Characterizing Demand Graphs for (Fixed-Parameter) Shallow-Light Steiner Network

Amy Babay
Johns Hopkins University, Baltimore, Maryland, USA
babay@cs.jhu.edu

Michael Dinitz
Johns Hopkins University, Baltimore, Maryland, USA
mdinitz@cs.jhu.edu

Zeyu Zhang
Johns Hopkins University, Baltimore, Maryland, USA
zyzhang92@gmail.com

Abstract

We consider the Shallow-Light Steiner Network problem from a fixed-parameter perspective. Given a graph G, a distance bound L, and p pairs of vertices $(s_1, t_1), \ldots, (s_p, t_p)$, the objective is to find a minimum-cost subgraph G' such that s_i and t_i have distance at most L in G' (for every $i \in [p]$). Our main result is on the fixed-parameter tractability of this problem for parameter p. We exactly characterize the demand structures that make the problem "easy", and give FPT algorithms for those cases. In all other cases, we show that the problem is W[1]-hard. We also extend our results to handle general edge lengths and costs, precisely characterizing which demands allow for good FPT approximation algorithms and which demands remain W[1]-hard even to approximate.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability

Keywords and phrases fixed-parameter tractable, network design, shallow-light steiner network, demand graphs

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2018.33

Related Version A full version of the paper is available at [2], `https://arxiv.org/abs/1802.10566`.

Funding Supported in part by NSF award 1535887.

1 Introduction

In many network design problems we are given a graph $G = (V, E)$ and some demand pairs $(s_1, t_1), (s_2, t_2), \ldots, (s_p, t_p) \subseteq V \times V$, and are asked to find the "best" (usually minimum-cost) subgraph in which every demand pair satisfies some type of connectivity requirement. In the simplest case, if the demands are all pairs and the connectivity requirement is just to be connected, then this is the classical Minimum Spanning Tree problem. If we consider other classes of demands, then we get more difficult but still classical problems. Most notably, if the demands form a star (or any connected graph on V), then we have the famous Steiner Tree problem. If the demands are completely arbitrary, then we have the Steiner Forest problem. Both problems are known to be in FPT parameterized by the number of demands [9] (i.e., they can be solved in $f(p) \cdot poly(n)$ time for some function f).

38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2018).
Editors: Sumit Ganguly and Paritosh Pandya; Article No. 33; pp. 33:1–33:22
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

There are many obvious generalizations of Steiner Tree and Steiner Forest of the general network design flavor given above. We will be particularly concerned with *length-bounded* variants, which are related to (but still quite different from) *directed* variants. In Directed Steiner Tree (DST) the input graph is directed and the demands are a directed star (either into or out of the root), while in Directed Steiner Network (DSN) the input graph and demands are both directed, but the demands are an arbitrary subset of $V \times V$. Both problems have been well-studied (e.g., [5, 19, 6, 8, 1]), and in particular it is known that the same basic dynamic programming algorithm used for Steiner Tree will also give an FPT algorithm for DST. However, DSN is known to be W[1]-hard, so it is not believed to be in FPT [11].

In the length-bounded setting, we typically assume that the input graph and demands are undirected but each demand has a distance bound, and a solution is only feasible if every demand is connected within distance at most the given bound (rather than just being connected). One of the most basic problems of this form is the Shallow-Light Steiner Tree problem (SLST), where the demands form a star with root $r = s_1 = s_2 = \cdots = s_p$ and there is a global length bound L (so in any feasible solution the distance from r to t_i is at most L for all $i \in [p]$). As with DST and DSN, SLST has been studied extensively [16, 18, 14, 13]. If we generalize this problem to arbitrary demands, we get the Shallow-Light Steiner Network problem, which is the main problem we study in this paper. Surprisingly, it has not received nearly the same amount of study (to the best of our knowledge, this paper is the first to consider it explicitly). It is formally defined as follows (note that we focus on the special case of unit lengths, and will consider general lengths in Section 5:

▶ **Definition 1** (Shallow-Light Steiner Network). Given a graph $G = (V, E)$, a cost function $c : E \to \mathbb{R}^+$, a length function $l : E \to \mathbb{R}^+$, a distance bound L, and p pairs of vertices $\{s_1, t_1\}, \ldots, \{s_p, t_p\}$. The objective of SLSN is to find a minimum cost subgraph $G' = (V, S)$, such that for every $i \in [p]$, there is a path between s_i and t_i in G' with length less or equal to L.

Let H be the graph with $\{s_1, \ldots, s_p, t_1, \ldots, t_p\}$ as its vertex set and $\{\{s_1, t_1\}, \ldots, \{s_p, t_p\}\}$ as its edge set. We call H the *demand graph* of the problem. We use $|H|$ to represent the number of edges in H.

Both the directed and the length-bounded settings share a dichotomy between considering either star demands (DST/SLST) or totally general demands (DSN/SLSN). But this gives an obvious set of questions: what demand graphs make the problem "easy" (in FPT) and what demand graphs make the problem "hard" (W[1]-hard)? Recently, Feldmann and Marx [11] gave a complete characterization for this for DSN. Informally, they proved that if the demand graph is transitively equivalent to an "almost-caterpillar" (the union of a constant number of stars where their centers form a path, as well as a constant number of extra edges), then the problem is in FPT, and otherwise the problem is W[1]-hard.

While *a priori* there might not seem to be much of a relationship between the directed and the length-bounded problems, there are multiple folklore results that relate them, usually by means of some sort of layered graph. For example, any FPT algorithm for the DST problem can be turned into an FPT algorithm for SLST (with unit edge lengths) and vice versa through such a reduction (though this is a known result, to the best of our knowledge it has not been written down before, so we include it for completeness in Section 3.2). Such a relationship is not known for more general demands, though.

In light of these relationships between the directed and the length-bounded settings and the recent results of [11], it is natural to attempt to characterize the demand graphs that make SLSN easy or hard. We solve this problem, giving (as in [11]) a complete characterization of easy and hard demand graphs. Our formal results are given in Section 2, but informally we show that SLSN is significantly harder than DSN: the only "easy" demand graphs are stars (in which case the problem is just SLST) and constant-size graphs. Even tiny modifications, like a star with a single independent edge, become W[1]-hard (despite being in FPT for DSN).

1.1 Connection to Overlay Routing

SLSN is particularly interesting due to its connection to overlay routing protocols that use *dissemination graphs* to support next-generation Internet services. In fact, our motivation for studying the fixed-parameter complexity of SLSN is from our use of heuristics for SLSN in a recent system [3] in which the number of demands was relatively small.

Many emerging applications (such as remote surgery) require extremely low-latency yet highly reliable communication, which the Internet does not natively support. Babay et al. [3] recently showed that such applications can be supported by using overlay networks to enable routing schemes based on *subgraphs* (dissemination graphs) rather than paths. Their extensive analysis of real-world data shows that two node-disjoint overlay paths effectively overcome any one fault in the middle of the network, but specialized dissemination graphs are needed to address problems at a flow's source or destination. Because problems affecting a source typically involve probabilistic loss on that source's outgoing links, a natural approach to increase the probability of a packet being successfully transmitted is to increase the number of outgoing links on which it is sent. In [3], when a problem is detected at a particular flow's source, that source switches to use a dissemination graph that floods its packets to all of its overlay neighbors and then forwards them from these neighbors to the destination. The paths from the source's neighbors to the destination must meet the application's strict latency requirement, but since the bandwidth used on every edge a packet traverses must be paid for, the total number of edges used should be minimized. Thus, constructing the optimal dissemination graph in this setting is precisely the Shallow-Light Steiner Tree problem, where the root of the demands is the destination and the other endpoints are the neighbors of the source.

However, in order to achieve the desired reliability, it was shown in [3] that *simultaneous* failures at both the source *and* the destination of a flow must also be addressed. Since it is not known in advance which neighbors of the source or destination will be reachable during a failure, the most resilient approach is to require a latency-bounded path from *every* neighbor of the source to *every* neighbor of the destination. This is precisely SLSN with a complete bipartite demand graph. Since no FPT algorithm for SLSN with complete bipartite demands was known, [3] relied on a heuristic that worked well in practice. The search for an FPT algorithm for SLSN with complete bipartite demands was the main motivation for this work.

In the context of dissemination-graph-construction problems, our results provide a good solution for problems affecting either a source or a destination: the FPT algorithm for the SLST problem is quite practical, since overlay topologies typically have bounded degree (and thus a bounded total number of demands). Unfortunately a trivial corollary of our main result implies that the other case which was particularly important in this setting, SLSN with complete bipartite demands, is W[1]-hard. This has important applications to future system design, since (like all hardness results) it will allow system designers to focus on issues other than perfect algorithms, even for dissemination graphs that provide only slightly more resiliency than SLST.

2 Our Results and Techniques

In order to distinguish the easy from the hard cases of the SLSN problem with respect to the demand graph, we should first define the problem with respect to a class (set) of demand graphs.

▶ **Definition 2.** Given a class $\mathcal{C}$ of graphs. The problem of Shallow-Light Steiner Network with restricted demand graph class $\mathcal{C}$ ($\text{SLSN}_{\mathcal{C}}$) is the SLSN problem with the additional restriction that the demand graph H of the problem must be isomorphic to some graph in $\mathcal{C}$.

We define $\mathcal{C}_\lambda$ as the class of all demand graphs with at most λ edges, and $\mathcal{C}^*$ as the class of all star demand graphs (there is a central vertex called the root, and every other vertex in the demand graph is adjacent to the root and only the root). Our main result is that these are *precisely* the easy classes: We first prove that SLSN is in XP parameterized by the number of demands (i.e. solvable in $n^{f(p)}$ time for some function f), which immediately implies that $\text{SLSN}_{\mathcal{C}_\lambda}$ can be solved in polynomial time if λ is a constant. Note that $\text{SLSN}_{\mathcal{C}^*}$ is precisely the SLST problem, for which a folklore FPT algorithm exists, thus $\text{SLSN}_{\mathcal{C}^*}$ (while NP-hard) is in FPT for parameter p. We also show that, for any other class $\mathcal{C}$ (i.e., any class which is not just a subset of $\mathcal{C}^* \cup \mathcal{C}_\lambda$ for some constant λ), the problem $\text{SLSN}_{\mathcal{C}}$ is W[1]-hard for parameter p. In other words, if the class of demand graphs includes arbitrarily large non-stars, then the problem is W[1]-hard parameterized by the number of demands.

More formally, we prove the following theorems.

▶ **Theorem 3.** *The unit-length arbitrary-cost* SLSN *problem with parameter p is in* XP*, and it can be solved in $n^{O(p^4)}$ time.*

By "unit-length arbitrary-cost" we mean that the length $l(e) = 1$ for all edges $e \in E$, while the cost c is arbitrary. To prove this theorem, we first prove a structural lemma which shows that the optimal solution must be the union of several lowest cost paths with restricted length (these paths may be between steiner nodes, but we show that there cannot be too many). Then we just need to guess all the endpoints of these paths, as well as all the lengths of these paths. We prove that there are only $n^{O(p^4)}$ possibilities, and the running time is also $n^{O(p^4)}$. The algorithm and proof is in Section 3.1.

