{4/5}\tau/k$, the cost of an optimal solution is τ, and $k < n^2$, we have, $|D_b^1| < 2k/n^{4/5} < 2n^{6/5}$. This also implies that if we ever have a situation where all pairs in D_b^2 and D_b^3 are resolved, then as $k_b > 4n^{6/5}$, $|D_b^1| < k_b/2$. Now, we have two cases based on whether $|D_b^2| > |D_b^3|$ or not.

We define the *density* of a set of paths P to be the ratio of the total cost of these paths to the number of pairs settled by those paths. Note that the total cost of a single path is the sum of the upfront cost and pay-per-use cost (since we have uniform demand). When we take sets of paths, we count every edge only once for upfront cost.

We first see how to efficiently construct a subset P_1 of paths with density $\tilde{O}(n^{4/5+\varepsilon})\tau/|D|$. Then we iteratively find paths with that density, remove the pairs corresponding to those paths, and repeat until we resolve all bad pairs. This gives a total cost of $\tilde{O}(n^{4/5+\varepsilon})\tau$. We construct P_1 by building two other sets P_2 and P_3 and picking the smaller density of them.

Since $|D_b^1| < k_b/2$, when $|D_b^2| > |D_b^3|$, we have $|D_b^2| > k_b/4$. When $|D_b^2| \leq |D_b^3|$, we have $|D_b^3| \geq k_b/4$.

2.2.1 When $|D_b^2| > |D_b^3|$

We use Minimum-density Distance-constrained Junction Tree as a black box for resolving this case. The following lemma is a slight variant of a standard result in previous literature [9, 28, 32].

▶ **Lemma 20.** *When $|D_b^2| > |D_b^3|$, we can get a set of paths P_2 that has density at most $\tilde{O}(n^{4/5+\varepsilon} \cdot \tau/k_b)$*

2.2.2 When $|D_b^3| \geq |D_b^2|$

To handle this case, we first build a linear program (LP) and solve it.

$$\text{min} \quad \sum_{e \in E} \sigma(e) \cdot x_e \tag{5a}$$

$$\text{subject to} \quad \sum_{(u,v) \in D_b} y_{uv} \geq k_b/4, \tag{5b}$$

$$\sum_{\Pi \ni p \ni e} f_p \leq x_e \qquad \forall (u,v) \in D_b, e \in E, \tag{5c}$$

$$\sum_{p \in \Pi(u,v)} f_p \geq y_{u,v} \qquad \forall (u,v) \in D_b, \tag{5d}$$

$$\sum_{p \in \Pi(u,v)} \delta(p) f_p \leq \frac{n^{4/5}\tau}{2k} \cdot y_{uv} \qquad \forall (u,v) \in D_b, \tag{5e}$$

$$0 \leq y_{u,v}, f_p, x_e \leq 1 \qquad \forall (u,v) \in D_b, p \in \Pi(u,v), e \in E. \tag{5f}$$

Intuitively, x_e denotes the edge indicators for $e \in E$, y_{uv} denotes the total flow value that connects u to v via cheap and feasible routes, and f_p denotes the flow value for path p. For every $(u,v) \in D_b$, let $\Pi(u,v)$ be the set of paths P from u to v in G such that $\sigma(P) \leq \tau/n^{4/5}$ and $\textsc{Len}(P) \leq \textsc{Dis}(u,v)$ (i.e., the distance constraints are satisfied). Also, let $\Pi = \cup_{(u,v) \in D_b} \Pi(u,v)$. LP (5) attempts to find a cheap and feasible (resource constraint-wise) solution where all the paths are taken from Π. We first use a separation-oracle-base approach to approximately solve LP (5) (described in more detail in Section 2.3), but the problem here is that this solution could have paths that are expensive in the pay-per-use cost δ. We resolve this issue by careful processing based on [5].