▶ **Theorem 4.** *The unit-length arbitrary-cost* $\text{SLSN}_{\mathcal{C}^*}$ *problem is* FPT *for parameter p.*

As mentioned, $\text{SLSN}_{\mathcal{C}^*}$ is exactly the same as SLST, so we use a folklore reduction between SLST and DST in Section 3.2 to prove this theorem.

▶ **Theorem 5.** *If $\mathcal{C}$ is a recursively enumerable class, and $\mathcal{C} \nsubseteq \mathcal{C}_\lambda \cup \mathcal{C}^*$ for any constant λ, then* $\text{SLSN}_{\mathcal{C}}$ *is* W[1]*-hard for parameter p, even in the unit-length and unit-cost case.*

Many W[1]-hardness results for network design problems reduce from the Multi-Colored Clique (MCC) problem, and ours are no exception. We reduce from MCC to $\text{SLSN}_{\mathcal{C}'}$, where $\mathcal{C}'$ is a specific subset of $\mathcal{C}$ which has some particularly useful properties, and which we show must exist for any such $\mathcal{C}$. Since $\mathcal{C}' \subseteq \mathcal{C}$, this will imply the theorem. The reduction is in Section 4.2.

All of the above results are in the unit-length setting. We extend both our upper bounds and hardness results to handle arbitrary lengths, but with some extra complications. If $p = 1$ (there is only one demand), then with arbitrary lengths and arbitrary costs the SLSN problem is equivalent to the Restricted Shortest Path problem, which is known to be

NP-hard [15]. Therefore we can no longer hope for a polynomial time exact solution when $p = 1$, and thus cannot hope for an FPT algorithm (with parameter p). So we change our notion of "easy" from "solvable in FPT" to "arbitrarily approximable in FPT": we show $(1+\varepsilon)$-approximation algorithms for the easy cases, and prove that there is no $\left(\frac{5}{4}-\varepsilon\right)$-approximation algorithm for the hard cases in $f(p) \cdot poly(n)$ time for any function f. We discuss these results in Section 5.

▶ **Theorem 6.** *For any constant $\lambda > 0$, there is a fully polynomial time approximation scheme (*FPTAS*) for the arbitrary-length arbitrary-cost* $\mathrm{SLSN}_{\mathcal{C}_\lambda}$ *problem.*

▶ **Theorem 7.** *There is a $(1+\epsilon)$-approximation algorithm in $O(4^p \cdot poly(\frac{n}{\varepsilon}))$ time for the arbitrary-length arbitrary-cost* $\mathrm{SLSN}_{\mathcal{C}^*}$ *problem.*

For both upper bounds, we use basically the same algorithm as in the unit-length arbitrary-cost case, with some changes inspired by the $(1+\varepsilon)$-approximation algorithm for the Restricted Shortest Path problem [17].

Our next theorem is analogous to Theorem 5, but since costs are allowed to be arbitrary we can prove stronger hardness of approximation (under stronger assumptions).

▶ **Theorem 8.** *Assume that (randomized) Gap-Exponential Time Hypothesis (Gap-ETH, see [4]) holds. Let $\varepsilon > 0$ be a small constant, and $\mathcal{C}$ be a recursively enumerable class where $\mathcal{C} \nsubseteq \mathcal{C}_\lambda \cup \mathcal{C}^*$ for any constant λ. Then, there is no $\left(\frac{5}{4}-\varepsilon\right)$-approximation algorithm in $f(p) \cdot n^{O(1)}$ time for* $\mathrm{SLSN}_{\mathcal{C}}$ *for any function f, even in the unit-length and polynomial-cost case.*

Note that this theorem uses a much stronger assumption (Gap-ETH rather than W[1] $\neq$ FPT), which assumes that there is no (possibly randomized) algorithm running in $2^{o(n)}$ time that can distinguish whether a 3SAT formula is satisfiable or at most a $(1-\varepsilon)$-fraction of its clauses can be satisfied. This enables us to utilize the hardness result for a generalized version of the MCC problem from [7], which will allow us to modify our reduction from Theorem 5 to get hardness of approximation.

2.1 Relationship to [11]

As mentioned, our results and techniques are strongly motivated and influenced by the work of Feldmann and Marx [11], who proved similar results in the directed setting. Informally, they showed that Directed Steiner Network is in FPT if the demand graph is transitively equivalent to an "almost-caterpillar", and otherwise it is W[1]-hard. Since "transitively equivalent to an almost-caterpillar" is a complex and subtle class, this showed that the tractability of DSN exhibits interesting behavior. Our results, on the other hand, show that SLSN is extraordinarily hard: there simply are not any algorithms possible for demand graphs that are even a little bit complex, despite the folklore relationships between directed settings and length-bounded settings. Thus our hardness proof is significantly more complicated than the reduction in [11], despite sharing some ideas.

The main case of the hardness reduction of [11] (which, like our reduction, is from MCC) is when the demand graph is a 2-by-k complete bipartite graph (i.e., two stars with the same leaf set). For this case, their reduction from MCC uses one star to control the choice of edges in the clique and another star to control the choice of vertices in the clique. They set this up so that if there is a clique of the right size then the "edge demands" and the "vertex demands" can be satisfied with low cost by making choices corresponding to the clique, while if no such clique exists then any way of satisfying the two types of demands simultaneously must have larger cost.

The 2-by-k complete bipartite graph is also a hard demand graph in our setting, and the same reduction from [11] can be straightforwardly modified to prove this (this appears as one of our cases). However, we prove that far simpler demand graphs are also hard. Most notably, the “main” case of our proof is when the demand graph is a single star together with one extra edge. Since we have only a single star in our demand graph, we cannot have two “types” of demands (vertex demands and edge demands) in our reduction. So we instead use the star to correspond to “edge demands” and use the single extra edge to simultaneously simulate all of the “vertex demands”. This makes our reduction significantly more complicated.

With respect to upper bounds, the algorithm of [11] is quite complex in part due to the complexity of the demand graphs that it must solve. Our hardness results for SLSN imply that we need only concern ourselves with demand graphs that are star or have constant size. The star setting is relatively simple due to a reduction to DST, but it is not obvious how to use any adaptation of [11] (or the earlier [10]) to handle a constant number of demands for SLSN. Our algorithm ends up being relatively simple, but requires a structural lemma which was not necessary in the DSN setting.

3 Algorithms for Unit-Length Arbitrary-Cost SLSN

In this section we discuss the “easy” cases of SLSN. We first present an XP algorithm for SLSN in Section 3.1. In Section 3.2, we describe a reduction from SLSN with star demand graphs to DST, which gives an FPT algorithm.

3.1 The XP algorithm

The XP algorithm for Theorem 3 relies on the following structural lemma, which allows us to limit the structure of the optimal solution and finally find it out. Note that this lemma works not only for the unit-length case, but also for the arbitrary-length case.

▶ **Lemma 9.** *In any feasible solution $S \subseteq E$ of the* SLSN *problem, there exists a way to assign a path P_i between s_i and t_i in S for each demand $\{s_i, t_i\} \in H$ such that:*

- *For each $i \in [p]$, the total length of P_i is at most L and there is no cycle in P_i.*
- *For each $i, j \in [p]$ and $u, v \in P_i \cap P_j$, the paths between u and v in P_i and P_j are the same.*

Proof. We give a constructive proof. Let $m = |S|$ and $S = \{e_1, \ldots, e_m\}$. We first want to modify the lengths to ensure that there is always a unique shortest path. Let Δ denote the minimum length difference between any two subsets of S with different total length, i.e.,

$$\Delta = \min_{A,B \subseteq S, \sum_{e\in A} l(e) \neq \sum_{e \in B} l(e)} \left| \sum_{e\in A} l(e) - \sum_{e \in B} l(e) \right|.$$

We create a new length function g where $g(e_i) = l(e_i) + \Delta \cdot 2^{-i}$. Note that Δ is always non-zero for any S which has at least 2 edges, and the problem is trivial when $|S| = 1$.

We now show that any two paths have different lengths under g. Consider any two different paths P_x and P_y. If $\sum_{e \in P_x} l(e) \neq \sum_{e \in P_y} l(e)$, then without loss of generality we assume $\sum_{e \in P_x} l(e) < \sum_{e \in P_y} l(e)$. Then

$$\sum_{e \in P_x} g(e) \leq \sum_{e \in P_x} l(e) + \sum_{i=1}^{m} \Delta \cdot 2^{-i} < \sum_{e \in P_x} l(e) + \Delta \leq \sum_{e \in P_y} l(e) < \sum_{e \in P_y} g(e). \tag{1}$$

Algorithm 1 Unit-length arbitrary-cost SLSN.

```
Let M ← ∑_{e∈E} c(e) and S ← E
for Q ⊆ V where |Q| ≤ p(p − 1) do
    Q′ ← Q ∪ (⋃_{i=1}^{p} {s_i, t_i})
    for E′ ⊆ {{u, v} | u, v ∈ Q′, u ≠ v} and l′ : E′ → [L] do
        T ← ∅
        for {u, v} ∈ E′ do
            T ← T ∪ {the lowest cost path between u and v with length at most l′({u, v})}
             // if such path does not exist, T remains the same
        end for
        if T is a feasible solution and ∑_{e∈T} l′(e) < M then
            M ← ∑_{e∈T} c(e) and S ← T
        end if
    end for
end for
return S
```

Otherwise, if $\sum_{e\in P_x} l(e) = \sum_{e\in P_y} l(e)$, then

$$\sum_{e\in P_x} g(e) - \sum_{e\in P_y} g(e) = \sum_{i:e_i\in P_x} \Delta\cdot 2^{-i} - \sum_{i:e_i\in P_y} \Delta\cdot 2^{-i} \neq 0.$$

Therefore in both cases P_x and P_y have different lengths under g.

For each demand $\{s_i, t_i\} \in H$, we let P_i be the shortest path between s_i and t_i in S under the new length function g. Because any two paths under g have different length, the shortest path between each $\{s_i, t_i\} \in H$ is unique. In addition, because these are shortest paths and edge lengths are positive, they do not contain any cycles.

For each $i \in [p]$, we can see that P_i is also one of the shortest paths between s_i and t_i under original length function l. This is because in equation (1) we proved that a shorter path under length function l is still a shorter path under length function g. Since S is a feasible solution, the shortest path between s_i and t_i in S must have length at most L. Thus for each $i \in [p]$, we have $\sum_{e\in P_i} l(e) \le L$.

For any two different paths P_i and P_j, let $u, v \in P_i \cap P_j$. If the subpath of P_i between u and v is different from the subpath of P_j between u and v, then by the uniqueness of shortest paths under g we know that either P_i or P_j is not a shortest path (since one of them could be improved by changing the subpath between u and v). This contradicts our definition of P_i and P_j, and hence they must use the same subpath between u and v. ◀

Lemma 9 implies that any two paths P_i, P_j in the optimal solution are either disjoint, or share exactly one (maximal) subpath. Since there are only p demands, the total number of shared subpaths is at most $\binom{p}{2}$. Therefore we can solve the unit-length arbitrary-cost $\text{SLSN}_{\mathcal{C}_\lambda}$ by guessing these subpaths.

Informally, we guess the set of endpoints of all the "maximal overlapping subpaths" (Q), guess how these endpoints are paired up to create the distinct subpaths (E'), guess the length of each subpath, and then find the lowest cost path that connects the endpoints of each guessed subpath and is within the guessed length. The full algorithm is given as Algorithm 1.