Because the optimal set of paths for the pairs in D_b^3 (i.e., $P_3^* = \{P_{uv}^* \mid (u,v) \in D_b^3\}$) corresponds to a basic feasible solution of LP (5), the objective value of $(\hat{x}, \hat{y}, \hat{f})$ is at most $(1+\zeta)\cdot\tau$. Now, let $(\check{x}, \check{y}, \check{f}) = (2\hat{x}, \hat{y}, 2\hat{f})$. Set $\check{f}_p = 0$ for every path p such that $\delta(p) > n^{4/5}\tau/k$. Then we reduce other $\check{f}_p$ values so that $\sum_{p \in \Pi(u,v)} \check{f}_p = y_{u,v}\ \forall (u,v) \in D_b$, and also prune the

x_e values such that $\check{x}_e = \max_{(u,v)\in D_b}\{\sum_{p:e\in P\in\Pi_{(u,v)}} \check{f}_p\}$ for all $e \in E$ (i.e., we prune x_e to make it as small as it can be while ensuring it can handle the necessary flow). This new $(\check{x}, \check{y}, \check{f})$ is another basic feasible solution to LP (5).

For now, we have $\sum_{p\in\Pi(u,v)} \check{f}_p = y_{u,v}\ \forall (u,v) \in D_b$; in addition, any path that still has $\check{f}_p > 0$ has both cheap investment and cheap maintenance (i.e., $\sigma(P) \leq \tau/n^{4/5}$ and $\delta(P) \leq n^{4/5}\tau/k$). Furthermore, these paths also satisfy the distance constraint of the demand pair they connect. All we have to do now is to round this solution.

2.2.2.1 Rounding our solution

Now we need to round the solution of LP (5) appropriately to decide which edges we need to include in our final solution. The overall structure of our rounding procedure is similar to that of [32]. Let $(\check{x}, \check{y})$ be a feasible approximate solution to LP (5) we get after processing $(\hat{x}, \hat{y})$. Let K_2 be the set of edges obtained by running Algorithm 3 on $\check{x}$.

The following lemma is an adaptation of Claim 2.3 from [9, 32].

Algorithm 3 Bad pair rounding [LP rounding] (x_e, D).

$E'' \leftarrow \phi$.
for $e \in E$ **do**
    Add e to G_{temp} with probability $\min\{n^{4/5}\ln n \cdot x_e, 1\}$;
for $(s,t) \in D$ **do**
    Find the cheapest path (in terms of pay-per-use cost) $p_{(s,t)}$ in G_{temp} that fulfills the distance constraints $\textsc{Dis}(s,t)$. Add the path to P_3 if it has pay-per-use cost $\leq L_2$.
return P_3, G_{temp}

▷ Claim 21. Let $A \subseteq E$. If Algorithm 3 receives a fractional vector $\{\check{x}_e\}$ with nonnegative entries satisfying $\sum_{e\in A} \check{x}_e \geq 1/10$, the probability that it outputs a set E'' disjoint from A is at most $\exp((-1/10)\cdot n^{4/5}\cdot \ln n)$.

Proof. If A contains an edge e with $\check{x}_e \geq 1/(n^{4/5}\ln n)$, then e is definitely included in E''.

Otherwise, the probability that no edge in A is included in E'' is

$$\prod_{e\in A}(1 - n^{4/5}\ln n\cdot \check{x}_e) \leq \exp\left(-\sum_{e\in A} n^{4/5}\ln n\cdot \check{x}_e\right) \leq \exp\left(-\frac{1}{10}n^{4/5}\ln n\right). \qquad \triangleleft$$

Let us now define anti-spanners which serve as a useful tool to analyze the rounding algorithm for our LP. As in [32], our definition of anti-spanners has to be more general than Definition 2.4 in [9] to account for the fact we also have multiple constraints for a single pair (and pay-per-use cost).

▶ **Definition 22.** *A set $A \subseteq E$ is an anti-spanner for a terminal pair $(s,t) \in E$ if $(V, E\setminus A)$ contains no feasible path from s to t of total cost at most L. If no proper subset of anti-spanner A for (s,t) is an anti-spanner for (s,t), then A is minimal. The set of all minimal anti-spanners for all bad edges is denoted by $\mathcal{A}$.*

The following lemma is an analogue of Claim 2.5 from [9]. The proof here is almost identical to that of [32].