▶ **Claim 10.** *The running time of Algorithm 1 is* $n^{O(p^4)}$.

Proof. Clearly there are at most $n^{p(p-1)}$ possibilities for Q, and for each Q there are at most $2^{(p(p-1)+2p)^2}$ possible sets E' and at most $L^{(p(p-1)+2p)^2}$ possible l'. Since we assume unit edge lengths, we can use the Bellman-Ford algorithm to find the lowest cost path within a given length bound in polynomial time. Checking feasibility also takes polynomial time using standard shortest path algorithms. Thus, the running time is at most $n^{p(p-1)} \cdot 2^{(p(p+1))^2} \cdot n^{(p(p+1))^2} \cdot poly(n)$. ◀

Proof of Theorem 3

By Claim 10, the running time of Algorithm 1 is $n^{O(p^4)}$. Now we will prove correctness. The algorithm always returns a feasible solution, because we replace S by T only if T is feasible, and thus S is always a feasible solution. Therefore, we only need to show that this algorithm returns a solution with cost at most the cost of the optimal solution.

Let the optimal solution be S^*. We assign P_i^* for all $i \in [p]$ as in Lemma 9. Recall that path P_i^* and P_j^* can share at most one (maximal) subpath for each $i, j \in [p]$ where $i \neq j$. Let Q^* be the endpoint set of the (maximal) subpaths which are shared by some P_i^* and P_j^*, and let $Q'^* = Q^* \cup \bigcup_{i=1}^{p} \{s_i, t_i\}$.

We can see that the optimal solution S^* can be partitioned to a collection of paths by Q^*. We use E'^* to represent whether two vertices in Q'^* are "adjacent" on some path P_i^*: for any $u, v \in Q'^*$ where $u \neq v$, the set E'^* contains $\{u, v\}$ if and only if there exists $i \in [p]$ such that $u, v \in P_i^*$, and there is no vertex $w \in Q'^* \setminus \{u, v\}$ which is in the subpath between u and v in P_i^*. For each $\{u, v\} \in E'^*$, let $P_{\{u,v\}}^*$ be the subpath between u and v on path P_i^*. This is well defined because by Lemma 9 the subpath is unique. We define $l'^*(\{u, v\})$ as the length of $P_{\{u,v\}}^*$ for each $\{u, v\} \in E'^*$

Note that for any $\{u, v\} \neq \{u', v'\} \in E'^*$, we also know that $P_{\{u,v\}}^*$ and $P_{\{u',v'\}}^*$ are edge-disjoint. To see this, assume that they do share an edge, and let u'' and v'' be the endpoints of the (maximal) shared subpath between $P_{\{u,v\}}^*$ and $P_{\{u',v'\}}^*$. Then u'' and v'' are both in Q'^*, and at least one of them is in $Q'^* \setminus \{u, v\}$ or in $Q'^* \setminus \{u', v'\}$, which contradicts our definition of E'^*.

Since the algorithm iterates over all possibilities for Q, E' and l', there is some iteration in which $Q = Q'^*$, $E' = E'^*$, and $l' \equiv l'^*$. We will show that the algorithm also must find an optimal feasible solution in this iteration.

For each $i \in [p]$, the path P_i^* is partitioned to edge-disjoint subpaths by Q'^*. Let q_i be the number of subpaths, and let the endpoints be $s_i = v_{i,0}, v_{i,1}, \ldots, v_{i,q_i-1}, v_{i,q_i} = t_i$. We further let these subpaths be $P_{\{s_i,v_{i,1}\}}^*, P_{\{v_{i,1},v_{i,2}\}}^*, \ldots, P_{\{v_{i,q_i-1},t_i\}}^*$. By the definition of l'^*, for each $j \in [q_i]$, there must be a path between $v_{i,j-1}$ and $v_{i,j}$ with length at most $l'^*(\{v_{i,j-1}, v_{i,j}\})$ in graph G. Thus after the algorithm visited $\{v_{i,j-1}, v_{i,j}\} \in E'^*$, the edge set T must contains a path between u and v with length at most $l'^*(\{v_{i,j-1}, v_{i,j}\})$. Therefore we know that the edge set T in this iteration contains a path between s_i and t_i with length $\sum_{j=1}^{q_i} l'^*(\{v_{i,j-1}, v_{i,j}\}) \leq L$, and thus it is a feasible solution.

Let $MinCost(u, v, d)$ be the lowest cost for a path between u and v with distance at most d in graph G, then the total cost of this solution is $\sum_{\{u,v\} \in E'^*} MinCost(u, v, l'^*(\{u, v\}))$. Moreover, for each $\{u, v\} \in E'^*$ and $\{u', v'\} \in E'^*$ with $\{u, v\} \neq \{u', v'\}$, the paths $P_{\{u,v\}}^*$ and $P_{\{u',v'\}}^*$ are edge-disjoint, and each $P_{\{u,v\}}^*$ has cost at least $MinCost(u, v, l'^*(\{u, v\}))$. Thus the cost of the optimal solution is at least $\sum_{\{u,v\} \in E'^*} MinCost(u, v, l'^*(\{u, v\}))$, and so the algorithm outputs an optimal solution and it runs in polynomial time. ◀

▶ **Corollary 11.** *The arbitrary-length unit-cost* SLSN *problem with parameter p is in* XP.

Proof. We can use the same technique, but instead of guessing the length l' we guess the cost c', and then find shortest path under cost bound c'. We can also use Bellman-Ford algorithm in this step. ◀

3.2 Star Demand Graphs ($\mathrm{SLSN}_{\mathcal{C}^*}$)

We prove Theorem 4 by reducing $\mathrm{SLSN}_{\mathcal{C}^*}$ to DST, which has a known FPT algorithm [10]. This reduction is essentially folklore, but is included in Section 3.2 of the full paper [2] for completeness. This reduction transforms a unit-length arbitrary-cost $\mathrm{SLSN}_{\mathcal{C}^*}$ instance $(G, c, l \equiv 1, \{\{s_1, t_1\}, \ldots, \{s_p, t_p\}\}, L)$ into a DST instance by creating a layered graph G' with $L+1$ layers. Each layer includes $|V|$ vertices (one for each vertex in G). Letting $v^{(i)}$ represent vertex v in layer i, each vertex $v^{(i-1)}$ (for $i \in [1, L]$) is connected to vertex $v^{(i)}$ with a 0-cost edge $(v^{(i-1)}, v^{(i)})$. Each such $v^{(i-1)}$ is also connected to each vertex $u^{(i)}$ such that $(v, u) \in E(G)$ by an edge $(v^{(i-1)}, u^{(i)})$ with cost $c(u, v)$. For the demands of the DST instance, we require the demand-source $s = s_1, \ldots s_p$ of the $\mathrm{SLSN}_{\mathcal{C}^*}$ instance in layer 0 (i.e., $s^{(0)}$) to be connected to layer-L endpoints $t_1^{(L)}, \ldots t_p^{(L)}$, giving us an instance $(G', c', s^{(0)}, t_1^{(L)}, \ldots, t_p^{(L)})$ of DST. We solve this DST instance using the algorithm of [10] and construct a solution to the $\mathrm{SLSN}_{\mathcal{C}^*}$ by including each edge (v, u) such that edge $(v^{(i-1)}, u^{(i)})$ for some layer i appears in the DST solution.

4 W[1]-Hardness for Unit-Length Unit-Cost SLSN

In this section we prove our main hardness result, Theorem 5. We begin with some preliminaries, then give our reduction and proof.

4.1 Preliminaries

We prove Theorem 5 by constructing an FPT reduction from the Multi-Colored Clique (MCC) problem to the unit-length unit-cost $\mathrm{SLSN}_{\mathcal{C}}$ problem for any $\mathcal{C} \nsubseteq \mathcal{C}_\lambda \cup \mathcal{C}^*$. We begin with the MCC problem.

▶ **Definition 12** (Multi-Colored Clique). Given a graph $G = (V, E)$, a number $k \in \mathbb{N}$ and a coloring function $c : V \to [k]$. The objective of the MCC problem is to determine whether there is a clique $T \subseteq V$ in G with $|T| = k$ where $c(x) \neq c(y)$ for all $x, y \in T$.

For each $i \in [k]$, we define $C_i = \{v \in V : c(v) = i\}$ to be the vertices of color i. We can assume that the graph does not contain edges where both endpoints have the same color, since those edges do not affect the existence of a multi-colored clique. It has been proven that the MCC problem is W[1]-complete.

▶ **Theorem 13** ([12]). *The* MCC *problem is* W[1]*-complete with parameter k.*

We first define a few important classes of graphs. These are the major classes that fall outside of $\mathcal{C}^* \cup \mathcal{C}_\lambda$, so we will need to be able to reduce MCC to SLSN where the demand graphs are in these classes, and then this will allow us to prove the hardness for general $C \nsubseteq \mathcal{C}^* \cup \mathcal{C}_\lambda$. For every $k \in \mathbb{N}$, we define the following graph classes. Each of the first four classes is just one graph up to isomorphism, but classes 5 and 6 are sets of graphs, so we use the notation $\mathcal{H}$ instead of H for these classes. Note that each of the first three classes is just a star with an additional edge, so we use * to make this clear.

1. $H^*_{k,0}$: a star with $k(k-1)$ leaves and an edge with both endpoints *not* in the star.
2. $H^*_{k,1}$: a star with $(k(k-1)+1)$ leaves and an edge $\{u,v\}$ where u is a leaf of the star and v is not in the star.
3. $H^*_{k,2}$: a star with $(k(k-1)+2)$ leaves, and an edge $\{u,v\}$ where both u and v are leaves of the star.
4. $H_{k,k}$: $k(k-1)+1$ edges where all endpoints are different (i.e., a matching of size $k(k-1)+1$).
5. $\mathcal{H}_{2,k}$: the class of graphs that have exactly $k(k-1)+2$ vertices, and contain a 2 by $k(k-1)$ complete bipartite subgraph (not necessarily an induced subgraph).
6. $\mathcal{H}_k$: the class of graphs that contain at least one of the graphs in previous five classes as an induced subgraph.

We first prove the following lemma.

▶ **Lemma 14.** *For any $k \geq 2$, if a graph H is not a star and H has at least $8k^{10}$ edges, then $H \in \mathcal{H}_k$, and we can find an induced subgraph which is isomorphic to a graph in $\{H^*_{k,0}, H^*_{k,1}, H^*_{k,2}, H_{k,k}\} \cup \mathcal{H}_{2,k} \cup \mathcal{H}_k$ in poly($|H|$) time.*

Proof. We give a constructive proof. We first claim that either there is a vertex in H which has degree at least $2k^4$ or there is an induced matching in H of size k^2. Suppose that all vertices have degree less than $2k^4$. Then we can create an induced matching by adding an arbitrary edge $\{u,v\} \in H$ to a edge set M, removing all vertices that are adjacent to either u or v from H, and repeating until there are no more edges in H. In each iteration we reduce the total number of edges by at most $2 \cdot 2k^4 \cdot 2k^4$, thus $|M| \geq \frac{8k^{10}}{8k^8} = k^2$. Since when we add an edge $\{u,v\}$ we also remove all vertices adjacent to u or v, every future edge we add to M will have endpoints which are not adjacent to u or v, and thus M is an induced matching of H with size k^2.

If H has an induced matching of size k^2, then $H \in \mathcal{H}_k$ because it contains $H_{k,k}$ as an induced subgraph, and thus we are done.

Otherwise, H has a vertex s with degree at least $2k^4$. Let S be the neighbors of s. If there is any vertex other than s that is adjacent to at least $k(k-1)$ vertices in S, then H contains a 2 by $k(k-1)$ complete bipartite subgraph, so it contains an induced subgraph $H' \in \mathcal{H}_{2,k}$ and thus is in $\mathcal{H}_k$.

So suppose that there is no vertex other than s that is adjacent to at least $k(k-1)$ vertices in S. Consider the case that there is no edge between any pair of vertices in S; then, because H is not a star, there must be an edge $\{u,v\} \in H$ with at least one of u,v not in $S \cup \{s\}$. Since both u and v are adjacent to at most $k(k-1)$ vertices in S, there are at least $k^4 - 2 \cdot k(k-1) \geq k(k-1)$ vertices in S that are not adjacent to either u or v. Let the set of these vertices be T. Then the induced subgraph on vertex set $T \cup \{s,u,v\}$ is either $H^*_{k,0}$ or $H^*_{k,1}$, depending on whether $\{u,v\} \cap T$ is an empty set.

Now the only remaining case is that there is at least one edge in H with both endpoints in S. In this case, we can find $H^*_{k,2}$ as an induced subgraph as follows: We first let $S_0 = S$. Then, in each iteration t we let v_t be a vertex in S_{t-1} that is adjacent to the fewest number of other vertices in S_{t-1}. We add v_t to the vertex set T, and then delete v_t and all the vertices in S_{t-1} that are adjacent to v_t to get S_t. This process repeats until we have $|T| = k(k-1)$.

We can use induction to show that, after each iteration $t \leq k(k-1)$, there is always at least one edge in H where both endpoints are in S_t. The base case is $t = 0$, where such an edge clearly exists. Assume the claim holds for iteration $t-1$, consider the iteration $t \leq k(k-1)$. If v_t is not adjacent to any other vertex in S_{t-1}, then removing v_t does not affect the fact that there is at least one edge left, and thus the claim still holds. Otherwise,

v_t is adjacent to at least one vertex in S_{t-1}. Thus, each vertex in S_{t-1} must be adjacent to at least one vertex in S_{t-1}. Since there is no vertex other than s which is adjacent to at least $k(k-1)$ vertices in S, we know that at most k^2 vertices are deleted in each iteration, and thus there are still at least $2k^4 - k^2 \cdot k(k-1) \geq k^4$ vertices in S_{t-1}. Because removing v_t and its neighbors can only affect the degree of at most $k^2(k-1)^2$ vertices in S_{t-1}, there must still be an edge left between the vertices in S_t.

Let $\{u, v\}$ be one of the edges in H where both endpoints are in S_t, then the induced subgraph on vertex set $T \cup \{s, u, v\}$ is $H^*_{k,2}$. Thus $H \in \mathcal{H}_k$.

It is easy to see that all the previous steps directly find an induced subgraph which is isomorphic to a graph in $\{H^*_{k,0}, H^*_{k,1}, H^*_{k,2}, H_{k,k}\} \cup \mathcal{H}_{2,k} \cup \mathcal{H}_k$ and takes polynomial time, thus the lemma is proved. ◀

4.2 Reduction

In this subsection, we will prove the following reduction theorem.

▶ **Theorem 15.** *Let $(G = (V, E), c)$ be an* MCC *instance with parameter k, and let $H \in \mathcal{H}_k$ be a demand graph. Then a unit-length unit-cost* SLSN *instance (G', L) with demand graph H can be constructed in poly$(|V||H|)$ time, and there exists a function g (computable in time poly$(|H|)$) such that the* MCC *instance has a clique with size k if and only if the* SLSN *instance has a solution with cost $g(H)$.*

In order to prove this theorem, we first introduce a construction for any demand graph $H \in \{H^*_{k,0}, H^*_{k,1}, H^*_{k,2}, H_{k,k}\} \cup \mathcal{H}_{2,k}$, and then use the instances constructed in these cases to construct the instance for general $H \in \mathcal{H}_k$.

The construction for $H \in \mathcal{H}_{2,k}$ is similar to [11], which proves the W[1]-hardness of the DSN problem. We change all the directed edges in their construction to undirected, and add some edges and dummy vertices. This construction is presented in Appendix A.3. To handle $H^*_{k,0}$, $H^*_{k,1}$, $H^*_{k,2}$, and $H_{k,k}$, we need to change this basic construction due to the simplicity of the demand graphs. Because the constructions for these four graphs are quite similar, we first introduce the construction for $H^*_{k,0}$ in Section 4.2.1, and then show how to modify it for $H^*_{k,1}$, $H^*_{k,2}$, and $H_{k,k}$ in Appendix A.2.

4.2.1 Case 1: $H^*_{k,0}$

Given an MCC instance $(G = (V, E), c)$ with parameter k, we create a unit-length and unit-cost SLSN instance (G', L) with demand graph $H^*_{k,0}$ as follows.

We first create a graph G^*_k with integer edge lengths (we will later replace all non-unit length edges by paths). See Figure 1 for an overview of this graph. The vertex set V^*_k contains 6 layers of vertices and another group of vertices. The first layer V_1 is just a root r. The second layer V_2 contains a vertex $z_{\{i,j\}}$ for each $1 \leq i < j \leq k$, so there are $\binom{k}{2}$ vertices. The third layer V_3 contains a vertex z_e for each $e \in E$, so there are $|E|$ vertices. The fourth layer V_4 contains a vertex $x_{v,j}$ for each $v \in V$ and $j \in [k]$ with $j \neq c(v)$, so there are $|V| \cdot (k-1)$ vertices. The fifth layer V_5 again contains a vertex $x'_{v,j}$ for each $v \in V$ and $j \in [k]$ with $j \neq c(v)$. The sixth layer V_6 contains a vertex $l_{i,j}$ for each $i, j \in [k]$ where $i \neq j$, so there are $k(k-1)$ vertices. Finally, we have a vertex y_i for $i = 0, \ldots, k$, so there are $k+1$ vertices in the set V_y.

Let $f_i : \mathbb{N} \to \mathbb{N}$ be the function defined by $f_i(j) = j+1$ if $j+1 \neq i$ and $f_i(j) = j+2$ if $j+1 = i$. This function gives the next integer after j, but skips i. Let $f^t_i(j) = f_i(f_i(\ldots f_i(j)))$

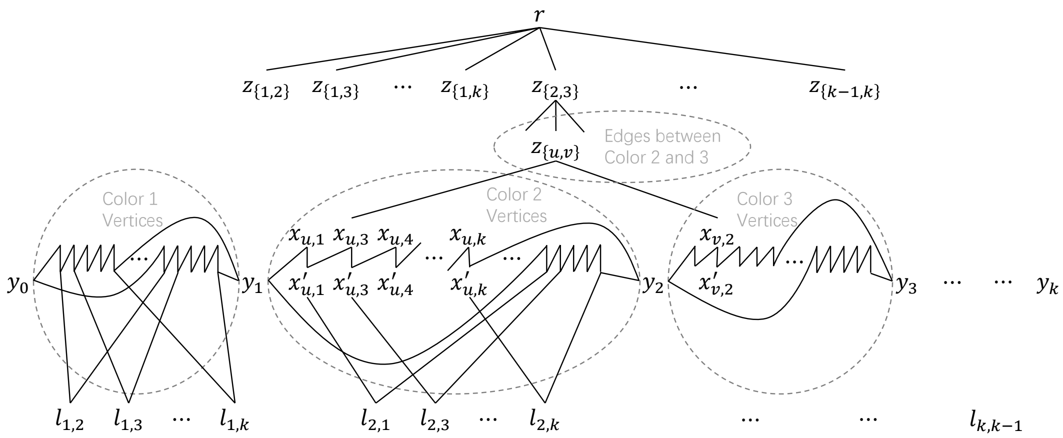

Figure 1 G_k^*.

denote this function repeated t times. Recall that $C_i = \{v \in V : c(v) = i\}$. The edge set E_k^* contains following edges, with lengths as indicated:

- $E_1 = \{\{r, z_{\{i,j\}}\} \mid 1 \le i < j \le k\}$, each edge in E_1 has length 2.
- $E_2 = \{\{z_{\{c(u),c(v)\}}, z_e\} \mid e = \{u,v\} \in E\}$, each edge in E_2 has length 1.
- $E_3 = \{\{z_e, x_{u,c(v)}\} \mid e = \{u,v\} \in E\}$, each edge in E_3 has length $2k^2 - 2$. Note that if $\{z_e, x_{u,c(v)}\} \in E_3$, then $\{z_e, x_{v,c(u)}\} \in E_3$
- $E_4 = \{\{x_{v,j}, x'_{v,j}\} \mid v \in V, j \ne c(v)\}$, each edge in E_4 has length 1.
- $E_5 = \{\{x'_{v,j}, l_{c(v),j}\} \mid v \in V, j \ne c(v)\}$, each edge in E_5 has length $2k^2 - 2$.
- $E_{yx} = \{\{y_{i-1}, x_{v,f_i(0)}\} \mid i \in [k], v \in C_i\}$, each edge in E_{yx} has length 4.
- $E_{xx} = \{\{x'_{v,j}, x_{v,f_{c(v)}(j)}\} \mid v \in V, j \in [k] \setminus \{c(v), f^{k-1}_{c(v)}(0)\}\}$, each edge in E_{xx} has length 3.
- $E_{xy} = \{\{x'_{v,f_i^{k-1}(0)}, y_i\} \mid i \in [k], v \in C_i\}$, each edge in E_{xy} has length 3.

Let G' be the graph obtained from G_k^* by replacing each edge $e \in E_k^*$ by a *length*(e)-hop path. We create an instance of SLSN on G' by setting the demands to be $\{r, l_{i,j}\}$ for all $i, j \in [k]$ where $i \ne j$, as well as $\{y_0, y_k\}$. Note that these demands form a star with $k(k-1)$ leaves and an edge with both endpoints not in the star, so it is isomorphic to $H^*_{k,0}$. We set the distance bound L to be $4k^2$.

This construction clearly takes $poly(|V||H^*_{k,0}|)$ time. Let $g(H^*_{k,0}) = 4k^4 - 4k^3 + \frac{3}{2}k^2 + \frac{5}{2}k$, which is clearly computable in $poly(H^*_{k,0})$ time. We will first prove the easy direction in the correctness of the construction.