▶ **Lemma 23.** *Let $\mathcal{A}$ be the set of all minimal anti-spanners for bad pairs. Then $|\mathcal{A}|$ is upper-bounded by $|D|\cdot 2^{(n/\beta)^2/2}$.*

Proof. Let $PS(s,t)$ be the power set of all edges in the local graph for a given bad pair (s,t). Since (s,t) is a bad pair we have at most n/β vertices and $(n/\beta)^2/2$ edges in the local graph, therefore $|PS(s,t)| \leq 2^{(n/\beta)^2/2}$ for any (s,t) that is a bad pair. Now, every anti-spanner for a specific demand pair $(s,t) \in D$ is a set of edges and therefore corresponds to an element in $PS(s,t)$. Let $PS_{\text{bad}} = \bigcup_{(s,t)} PS(s,t)$ where $(s,t) \in D$ are bad pairs. Every anti-spanner for a bad pair is a set of edges and therefore corresponds to an element in PS_{thin}. We have $|\mathcal{A}| \leq |PS_{\text{bad}}| \leq |D| \cdot 2^{(n/\beta)^2/2}$ which proves the lemma. ◀

The rest of this discussion is quite similar to [9] although the exact constants and the expressions involved are different because our bounds for $|\mathcal{A}|$ are weaker.

▶ **Lemma 24.** *With high probability set K_2 settles every bad pair (s,t) with $\breve{y}_{s,t} \geq 1/10$.*

Proof. For every bad pair $(s,t) \in D$ with $\breve{y}_{s,t} \geq 1/10$, if A is an anti-spanner for (s,t) then $\sum_{e \in A} \breve{x}_e \geq \sum_{P \in \Pi(s,t)} \breve{f}_p \geq 1/10$, where $\breve{f}_p$ is the value of the variable f_p in LP (5) that corresponds to the solution $(\breve{x}, \breve{y})$.

By Claim 21, the probability that A is disjoint from K_2 is at most $\exp(-n^{4/5} \cdot \ln n/10)$. Further using Lemma 23, we can bound the number of minimal anti-spanners for bad pairs and then if we apply union bound, we have the probability that K_2 is disjoint from any anti spanner for a bad pair is at most

$$\exp\left(-\frac{1}{10} n^{4/5} \cdot \ln n\right) \cdot |D| \cdot 2^{(n/\beta)^2/2}. \tag{6}$$

In the worst case, $|D|$ is n^2. Recall that $\beta = n^{3/5}$, so $(n/\beta)^2 = n^{4/5}$. Thus, (6) becomes

$$\exp\left(-\frac{1}{10} \cdot n^{4/5} \cdot \ln n + \ln\left(n^2 \cdot 2^{n^{4/5}/2}\right)\right) = \exp\left(-\Theta(n^{4/5} \ln n)\right).$$

Thus we have shown that the probability K_2 is disjoint from any anti-spanner for a bad pair is exponentially small when $\breve{y}_{s,t} \geq 1/10$. ◀

▶ **Lemma 25.** *When $0 \leq |C| < |D|/2$, with high probability, the density of K_2 is at most $\tilde{O}(n^{4/5+\varepsilon} \cdot \tau/|D|)$.*

Proof. Firstly, notice that the expected cost of K_2 would be at most $n^{4/5+\varepsilon} \ln n \cdot \tau$. We also point out that the number of pairs $(s,t) \in D$ for which $\breve{y}_{s,t} < 1/10$ is at most $5k_b/6$ because otherwise the amount of flow between all pairs is strictly less than $k_b/4$ which violates a constraint of LP (5). Since with high probability all pairs for which $\breve{y}_{s,t} \geq 1/10$ are satisfied, this means that the expected density of K_2 is at most

$$\frac{n^{4/5+\varepsilon} \ln n \cdot \tau}{k_b/6} = \frac{6n^{4/5+\varepsilon} \ln n \cdot \tau}{k_b} = \frac{\tilde{O}(n^{4/5+\varepsilon} \cdot \tau)}{k_b}.$$ ◀