▶ **Lemma 16.** *If there is a multi-colored clique of size k in G, then there is a solution S for the* SLSN *instance (G', L) with demand graph $H^*_{k,0}$, and the total cost of S is $g(H^*_{k,0})$.*

Proof. Let $v_1, \ldots, v_k$ be a multi-colored clique of size k in G, where $v_i \in C_i$ for all $i \in [k]$. We create a feasible solution S to our SLSN instance, which contains following paths in G' (i.e., edges in G_k^*):

- $\{r, z_{\{i,j\}}\}$ for each $1 \le i < j \le k$. The total cost of these edges is $2 \cdot \binom{k}{2} = k^2 - k$.
- $\{z_{\{i,j\}}, z_{\{v_i,v_j\}}\}$ for each $1 \le i < j \le k$. The total cost of these edges is $\binom{k}{2} = \frac{k^2-k}{2}$.
- $\{z_{\{v_i,v_j\}}, x_{v_i,j}\}$ and $\{z_{\{v_i,v_j\}}, x_{v_j,i}\}$ for each $1 \le i < j \le k$. The total cost of these edges is $2 \cdot (2k^2 - 2) \cdot \binom{k}{2} = 2k^4 - 2k^3 - 2k^2 + 2k$.
- $\{x_{v_i,j}, x'_{v_i,j}\}$ for each $i, j \in [k]$ where $i \ne j$. The total cost of these edges is $2 \cdot \binom{k}{2} = k^2 - k$.
- $\{x'_{v_i,j}, l_{i,j}\}$ for each $i, j \in [k]$ where $i \ne j$. The total cost of these edges is $2 \cdot (2k^2 - 2) \cdot \binom{k}{2} = 2k^4 - 2k^3 - 2k^2 + 2k$.

- $\{y_{i-1}, x_{v_i,f_i(0)}\}$ for each $i \in [k]$. The total cost of these edges is $4k$.
- $\{x'_{v_i,j}, x_{v_i,f_i(j)}\}$ for each $i \in [k]$ and $j \in [k] \setminus \{i, f_i^{k-1}(0)\}$. The total cost of these edges is $3 \cdot k(k-2) = 3k^2 - 6k$.
- $\{x'_{v_i,f_i^{k-1}(0)}, y_i\}$ for each $i \in [k]$. The total cost of these edges is $3k$.

Therefore, the total cost is $k^2 - k + \frac{k^2-k}{2} + 2k^4 - 2k^3 - 2k^2 + 2k + k^2 - k + 2k^4 - 2k^3 - 2k^2 + 2k + 4k + 3k^2 - 6k + 3k = 4k^4 - 4k^3 + \frac{3}{2}k^2 + \frac{5}{2}k = g(H^*_{k,0})$.

Now we show the feasibility of this solution. For each $i, j \in [k]$ where $i \neq j$, the path between r and $l_{i,j}$ is $r - z_{\{i,j\}} - z_{\{v_i,v_j\}} - x_{v_i,j} - x'_{v_i,j} - l_{i,j}$. The length of this path is $2 + 1 + 2k^2 - 2 + 1 + 2k^2 - 2 = 4k^2$, thus it is a feasible path.

The path between y_0 and y_k is $y_0 - x_{v_1,2} - x'_{v_1,2} - x_{v_1,3} - x'_{v_1,3} - \ldots - x_{v_1,k} - x'_{v_1,k} - y_1 - x_{v_2,1} - x'_{v_2,1} - x_{v_2,3} - x'_{v_2,3} - \ldots - y_2 - \ldots - y_k$. The length of this path is $(4 + 1 \cdot (k-1) + 3 \cdot (k-2) + 3) \cdot k = 4k^2$, thus it is a feasible path. ◀

For the other direction, we begin the proof with a few claims. We first show that the only feasible way to connect r and $l_{i,j}$ is to pick one edge between every two adjacent layers. We can also see in Figure 1 that for each $i \in [k]$, there are $|C_i|$ disjoint "zig-zag" paths between y_{i-1} and y_i, and each path corresponds to a vertex with color i. We will also show that the only feasible way to connect y_0 and y_k is to pick one zig-zag path between each y_{i-1} and y_i. The proof of these claims are in Appendix A.1. From these claims we can then prove that, if the cost of the optimal solution is at most $g(H^*_{k,0})$, then there is a multi-colored clique in G.

▶ **Claim 17.** *For all $i, j \in [k]$ where $i \neq j$, any path $P_{i,j}$ between r and $l_{i,j}$ with length at most $4k^2$ must be of the form $r - z_{\{i,j\}} - z_{\{u,v\}} - x_{u,j} - x'_{u,j} - l_{i,j}$, where $u \in C_i$, $v \in C_j$ and $\{u, v\} \in E$.*

▶ **Claim 18.** *Any path P_y between y_0 and y_k with length at most $4k^2$ can be divided to k subpaths as follows. For each $i \in [k]$, there is a subpath P_{v_i} between y_{i-1} and y_i with length $4k$, of the form $y_{i-1} - x_{v_i,f_i(0)} - x'_{v_i,f_i(0)} - x_{v_i,f_i^2(0)} - x'_{v_i,f_i^2(0)} - \cdots - x_{v_i,f_i^{k-1}(0)} - x'_{v_i,f_i^{k-1}(0)} - y_i$, where $v_i \in C_i$.*

Now, we can prove the other direction in the correctness of the construction.

▶ **Lemma 19.** *Let S be an optimal solution for the* SLSN *instance (G', L) with demand graph $H^*_{k,0}$. If S has cost at most $g(H^*_{k,0}) = 4k^4 - 4k^3 + \frac{3}{2}k^2 + \frac{5}{2}k$, then there is a multi-colored clique of size k in G.*

Proof. For each $i, j \in [k]$ with $i \neq j$, let $P_{i,j}$ be a (arbitrarily chosen) path in S which connects r and $l_{i,j}$ with length at most $L = 4k^2$. Let $\mathcal{P} = \{P_{i,j} \mid i, j \in [k], i \neq j\}$ be the set of all these paths. We also let P_y be a (arbitrary) path in S of length at most L which connects y_0 and y_k.

From Claim 18, P_y can be divided to k subpaths, each of which corresponds to a vertex v_i. We will show that $v_1, \ldots, v_k$ form a clique in G (i.e., for each $1 \leq i < j \leq k$, we have $\{v_i, v_j\} \in E$).

We first prove that these paths must share certain edges due to the cost bound of the optimal solution. From Claim 17, we know that each $P_{i,j}$ costs exactly $2 + 1 + 2k^2 - 2 + 1 + 2k^2 - 2 = 4k^2$. In addition, from the form of $P_{i,j}$ we can also see that these paths are almost disjoint, except that $P_{i,j}$ and $P_{j,i}$ may share a length 2 edge $\{r, z_{\{i,j\}}\} \in E_1$ and a length 1 edge $\{z_{\{i,j\}}, z_e\} \in E_2$. Therefore, in order to satisfy the demands between r and all of the $l_{i,j}$'s, the total cost of the edges in $S \cap (E_1 \cup E_2 \cup E_3 \cup E_4 \cup E_5)$ is at least

$4k^2 \cdot k(k-1) - \binom{k}{2} \cdot (2+1) = 4k^4 - 4k^3 - \frac{3}{2}k^2 + \frac{3}{2}k$, even if every $P_{i,j}$ and $P_{j,i}$ do share edge $\{r, z_{\{i,j\}}\}$ and edge $\{z_{\{i,j\}}, z_e\}$.

We now calculate the cost of the edges in $S \cap (E_{yx} \cup E_{xx} \cup E_{xy})$. From Claim 18, the total cost of edges in $P_y \cap (E_{yx} \cup E_{xx} \cup E_{xy})$ is at least $(4+3\cdot(k-1)+3)\cdot k = 3k^2+k$. Thus, the total cost is already at least $\left(4k^4 - 4k^3 - \frac{3}{2}k^2 + \frac{3}{2}k\right) + (3k^2+k) = 4k^4 - 4k^3 + \frac{3}{2}k^2 + \frac{5}{2}k = g(H^*_{k,0})$, so S cannot contain any edge which has not been counted yet.

Therefore, every edge in $P_y \cap E_4$ must appear in some path in $\mathcal{P}$. In fact, by the form of the paths in $\mathcal{P}$, we can see that for each $i, j \in [k]$ where $i \neq j$, the edge $\{x_{v_i,j}, x'_{v_i,j}\} \in P_y \cap E_4$ can only appear in path $P_{i,j}$, rather than any other $P_{i',j'}$. Thus $x_{v_i,j}$ is in path $P_{i,j}$, and similarly $x_{v_j,i}$ is in path $P_{j,i}$. Recall that $P_{i,j}$ and $P_{j,i}$ must share an edge $\{z_{\{i,j\}}, z_e\}$ for some $e \in E$ because of the cost bound, and $z_{\{v_i,v_j\}}$ is the only vertex which adjacent to both $x_{v_i,j}$ and $x_{v_j,i}$, we can see that e can only be $\{v_i, v_j\}$. Therefore $\{v_i, v_j\} \in E$, which proves the lemma. ◀

4.2.2 Cases 2: $H^*_{k,1}$, 3: $H^*_{k,2}$, 4: $H_{k,k}$, and 5: $\mathcal{H}_{2,k}$

For Cases 2, 3, and 4, we can use essentially the same reduction as in Case 1. For Case 2, we just need to add a new demand $\{r, y_0\}$, and do some extra analysis to show that adding this demand does not change anything. For Case 3, we similarly add another demand $\{r, y_k\}$. Case 4 requires only adding another layer of vertices and edges before the root r. The details are in Appendix A.2. Case 5 is a variant of the reduction in [11], and we prove this case in Appendix A.3.

4.2.3 Case 6: $\mathcal{H}_k$

We now want to construct an SLSN instance for a demand graph $H \in \mathcal{H}_k$ from an MCC instance $(G = (V, E), c)$ with parameter k; since all other cases have been handled, this will complete the proof of Theorem 15. By the definition of $\mathcal{H}_k$, for some $t \in [5]$ there is a graph $H^{(t)}$ of Case t that is an induced subgraph of H. We use Lemma 14 to find the graph $H^{(t)}$. Let $(G^{(t)}, L)$ be the SLSN instance obtained by applying our reduction for Case t to the MCC instance (G, c), and let the corresponding function be $g^{(t)}$.

We now want to transform the SLSN instance $(G^{(t)}, L)$ with demand graph $H^{(t)}$ into a new SLSN instance (G', L) with demand graph H, so that instance $(G^{(t)}, L, H^{(t)})$ has a solution with cost $g^{(t)}(H^{(t)})$ if and only if instance (G', L, H) has a solution with cost $g(H) = g^{(t)}(H^{(t)}) + L \cdot (|H| - |H^{(t)}|)$. If there is such a construction which runs in polynomial time, then there is a multi-colored clique of size k in G if and only if instance (G', L, H) has a solution with cost $g(H)$. This will then imply Theorem 15.

The graph G' is basically just $G^{(t)}$ with some additional vertices and edges from $H \setminus H^{(t)}$. For each vertex v in H but not in $H^{(t)}$, we add a new vertex v to G'. For each edge $\{u, v\} \in H \setminus H^{(t)}$, we add an L-hop path between u and v to G'.

The construction still takes $poly(|V||H|)$ time, because the construction for the previous cases takes $poly(|V||H^{(t)}|)$ time and the construction for Case 6 takes $poly(|G^{(t)}||H|)$ time. Here $|H^{(t)}| \leq |H|$, and we know that $|G^{(t)}|$ is polynomial in $|V|$ and $|H^{(t)}|$.

▶ **Lemma 20.** SLSN *instance $(G^{(t)}, L, H^{(t)})$ has a solution with cost $g^{(t)}(H^{(t)})$ if and only if instance (G', L, H) has a solution with cost $g(H) = g^{(t)}(H^{(t)}) + L \cdot (|H| - |H^{(t)}|)$.*

Proof. If instance $(G^{(t)}, L, H^{(t)})$ has a solution with cost $g^{(t)}(H^{(t)})$. Let the solution be $S^{(t)}$. For each $e = \{u, v\} \in H \setminus H^{(t)}$, let the new L-hop path between u and v in G' be P_e. Then $S^{(t)} \cup \bigcup_{e \in H \setminus H^{(t)}} P_e$ is a solution to G' with cost $g^{(t)}(H^{(t)}) + L \cdot (|H| - |H^{(t)}|)$.