▶ **Lemma 26.** *For any $\varepsilon > 0$, when $|D_b^2| \leq |D_b^3|$, with high probability, the density of P_3 is at most $\tilde{O}(n^{4/5} \cdot \tau/k_b)$.*

Proof of Theorem 3. With the help of Corollary 18, we can settle all good pairs with high probability with cost $\leq \tilde{O}(n^{4/5} \cdot \tau)$. For bad pairs, if we ever have $k_b \leq 4n^{6/5}$, we can use Corollary 4 to resolve them with cost $\leq \tilde{O}(n^{3/5+\varepsilon} \cdot \tau)$. Otherwise, we can make two sets of paths P_2 and P_3 using a junction tree and by rounding the modified solution to LP (5) respectively. By Lemmas 20 and 26, we can see that at least one of these sets of paths will have a density $\leq \tilde{O}(n^{4/5+\varepsilon} \cdot \tau/k_b)$. We take the cheaper among them and keep repeating the process until we can resolve all bad pairs with a high probability. This process has total cost $\leq \tilde{O}(n^{4/5+\varepsilon} \cdot \tau)$. ◀

2.3 Solving LP (5) via 2-RCSP

This section is based on [5, 32]. We describe how to solve LP (5). Note that LP (5) has an exponential number of variables. So, we take the dual (7) that has polynomially many variables and exponentially many constraints. If we have a valid separation oracle we can solve LP (7) using the ellipsoid method.

$$\max \quad (k_b/4)\cdot\theta - \sum_{(u,v)\in D_b} \zeta_{(u,v)} \tag{7a}$$

$$\text{subject to} \quad \sum_{(u,v)\in D_b} \alpha_{e,(u,v)} \le \sigma_e \qquad \forall e \in E, \tag{7b}$$

$$\beta_{(u,v)} + \zeta_{(u,v)} \ge \theta + \frac{n^{4/5}\tau}{2k}\cdot\gamma_{(u,v)} \qquad \forall (u,v)\in D_b, \tag{7c}$$

$$\beta_{(u,v)} \le \sum_{e\in p} \alpha_{e,(u,v)} + \delta(p)\cdot\gamma_{(u,v)} \qquad \forall (u,v)\in D_b, \forall p\in\Pi_{(u,v)}, \tag{7d}$$

$$\alpha_{e,(u,v)}, \beta_{(u,v)}, \gamma_{(u,v)}, \zeta_{(u,v)}, \theta \ge 0 \qquad \forall e\in E, \forall (u,v)\in D_b, \forall p\in\Pi_{(u,v)}. \tag{7e}$$

We only have polynomially many constraints in (7b), (7c). However, we have exponentially many constraints in (7d). [5] uses the restricted shortest path from [43] for this purpose. Our set of paths $\Pi_{(u,v)}$ also has distance constraints to account for, and we also need to handle negative lengths. We use Corollary 18 as our separation oracle. The following claim is similar to Claim 2.13 in [32] used for weighted spanners.

▷ Claim 27. For $(s,t)\in D$, 2-RCSP is a separation oracle for constraints in equation (7d).

By replacing the LHS of (7d) with $(1+\zeta)\beta_{(u,v)}$, we can obtain a $(1+\zeta)$-approximate solution for LP (5) for some constant $\zeta > 0$. Since our $\delta(e)$ values are rational and $\ell_e \in [\mathsf{poly}(n)]_\pm$ (which allows us to set a tolerance parameter greater than $1/\mathsf{poly}(n)$ to ensure that distance constraints are satisfied), we can use the FPTAS for 2-RCSP as a separation oracle to efficiently extract the violating constraints in (7d). We defer the details to the full version of our paper [33].