If instance (G', L, H) has a solution with cost $g^{(t)}(H^{(t)}) + L \cdot (|H| - |H^{(t)}|)$, let the solution be S. Since for each $e = \{u, v\} \in H \setminus H^{(t)}$, the only path between u and v in G' within the length bound is the new L-hop path P_e, any valid solution must include all these P_e, which has total cost $L \cdot (|H| - |H^{(t)}|)$. In addition, for each demand $\{u, v\}$ which is also in H, any path between u and v in G' within the length bound will not include any new edge, because otherwise it will strictly contain an L-hop path, and have length more than L. Therefore, $S \setminus \bigcup_{e \in H \setminus H^{(t)}} P_e$ is a solution to $G^{(t)}$ with cost $g^{(t)}(H^{(t)})$. ◀

4.3 Proof of Theorem 5

If $\mathcal{C}$ is a recursively enumerable class, and $\mathcal{C} \nsubseteq \mathcal{C}_\lambda \cup \mathcal{C}^*$ for any constant λ, then for every $k \geq 2$, let H_k be the first graph in $\mathcal{C}$ where H_k is not a star and has at least $2k^{10}$ edges. The time for finding H_k is $f(k)$ for some function f. From Lemma 14 we know that $H_k \in \mathcal{H}_k$, so that we can use Theorem 15 to construct the $\text{SLSN}_\mathcal{C}$ instance with demand H_k.

The parameter $p = |H_k|$ of the instance is a function just of k, and the construction time is FPT from Theorem 15. Therefore this is a FPT reduction from the MCC problem to the unit-length unit-cost $\text{SLSN}_\mathcal{C}$ problem. Thus Theorem 13 implies that the unit-length unit-cost $\text{SLSN}_\mathcal{C}$ problem is W[1]-hard for parameter p. ◀

5 Overview of General Length and Cost Settings

As discussed in Section 2, we extended our results from the unit-length setting to the general length setting. We defer all detailed results to Section 5 and 6 of the full paper [2], and instead give only a brief overview of our results and techniques.

5.1 Upper bounds

Recall that we cannot have an exact FPT algorithm for $\text{SLSN}_{\mathcal{C}^*}$ and $\text{SLSN}_{\mathcal{C}_\lambda}$ since even if there is only a single demand the problem becomes the Restricted Shortest Path problem, which is known to be NP-hard [15]. But since Restricted Shortest Path admits an FPTAS [15, 17], it is natural to instead try to give a $(1 + \varepsilon)$-approximation algorithm for both problems. We show that with some modifications of the algorithms in the unit-length case, we can give an FPTAS for arbitrary-length arbitrary-cost $\text{SLSN}_{\mathcal{C}_\lambda}$, and can give a $(1 + \varepsilon)$-approximation algorithm in FPT time for arbitrary-length arbitrary-cost $\text{SLSN}_{\mathcal{C}^*}$.

For $\text{SLSN}_{\mathcal{C}_\lambda}$, Lemma 9 still holds, so we can still guess how the paths in the solution intersect with each other and what the endpoints of maximum shared subpaths are. However, we cannot guess the length of a subpath in this setting, since there are too many possibilities. We instead guess the cost of all the subpaths. Because we are aiming to find an approximation solution, we are allowed to have $(1 + \varepsilon)$ error on the cost of each subpath, so this allows us to reduce the search space. However, this is still not enough: if the space of the possible values is too large, then $\log_{1+\varepsilon}$ of it is still too large. So we then use an additional procedure from [17] which gives valid upper bound U and lower bound L on the optimal solution such that $U/L \leq n^2$. This sufficiently decreases the space of possible guesses so that we get a $(1 + \varepsilon)$-approximation in polynomial time. The full algorithm and analysis are in Section 5.2 of the full paper [2].

For the star demand graph, we cannot do the same reduction as in Section 3.2 because with arbitrary lengths the natural layered graph used in the reduction to DSN can have exponential layers. However, similar to Steiner Tree and DST, one can prove that the optimal solution for $\text{SLSN}_{\mathcal{C}^*}$ is always a tree. Therefore we look at the original FPT algorithm for Steiner Tree and DST and attempt to modify it to work in our setting. Given a star demand graph where the center is s and the leaf set is T, both algorithms use dynamic programming to solve the subproblems $f(v, R)$, which are to find the minimum cost tree with root $v \in V$ that contains $R \subseteq T$, starting from $|R| = 1$ to $|R| = |T|$. The base case when $|R| = 1$ is essentially a shortest path algorithm. Then we can build up larger trees since a tree with more than two leaves can always be partitioned to two subtrees and a path from the root.

We use a similar approach, first discretizing the possible costs to be powers of $(1 + \varepsilon)$. We define the subproblem $d(v, j, R)$ to be the smallest height of a tree (with the given edge lengths) such that the root is v, the total cost is at most j, and it contains all vertices in R. Then, we find the smallest j for which $d(s, j, T)$ is at most the length bound L, and this j is actually a good approximation to the optimal solution. The full algorithm and analysis are in Section 5.3 of the full paper [2].

5.2 Lower bounds

For the lower bound on $\text{SLSN}_{\mathcal{C}}$ with $\mathcal{C} \not\subseteq (\mathcal{C}_\lambda \cup \mathcal{C}^*)$, the same reduction as in Section 4 already shows that it is W[1]-hard to obtain a $\left(1 + \frac{1}{O(p^2)}\right)$-approximation. However, we would like a stronger hardness of approximation, one which would rule out good approximations (like we gave for $\text{SLSN}_{\mathcal{C}^*}$ and $\text{SLSN}_{\mathcal{C}_\lambda}$) even for large p. With some modifications of the cost of some edges in the instance constructed in Section 4, and a stronger assumption of Gap-ETH, we can show that there is no $(\frac{5}{4} - \varepsilon)$-approximation for $\text{SLSN}_{\mathcal{C}}$ which runs in FPT time, even for the unit-length polynomial-cost setting.

Consider the reduction in Section 4. We showed that if there is a low-cost solution to the SLSN instance that we created, then the paths satisfying the demands must share some specific edges with each other, and the existence of these edges implies the existence of a clique in the given MCC instance. For the polynomial-cost setting, we reduce from a different problem known as the Multi-Colored Densest k-Subgraph, which is a gap version of the MCC instance. Under the assumption of Gap-ETH, a corollary of [7] shows that for any constant $0 < \varepsilon < 1$, no FPT algorithm can distinguish between the case that there is a multi-colored k-clique and the case that every subgraph with k vertices has at most $\varepsilon \cdot \binom{k}{2}$ edges. By modifying the cost of some edges and making a slightly delicate inclusion-exclusion argument, we can show that if the cost of the SLSN solution is not too large then many edges still need to be shared by different paths, which ensures that a subgraph with k vertices and $\varepsilon \cdot \binom{k}{2}$ edges must exist. The entire reduction and the correctness proof is in Section 6 of the full paper [2].

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, SODA 2018*, pages 1865–1883, 2018.

2 Amy Babay, Michael Dinitz, and Zeyu Zhang. Characterizing Demand Graphs for (Fixed-Parameter) Shallow-Light Steiner Network. *arXiv preprint arXiv:1802.10566*, 2018.

3 Amy Babay, Emily Wagner, Michael Dinitz, and Yair Amir. Timely, Reliable, and Cost-Effective Internet Transport Service Using Dissemination Graphs. In *37th IEEE International Conference on Distributed Computing Systems, ICDCS 2017*, pages 1–12, 2017.

4 Parinya Chalermsook, Marek Cygan, Guy Kortsarz, Bundit Laekhanukit, Pasin Manurangsi, Danupon Nanongkai, and Luca Trevisan. From gap-ETH to FPT-inapproximability: Clique, dominating set, and more. In *Foundations of Computer Science (FOCS), 2017 IEEE 58th Annual Symposium on*, pages 743–754. IEEE, 2017.

5 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.

6 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):18, 2011.

7 Rajesh Chitnis, Andreas Emil Feldmann, and Pasin Manurangsi. Parameterized Approximation Algorithms for Directed Steiner Network Problems. *arXiv preprint arXiv:1707.06499*, 2017.

8 Eden Chlamtác, Michael Dinitz, Guy Kortsarz, and Bundit Laekhanukit. Approximating Spanners and Directed Steiner Forest: Upper and Lower Bounds. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017*, pages 534–553, 2017.

9 Stuart E Dreyfus and Robert A Wagner. The Steiner problem in graphs. *Networks*, 1(3):195–207, 1971.

10 Jon Feldman and Matthias Ruhl. The directed Steiner network problem is tractable for a constant number of terminals. *SIAM Journal on Computing*, 36(2):543–561, 2006.

11 Andreas Emil Feldmann and Dániel Marx. The Complexity Landscape of Fixed-Parameter Directed Steiner Network Problems. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016*, volume 55 of *LIPIcs*, pages 27:1–27:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

12 Michael R Fellows, Danny Hermelin, Frances Rosamond, and Stéphane Vialette. On the parameterized complexity of multiple-interval graph problems. *Theoretical Computer Science*, 410(1):53–61, 2009.

13 Longkun Guo, Kewen Liao, and Hong Shen. On the shallow-light steiner tree problem. In *Parallel and Distributed Computing, Applications and Technologies (PDCAT), 2014 15th International Conference on*, pages 56–60. IEEE, 2014.

14 Mohammad Taghi Hajiaghayi, Guy Kortsarz, and Mohammad R Salavatipour. Approximating buy-at-bulk and shallow-light k-Steiner trees. *Algorithmica*, 53(1):89–103, 2009.

15 Refael Hassin. Approximation schemes for the restricted shortest path problem. *Mathematics of Operations research*, 17(1):36–42, 1992.

16 Guy Kortsarz and David Peleg. Approximating shallow-light trees. Technical report, Association for Computing Machinery, New York, NY (United States), 1997.

17 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.

18 Joseph Naor and Baruch Schieber. Improved approximations for shallow-light spanning trees. In *Foundations of Computer Science, 1997. Proceedings., 38th Annual Symposium on*, pages 536–541. IEEE, 1997.

19 Leonid Zosin and Samir Khuller. On directed Steiner trees. In *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 59–63. Society for Industrial and Applied Mathematics, 2002.

A Proofs in Section 4

A.1 Proof of Claims in Case 1

A.1.1 Proof of Claim 17

We can see that G_k^* is a 6-layer graph with a few additional paths between the fourth layer and the fifth layer. Thus $P_{i,j}$ must contain at least one edge between each two adjacent layers. From the construction of G_k^*, all the edges between two adjacent layers have the same length. If we sum up the length from r to the fourth layer plus the length from the fifth layer to $l_{i,j}$, it is already $2+1+2k^2-2+2k^2-2=4k^2-1$. Thus, between the fourth layer and the fifth layer we can only choose one length 1 edge.