References

1 Amir Abboud and Greg Bodwin. Reachability preservers: New extremal bounds and approximation algorithms. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1865–1883. SIAM, 2018. doi:10.1137/1.9781611975031.122.

2 Reyan Ahmed, Greg Bodwin, Faryad Darabi Sahneh, Keaton Hamm, Mohammad Javad Latifi Jebelli, Stephen Kobourov, and Richard Spence. Graph spanners: A tutorial review. *Computer Science Review*, 37:100253, 2020. doi:10.1016/J.COSREV.2020.100253.

3 Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph Naor. The online set cover problem. *SIAM J. Comput.*, 39(2):361–370, 2009. doi:10.1137/060661946.

4 Matthew Andrews. Hardness of buy-at-bulk network design. In *45th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 115–124. IEEE, 2004. doi:10.1109/FOCS.2004.32.

5 Spyridon Antonakopoulos. *Buy-at-bulk-related problems in network design.* PhD thesis, Columbia University, 2009.

6 Baruch Awerbuch. Communication-time trade-offs in network synchronization. In *Proceedings of the Fourth Annual ACM Symposium on Principles of Distributed Computing*, PODC '85, pages 272–276, New York, NY, USA, 1985. Association for Computing Machinery. doi:10.1145/323596.323621.

7 Baruch Awerbuch and Yossi Azar. Buy-at-bulk network design. In *Proceedings 38th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 542–547. IEEE, 1997. doi:10.1109/SFCS.1997.646143.

8 Surender Baswana and Telikepalli Kavitha. Faster algorithms for all-pairs approximate shortest paths in undirected graphs. *SIAM J. Comput.*, 39(7):2865–2896, 2010. doi:10.1137/080737174.

9 Piotr Berman, Arnab Bhattacharyya, Konstantin Makarychev, Sofya Raskhodnikova, and Grigory Yaroslavtsev. Approximation algorithms for spanner problems and directed steiner forest. *Information and Computation*, 222:93–107, 2013. doi:10.1016/J.IC.2012.10.007.

10 Arnab Bhattacharyya, Elena Grigorescu, Kyomin Jung, Sofya Raskhodnikova, and David P Woodruff. Transitive-closure spanners. *SIAM Journal on Computing*, 41(6):1380–1425, 2012. doi:10.1137/110826655.

11 Greg Bodwin and Virginia Vassilevska Williams. Better distance preservers and additive spanners. In Robert Krauthgamer, editor, *SODA*, pages 855–872. SIAM, 2016. doi:10.1137/1.9781611974331.CH61.

12 Deeparnab Chakrabarty, Alina Ene, Ravishankar Krishnaswamy, and Debmalya Panigrahi. Online buy-at-bulk network design. *SIAM J. Comput.*, 47(4):1505–1528, 2018. doi:10.1137/16M1117317.

13 Moses Charikar, Chandra Chekuri, To-Yat Cheung, Zuo Dai, Ashish Goel, Sudipto Guha, and Ming Li. Approximation algorithms for directed steiner problems. *Journal of Algorithms*, 33(1):73–91, 1999. doi:10.1006/JAGM.1999.1042.

14 Moses Charikar and Adriana Karagiozova. On non-uniform multicommodity buy-at-bulk network design. In *Proceedings of the 37th Annual ACM Symposium on Theory of computing (STOC)*, pages 176–182, 2005. doi:10.1145/1060590.1060617.

15 C Chekuri, MT Hajiaghayi, G Kortsarz, and MR Salavatipour. Approximation algorithms for node-weighted buy-at-bulk network design. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1265–1274, 2007.

16 Chandra Chekuri, Guy Even, Anupam Gupta, and Danny Segev. Set connectivity problems in undirected graphs and the directed steiner network problem. *ACM Transactions on Algorithms (TALG)*, 7(2):1–17, 2011. doi:10.1145/1921659.1921664.