We know that the vertex in the fifth layer must adjacent to $l_{i,j}$, so it must be $x'_{u,j}$ for some $u \in C_i$. Thus, the edge between the fourth layer and the fifth layer must be $\{x_{u,j}, x'_{u,j}\}$, because this is the only length 1 edge adjacent to $x'_{u,j}$. In addition, the only way to go from r to $x_{u,j}$ with one edge per layer is to pass through vertex $z_{\{i,j\}}$ and $z_{\{u,v\}}$ for some $v \in C_j$ and $\{u,v\} \in E$. Therefore $P_{i,j}$ must correspond to an edge $\{u,v\} \in E$ where $u \in C_i$ and $v \in C_j$, and it has form $r - z_{\{i,j\}} - z_{\{u,v\}} - x_{u,j} - x'_{u,j} - l_{i,j}$. ◀

A.1.2 Proof of Claim 18

For the path connecting y_0 and y_k, we first prove another claim.

▶ **Claim 21.** *Any path P_y between y_0 and y_k with length at most $4k^2$ does not contain any edge in $E_1 \cup E_2 \cup E_3 \cup E_5$.*

Proof. We prove the claim by contradiction. If P_y contains an edge in $E_1 \cup E_2 \cup E_3 \cup E_5$, it must contain at least two edges with length $2k^2-2$ (one edge to go out of the fourth and the fifth layer, and another one to go back). Since any edge which has endpoint y_0 has length 4 and any edge which has endpoint y_k has length 3, the total length $2\cdot(2k^2-2)+4+3=4k^2+3$ already exceeds the length bound $4k^2$, giving a contradiction. ◀

Since we have Claim 21, it suffices to consider the edge set $E_4 \cup E_{yx} \cup E_{xx} \cup E_{xy}$. We can see that $E_4 \cup E_{yx} \cup E_{xx} \cup E_{xy}$ can be partitioned to $k|V|$ paths, where for each $i \in [k]$ and each $v \in C_i$, there is a path P_v which connects y_{i-1} and y_i with length $4k$. The path is $y_{i-1} - x_{v,f_i(0)} - x'_{v,f_i(0)} - x_{v,f_i^2(0)} - x'_{v,f_i^2(0)} - \dots - x_{v,f_i^{k-1}(0)} - x'_{v,f_i^{k-1}(0)} - y_1$. We can see that these paths are vertex disjoint except for the endpoints $y_0, y_1, \dots, y_k$.

Therefore, the only way to go from y_0 to y_k is by passing through $y_0, y_1, \dots, y_k$ one-by-one. Thus, for each $i \in [k]$, P_y must contain a subpath P_{v_i} where $v_i \in C_i$. Because each of these subpaths has length $4k$, the total cost is already $4k \cdot k = 4k^2$, which is exactly the length bound. Therefore, P_y can not contain any other edge, which proves the lemma. ◀

A.2 Case 2, 3, and 4

Cases 2, 3, and 4 are basically the same as Case 1, so we discuss them in the same subsection.

Case 2: $H_{k,1}^*$

We use the same G_k^*, G', and L in the construction of the SLSN instance for demand graph $H_{k,0}^*$, and also set $g(H_{k,1}^*) = 4k^4 - 4k^3 + \frac{3}{2}k^2 + \frac{5}{2}k$. The only difference is the demand graph. Besides the demand of $\{r, l_{i,j}\}$ for all $i,j \in [k]$ where $i \neq j$, and $\{y_0, y_k\}$, there is a new demand $\{r, y_0\}$. Clearly this new demand graph is a star with $(k(k-1)+1)$ leaves, and an edge in which exactly one of the endpoints is a leaf of the star, so it is isomorphic to $H^*_{k,1}$.

Assume there is a multi-colored clique of size k in G. The paths connecting previous demands in the solution of the SLSN instance are the same as Case 1. The path between r and y_0 is $r - z_{\{1,2\}} - z_{\{v_1,v_2\}} - x_{v_1,2} - y_0$. All the edges in this path are already in the previous paths, so the cost remains the same. The length of this path is $2+1+2k^2-2+4 = 2k^2+5 < 4k^2$, which satisfies the length bound.

Assume there is a solution for the SLSN instance $(G', L, H^*_{k,1})$ with cost $4k^4 - 4k^3 + \frac{3}{2}k^2 + \frac{5}{2}k$. The proof that there exists a multi-colored clique of size k in G is the same as Case 1.

Case 3: $H^*_{k,2}$

As in Case 2, only the demand graph changes. The new demand graph is the same as in Case 2 but again with a new demand $\{r, y_k\}$. Since $\{r, y_0\}$ was already a demand, our new demand graph is a star with $(k(k-1)+2)$ leaves (the $l_{i,j}$'s and y_0 and y_k), and an edge between two of its leaves (y_0 and y_k), which is isomorphic to $H^*_{k,2}$.

Assume there is a multi-colored clique of size k in G. The paths connecting previous demands in the solution of the SLSN instance are the same as Case 2. The path between r and y_k is $r - z_{\{k-1,k\}} - z_{\{v_{k-1},v_k\}} - x_{v_k,k-1} - y_k$. All the edges in this path are already in the previous paths, so the cost stays the same. The length of this path is $2+1+2k^2-2+4 = 2k^2+5 < 4k^2$, which satisfies the length bound.

Assume there is a solution for the SLSN instance $(G', L, H^*_{k,2})$ with cost $4k^4 - 4k^3 + \frac{3}{2}k^2 + \frac{5}{2}k$. The proof that there exists a multi-colored clique of size k in G is the same as Case 1.

Case 4: $H_{k,k}$

In order to get $H_{k,k}$ as our demand graph, we have to slightly change the construction from Case 1. We still first make a weighted graph $G_{k,k} = (V_{k,k}, E_{k,k})$ and then transform it to the unit-length unit-cost graph G'. For the vertex set $V_{k,k}$, we add another layer of vertices $V_0 = \{l'_{i,j} \mid i,j \in [k], i \neq j\}$ to V^*_k before the first layer V_1. For the edge set $E_{k,k}$, we include all the edges in E^*_k, but change the length of edges in E_1 to length 1. We also add another edge set $E_0 = \{\{l'_{i,j}, r\} \mid i,j \in [k], i \neq j\}$. Each edge in E_0 has length 1.

The demands are $\{l'_{i,j}, l_{i,j}\}$ for each $i,j \in [k]$ where $i \neq j$, as well as $\{y_0, y_k\}$. This is a matching of size $k(k-1)+1$, which is isomorphic to $H_{k,k}$. We still set the length bound to be $L = 4k^2$, and set $g(H_{k,k}) = 4k^4 - 4k^3 + 2k^2 + 2k$.

If there is a multi-colored clique of size k in G, the construction for the solution in G' is similar to Case 1. For each $i,j \in [k]$ where $i \neq j$, the path between $l'_{i,j}$ and $l_{i,j}$ becomes $l'_{i,j} - r - z_{\{i,j\}} - z_{\{v_i,v_j\}} - x_{v_i,j} - x'_{v_i,j} - l_{i,j}$ (i.e., one more layer before the root r). It is easy to see that the length bound and size bound are still satisfied.

Assume there is a solution for the SLSN instance $(G', L, H_{k,k})$ with cost $4k^4-4k^3+2k^2+2k$. The proof that there exists a multi-colored clique of size k in G is essentially the same as Case 1, except the path between $l'_{i,j}$ and $l_{i,j}$ has one more layer.

A.3 Case 5: $\mathcal{H}_{2,k}$

In this case, we slightly modify the reduction of [11]. We first change all the edges from directed to undirected. In addition, in [11] the demand graph is precisely a 2-by-$k(k-1)$ bipartite graph, but we also handle the generalization in which there may be more demands

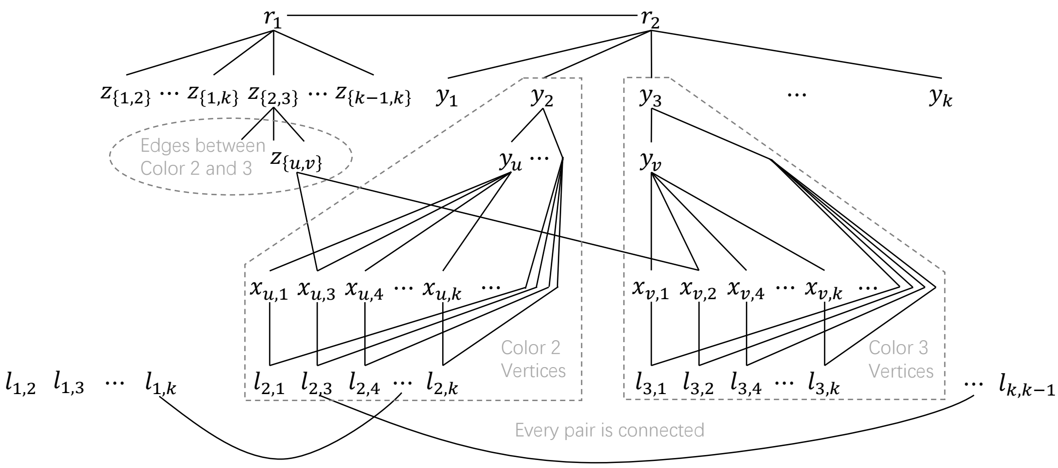

Figure 2 $G_{2,k}$.

between vertices on each sides (i.e., the 2-by-$k(k-1)$ bipartite graph is just a subgraph of our demands). In order to do this, we add some dummy vertices and some edges.

Given an MCC instance $(G = (V, E), c)$ with parameter k, and a demand graph $H \in \mathcal{H}_{2,k}$, we create a unit-length and unit-cost SLSN instance G' with demand isomorphic to H as follows.

We first create a weighted graph $G_{2,k} = (V_{2,k}, E_{2,k})$. The vertex set $V_{2,k}$ contains 5 layers of vertices. The first layer V_1 is just two roots r_1, r_2. The second layer V_2 contains a vertex $z_{\{i,j\}}$ for each $1 \le i < j \le k$, and a vertex y_i for each $i \in [k]$. The third layer V_3 contains a vertex z_e for each $e \in E$, and a vertex y_v for each $v \in V$. The fourth layer V_4 contains a vertex $x_{v,j}$ for each $v \in V$ and $j \neq c(v)$. The fifth layer V_5 contains a vertex $l_{i,j}$ for each $i, j \in [k]$ where $i \neq j$.

The edge set $E_{2,k}$ contains the following edges:

- $E_{11} = \{\{r_1, z_{\{i,j\}}\}, 1 \le i < j \le k\}$, each edge in E_{11} has length 1.
- $E_{12} = \{\{z_{\{c(u),c(v)\}}, z_e\} \mid e = \{u, v\} \in E\}$, each edge in E_{12} has length 1.
- $E_{13} = \{\{z_e, x_{u,c(v)}\} \mid e = \{u, v\} \in E\}$, each edge in E_{13} has length 1. Note that if $\{z_e, x_{u,c(v)}\} \in E_{13}$, then $\{z_e, x_{v,c(u)}\} \in E_{13}$
- $E_{21} = \{\{r_2, y_i\} \mid i \in [k]\}$, each edge in E_{21} has length 1.
- $E_{22} = \{\{y_{c(v)}, y_v\} \mid v \in V\}$, each edge in E_{22} has length 1.
- $E_{23} = \{\{y_v, x_{v,j}\} \mid v \in V, j \neq c(v)\}$, each edge in E_{23} has length 1.
- $E_{xl} = \{\{x_{v,j}, l_{c(v),j}\} \mid v \in V, j \neq c(v)\}$, each edge in E_{xl} has length 4.
- $E_{ll} = \{\{l_{i,j}, l_{i',j'}\} \mid i, j, i', j' \in [k], i \neq j, i' \neq j', (i, j) \neq (i', j')\}$, each edge in E_{ll} has length 7.