17 Chandra Chekuri, Mohammad Taghi Hajiaghayi, Guy Kortsarz, and Mohammad R Salavatipour. Approximation algorithms for nonuniform buy-at-bulk network design. *SIAM Journal on Computing*, 39(5):1772–1798, 2010. doi:10.1137/090750317.

18 Eden Chlamtáč, Michael Dinitz, Guy Kortsarz, and Bundit Laekhanukit. Approximating spanners and directed steiner forest: Upper and lower bounds. *ACM Transactions on Algorithms (TALG)*, 16(3):1–31, 2020. doi:10.1145/3381451.

19 Julia Chuzhoy, Anupam Gupta, Joseph Naor, and Amitabh Sinha. On the approximability of some network design problems. *ACM Transactions on Algorithms (TALG)*, 4(2):1–17, 2008. doi:10.1145/1361192.1361200.

20 Lenore Cowen and Christopher G. Wagner. Compact roundtrip routing in directed networks. *J. Algorithms*, 50(1):79–95, 2004. doi:10.1016/J.JALGOR.2003.08.001.

21 Michael Dinitz and Robert Krauthgamer. Directed spanners via flow-based linear programs. In *STOC*, pages 323–332, 2011. doi:10.1145/1993636.1993680.

22 Michael Dinitz and Zeyu Zhang. Approximating low-stretch spanners. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 821–840. SIAM, 2016. doi:10.1137/1.9781611974331.CH59.

23 Yevgeniy Dodis and Sanjeev Khanna. Design networks with bounded pairwise distance. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 750–759, 1999. doi:10.1145/301250.301447.

24 Dorit Dor, Shay Halperin, and Uri Zwick. All-pairs almost shortest paths. *SIAM J. Comput.*, 29(5):1740–1759, 2000. doi:10.1137/S0097539797327908.

25 Michael Elkin. Computing almost shortest paths. *ACM Trans. Algorithms*, 1(2):283–323, 2005. doi:10.1145/1103963.1103968.

26 Michael Elkin and David Peleg. The client-server 2-spanner problem with applications to network design. In Francesc Comellas, Josep Fàbrega, and Pierre Fraigniaud, editors, *SIROCCO 8, Proceedings of the 8th International Colloquium on Structural Information and Communication Complexity, Vall de Núria, Girona-Barcelona, Catalonia, Spain, 27-29 June, 2001*, volume 8 of *Proceedings in Informatics*, pages 117–132. Carleton Scientific, 2001.

27 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.

28 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.

29 Arnold Filtser. Hop-constrained metric embeddings and their applications. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 492–503. IEEE, 2022.

30 Lisa Fleischer, Jochen Könemann, Stefano Leonardi, and Guido Schäfer. Simple cost sharing schemes for multicommodity rent-or-buy and stochastic steiner tree. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 663–670, 2006. doi:10.1145/1132516.1132609.

31 Rohan Ghuge and Viswanath Nagarajan. Quasi-polynomial algorithms for submodular tree orienteering and directed network design problems. *Mathematics of Operations Research*, 47(2):1612–1630, 2022. doi:10.1287/MOOR.2021.1181.

32 Elena Grigorescu, Nithish Kumar, and Young-San Lin. Approximation algorithms for directed weighted spanners. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, 2023.

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

34 Elena Grigorescu, Young-San Lin, and Kent Quanrud. Online directed spanners and steiner forests. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021)*, pages 5:1–5:25. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.APPROX/RANDOM.2021.5.

35 Sudipto Guha, Adam Meyerson, and Kamesh Munagala. A constant factor approximation for the single sink edge installation problem. *SIAM Journal on Computing*, 38(6):2426–2442, 2009. doi:10.1137/050643635.

36 Anupam Gupta, Amit Kumar, Martin Pál, and Tim Roughgarden. Approximation via cost-sharing: a simple approximation algorithm for the multicommodity rent-or-buy problem. In *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, pages 606–615. IEEE, 2003. doi:10.1109/SFCS.2003.1238233.