We get a unit-length graph G' from $G_{2,k}$ by replacing every edge $e \in E_{2,k}$ by a $length(e)$-hop path. Our SLSN instance consists of the graph G', length bound $L = 7$, and the following demands (which will be isomorphic to H). For each $r \in \{r_1, r_2\}$ and $i, j \in [k]$ with $i \neq j$, there is a demand between r and $l_{i,j}$ (note that these demands form a 2 by $k(k-1)$ complete bipartite graph. Let this complete bipartite subgraph be B. For the rest of the demands, we arbitrarily choose a mapping between $V_1 = \{r_1, r_2\}$ and the 2-side of the bipartite graph in H, as well as a mapping between $V_5 = \{l_{i,j} \mid i, j \in [k], i \neq j\}$ and the $k(k-1)$-side. There is a demand between two vertices $u, v \in V_1 \cup V_5$ if there is an edge between u, v in H.

This construction clearly takes $poly(|V||H|)$ time. Let $g(H) = 7|H| - 7k^2 + 9k - 7 \cdot \mathbb{1}_{\{r_1,r_2\}\in H}$, where $\mathbb{1}_{\{r_1,r_2\}\in H}$ is an indicator variable for $\{r_1, r_2\}$ being a demand in H. This function is also computable in time $poly(|H|)$. We first prove the easy direction in the correctness of the reduction.

▶ **Lemma 22.** *If there is a multi-colored clique of size k in G, then there is a solution S for the* SLSN *instance (G', L) with demand graph $H \in \mathcal{H}_{2,k}$, and the total cost of S is $7|H| - 7k^2 + 9k - 7 \cdot \mathbb{1}_{\{r_1,r_2\}\in H}$.*

Proof. Let $v_1, \ldots, v_k$ be a multi-colored clique of size k in G, where $v_i \in C_i$ for all $i \in [k]$. We create a feasible solution S to our SLSN instance, which contains following paths in G' (i.e., edges in $G_{2,k}$):

- $\{r_1, z_{\{i,j\}}\}$ for each $1 \le i < j \le k$. The total cost of these edges is $\binom{k}{2} = \frac{k^2-k}{2}$.
- $\{z_{\{i,j\}}, z_{\{v_i,v_j\}}\}$ for each $1 \le i < j \le k$. The total cost of these edges is $\binom{k}{2} = \frac{k^2-k}{2}$.
- $\{z_{\{v_i,v_j\}}, x_{v_i,j}\}$ and $\{z_{\{v_i,v_j\}}, x_{v_j,i}\}$ for each $1 \le i < j \le k$. The total cost of these edges is $2 \cdot \binom{k}{2} = k^2 - k$.
- $\{r_2, y_i\}$ for each $i \in [k]$. The total cost of these edges is k.
- $\{y_i, y_{v_i}\}$ for each $i \in [k]$. The total cost of these edges is k.
- $\{y_{v_i}, x_{v_i,j}\}$ for each $i, j \in [k]$ where $i \ne j$. The total cost of these edges is $2 \cdot \binom{k}{2} = k^2 - k$.
- $\{x_{v_i,j}, l_{i,j}\}$ for each $i, j \in [k]$ where $i \ne j$. The total cost of these edges is $4 \cdot 2 \cdot \binom{k}{2} = 4k^2 - 4k$.
- $\{u, v\}$ for each $\{u, v\} \in H \setminus (B \cup \{\{r_1, r_2\}\})$. The total cost of these edges is $7 \cdot (|H| - 2 \cdot k(k-1) - \mathbb{1}_{\{r_1,r_2\}\in H}) = 7|H| - 14k^2 + 14k - 7 \cdot \mathbb{1}_{\{r_1,r_2\}\in H}$.

Therefore, the total cost is $\frac{k^2-k}{2} + \frac{k^2-k}{2} + k^2 - k + k + k + k^2 - k + 4k^2 - 4k + 7|H| - 14k^2 + 14k - 7 \cdot \mathbb{1}_{\{r_1,r_2\}\in H} = 7|H| - 7k^2 + 9k - 7 \cdot \mathbb{1}_{\{r_1,r_2\}\in H}$.

Now we show the feasibility of this solution. For each $i, j \in [k]$ where $i \ne j$, the path between r_1 and $l_{i,j}$ is $r_1 - z_{\{i,j\}} - z_{\{v_i,v_j\}} - x_{v_i,j} - l_{i,j}$, and the path between r_2 and $l_{i,j}$ is $r_2 - y_i - y_{v_i} - x_{v_i,j} - l_{i,j}$. Both paths have length 7, which is within the length bound. For each $\{u, v\} \in H \setminus (B \cup \{\{r_1, r_2\}\})$, u and v have an edge with length 7, thus a path under the length bound exists. Finally, if there exists a demand between r_1 and r_2, we can follow the path $r_1 - z_{\{1,2\}} - z_{\{v_1,v_2\}} - x_{v_1,2} - y_{v_1} - y_1 - r_2$, which has length 6. ◀

Now we prove the other direction.

Let S be an optimal solution for the SLSN instance (G', L) with demand graph $H^*_{k,0}$. If S has cost at most $4k^4 - 4k^3 + \frac{3}{2}k^2 + \frac{5}{2}k$, then there is a multi-colored clique of size k in G.

▶ **Lemma 23.** *Let S be an optimal solution for the* SLSN *instance (G', L) with demand graph $H \in \mathcal{H}_{2,k}$. If S has cost at most $7|H| - 7k^2 + 9k - 7 \cdot \mathbb{1}_{\{r_1,r_2\}\in H}$, then there is a multi-colored clique of size k in G.*

Proof. For each $i, j \in [k]$ where $i \ne j$, let $P_{1,i,j} \subseteq S$ be a (arbitrarily chosen) path between r_1 and $l_{i,j}$ with length at most 7, and $P_{2,i,j} \subseteq S$ be a (arbitrarily chosen) path between r_2 and $l_{i,j}$ with length at most 7. Let $\mathcal{P}_1 = \{P_{1,i,j} \mid i, j \in [k], i \ne j\}$, and $\mathcal{P}_2 = \{P_{2,i,j} \mid i, j \in [k], i \ne j\}$. As in lemma 19, we first show that some edges must be shared by multiple paths by calculating the total cost.

In order to satisfy the demand for each $\{l_{i,j}, l_{i',j'}\} \in H \setminus (B \cup \{\{r_1, r_2\}\})$, the only way is to use the edge between $l_{i,j}$ and $l_{i',j'}$ in E_{ll}. Otherwise, suppose the path has more than one edge, since the only edges incident on any $l_{i,j}$ have length either 4 or 7, the cost of two of these edges already exceeds the length bound. Thus the total cost of the edges in $S \cap E_{ll}$ is at least $7|H| - 7|B| - 7 \cdot \mathbb{1}_{\{r_1,r_2\}\in H} = 7|H| - 14k^2 + 14k - 7 \cdot \mathbb{1}_{\{r_1,r_2\}\in H}$.

We can see that each of the paths in $\mathcal{P}_1 \cup \mathcal{P}_2$ must have exactly one edge between every two adjacent levels, and they cannot have any other edges because of the length bound. Thus, each path $P_{1,i,j} \in \mathcal{P}_1$ must have form $r_1 - z_{\{i,j\}} - z_{\{u,v\}} - x_{u,j} - l_{i,j}$ for some $\{u,v\} \in E$ with $u \in C_i$ and $v \in C_j$, and each path in $P_{2,i,j} \in \mathcal{P}_2$ must have form $r_2 - y_i - y_v - x_{v,j} - l_{i,j}$ for some $v \in C_i$.

By looking at the form of paths in $\mathcal{P}_1$, we can see that these paths are almost disjoint, except that $P_{1,i,j}$ and $P_{1,j,i}$ may share edge $\{r_1, z_{\{i,j\}}\} \in E_{11}$ and edge $\{z_{\{i,j\}}, z_e\} \in E_{12}$. Since paths in $\mathcal{P}_1$ only contain edges in $E_{11} \cup E_{12} \cup E_{13} \cup E_{xl}$, the cost of edges in $S \cap (E_{11} \cup E_{12} \cup E_{13} \cup E_{xl})$ must be at least $7 \cdot k(k-1) - \binom{k}{2} - \binom{k}{2} = 6k^2 - 6k$, even if every $P_{1,i,j}$ and $P_{1,j,i}$ do share edge $\{r_1, z_{\{i,j\}}\}$ and edge $\{z_{\{i,j\}}, z_e\}$.

We then look at the form of paths in $\mathcal{P}_2$. We can see that the first 3 hops of these paths only contain edges in $E_{21} \cup E_{22} \cup E_{23}$. In addition, these paths are all disjoint on edges in E_{23}. Moreover, in order to reach all $l_{i,j}$ from r_2 within length 7, these paths should contain all edges in E_{21} and at least k edges in E_{22}. Therefore, the total cost of edges in $S \cap (E_{21} \cup E_{22} \cup E_{23})$ should be at least $k(k-1) + k + k = k^2 + k$.

By summing up all these edges, the total cost of edges in S is already at least $7|H| - 14k^2 + 14k - 7 \cdot \mathbb{1}_{\{r_1,r_2\} \in H} + 6k^2 - 6k + k^2 + k = 7|H| - 7k^2 + 9k - 7 \cdot \mathbb{1}_{\{r_1,r_2\} \in H} = g(H)$, which means S cannot contain any edge that has not been counted before.

Therefore, S must contain exactly k edges in E_{22}, and each of these edges must have a different y_i as an endpoint. We let these edges be $\{y_1, y_{v_1}\}, \ldots, \{y_k, y_{v_k}\}$, where $v_i \in C_i$ for all $i \in [k]$. We claim that $v_1, \ldots, v_k$ forms a (multicolored) clique in G.

For each $1 \le i < j \le k$, by looking at the form of paths in $\mathcal{P}_2$, we know that the path $P_{2,i,j}$ must be $r_2 - y_i - y_{v_i} - x_{v_i,j} - l_{i,j}$. Because of the total cost limitation, the edge $\{x_{v_i,j}, l_{i,j}\} \in P_{2,i,j} \cap E_{xl}$ must also appear in some path in $\mathcal{P}_1$. By looking at the form of the paths in $\mathcal{P}_1$, the only possible path is $P_{1,i,j}$. Similarly, path $P_{2,j,i}$ must share edge $\{x_{v_j,i}, l_{j,i}\}$ with $P_{1,j,i}$. Again by looking at the form of the paths in $\mathcal{P}_1$, the edge in $\{z_{\{i,j\}}, z_e\} \in S \cap E_{12}$ which is shared by $P_{1,i,j}$ and $P_{1,j,i}$ must have $e = \{v_i, v_j\}$, which means $\{v_i, v_j\} \in E$.

Therefore, $v_1, \ldots, v_k$ forms a clique in G. ◀