37 Anupam Gupta, Amit Kumar, and Tim Roughgarden. Simpler and better approximation algorithms for network design. In *Proceedings of the 35th Annual ACM Symposium on Theory of computing (STOC)*, pages 365–372, 2003. doi:10.1145/780542.780597.

38 Anupam Gupta, R Ravi, Kunal Talwar, and Seeun William Umboh. Last but not least: Online spanners for buy-at-bulk. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 589–599. SIAM, 2017. doi:10.1137/1.9781611974782.38.

39 Bernhard Haeupler, D Ellis Hershkowitz, and Goran Zuzic. Tree embeddings for hop-constrained network design. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 356–369, 2021. doi:10.1145/3406325.3451053.

40 Mohammad Taghi Hajiaghayi, Guy Kortsarz, and Mohammad R Salavatipour. Approximating buy-at-bulk and shallow-light k-steiner trees. *Algorithmica*, 53:89–103, 2009. doi:10.1007/S00453-007-9013-X.

41 Mohammad Taghi Hajiaghayi, Vahid Liaghat, and Debmalya Panigrahi. Online node-weighted steiner forest and extensions via disk paintings. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 558–567. IEEE, 2013. doi:10.1109/FOCS.2013.66.

42 MohammadTaghi Hajiaghayi, Vahid Liaghat, and Debmalya Panigrahi. Near-optimal online algorithms for prize-collecting steiner problems. In *Automata, Languages, and Programming: 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I 41*, pages 576–587. Springer, 2014. doi:10.1007/978-3-662-43948-7_48.

43 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.

44 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.

45 P Klein and R Ravi. A nearly best-possible approximation algorithm for node-weighted steiner trees. *Journal of Algorithms*, 1(19):104–115, 1995.

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

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

48 Madhav V Marathe, Ravi Sundaram, SS Ravi, Daniel J Rosenkrantz, and Harry B Hunt III. Bicriteria network design problems. *Journal of algorithms*, 28(1):142–171, 1998. doi:10.1006/JAGM.1998.0930.

49 Adam Meyerson, Kamesh Munagala, and Serge Plotkin. Cost-distance: Two metric network design. *SIAM Journal on Computing*, 38(4):1648–1659, 2008. doi:10.1137/050629665.

50 Joseph Naor, Debmalya Panigrahi, and Mohit Singh. Online node-weighted steiner tree and related problems. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 210–219. IEEE, 2011. doi:10.1109/FOCS.2011.65.

51 Jakub Pachocki, Liam Roditty, Aaron Sidford, Roei Tov, and Virginia Vassilevska Williams. Approximating cycles in directed graphs: Fast algorithms for girth and roundtrip spanners. In Artur Czumaj, editor, *SODA*, pages 1374–1392. SIAM, 2018. doi:10.1137/1.9781611975031.91.

52 David Peleg and Alejandro A. Schäffer. Graph spanners. *Journal of Graph Theory*, 13(1):99–116, 1989. doi:10.1002/jgt.3190130114.

53 David Peleg and Jeffrey D. Ullman. An optimal synchronizer for the hypercube. *SIAM J. Comput.*, 18(4):740–747, 1989. doi:10.1137/0218050.

54 Liam Roditty, Mikkel Thorup, and Uri Zwick. Roundtrip spanners and roundtrip routing in directed graphs. *ACM Trans. Algorithms*, 4(3):29:1–29:17, 2008. doi:10.1145/1367064.1367069.

55 F Sibel Salman, Joseph Cheriyan, Ramamoorthi Ravi, and Sairam Subramanian. Approximating the single-sink link-installation problem in network design. *SIAM Journal on Optimization*, 11(3):595–610, 2001. doi:10.1137/S1052623497321432.

56 Xiangkun Shen and Viswanath Nagarajan. Online covering with l_q-norm objectives and applications to network design. *Mathematical Programming*, 184, 2020. doi:10.1007/S10107-019-01409-9.

57 Kunal Talwar. The single-sink buy-at-bulk lp has constant integrality gap. In *International Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 475–486. Springer, 2002. doi:10.1007/3-540-47867-1_33.